International Journal of Computing Sciences and Technology Studies
https://journals.hu.edu.so/index.php/ijcsts

AI-Based Smart Traffic Monitoring and Violation Detection System: Design and Experimental Evaluation of a Prototype for Mogadishu, Somalia

Abdirahman Mohamed Abdulle, Abdifatah Mursal Mohamed, Dahran Hassan Ali, Ayub Abdi Warsame, Suleiman Ahmed Ibrahim, Mohamed Omar Hassan*
Department of Software Engineering, Faculty of Computer Science and Information Technology, Hormuud University, Mogadishu, Somalia

*Corresponding author's email: dq.sy2012@gmail.com

Abstract
The rapid urbanization and increasing traffic density in Mogadishu necessitate advanced technological interventions to enhance urban mobility and road safety. This study presents the design and implementation of an AI-Based Smart Traffic Monitoring and Violation Detection System, a prototype engineered to automate the identification of traffic infractions, such as driving in the wrong direction and illegal parking. The system architecture integrates a Python-based computer vision pipeline that utilizes the YOLOv8 object detection model for vehicle localization and the EasyOCR engine for automated license plate recognition. Data captured from the simulation environment were processed in real time and stored in a MySQL relational database that supports a PHP-based administrative dashboard designed for law enforcement oversight. This study employed an experimental multi-methodological approach, gathering primary data through experimental simulations within a controlled laboratory environment using miniature road models and toy vehicles. The performance analysis focused on the detection accuracy, OCR recognition rates, and system latency to validate the efficacy of the prototype. The system achieved a vehicle detection accuracy of 95.0%, correctly flagged 92.0% of wrong-direction driving incidents and 90.0% of illegal parking violations, and attained a license-plate OCR recognition rate of 88.5% (rising to over 93.0% for stationary vehicles), with an average end-to-end system latency of 450 milliseconds from violation capture to dashboard alert. The results indicate that the proposed system can help bridge the gap between manual traffic monitoring and automated enforcement, providing a cost-effective framework for smart city infrastructure in developing urban contexts, and indicate that open-source computer vision and OCR tools can deliver near-real-time violation detection even in resource-constrained environments.

Keywords: Artificial Intelligence, Computer Vision, YOLOv8, EasyOCR, License Plate Recognition, Traffic Violation Detection

1. Introduction
Transportation is a cornerstone of modern urban life, underpinning commerce, public transportation, healthcare, and education. As metropolitan areas expand globally, managing vehicle flow has become an increasingly urgent challenge, particularly in developing countries.

Many cities are shifting toward Intelligent Transportation Systems (ITS) powered by artificial intelligence (AI) and computer vision to replace outdated manual tracking methods. Deep learning architectures, such as YOLOv8, provide the computational speed required for real-time detection, whereas Optical Character Recognition (OCR) enables the automated digital logging of license plates. However, this transition remains uneven. While developed regions have deployed advanced automated enforcement networks, many African nations continue to rely on a physical police presence because of financial constraints and infrastructure gaps. This paper addresses this divide by presenting an automated, cost-effective prototype tailored to developing urban contexts.

In Mogadishu, rapid urban expansion and a growing vehicle population have outpaced the capacities of existing manual traffic enforcement frameworks. Law enforcement still depends almost entirely on the visual observation of officers stationed at major intersections, a reliance exposed to human constraints such as fatigue, cognitive overload during peak hours, and limited visibility. The result is a high rate of unpenalized infractions, including wrong-direction driving and illegal parking, which compound congestion and elevate the risk of preventable accidents. Compounding the problem, the absence of a centralized digital record-keeping system prevents authorities from tracking repeat offenders or analyzing historical traffic patterns, leaving urban planners without the data needed to inform infrastructure decisions.

Despite an extensive body of literature on AI-based traffic networks, most existing systems have been engineered for structured environments with standardized road markings, stable power grids, and clearly formatted license plates. Comparatively little research has examined the technical application of computer-vision frameworks in the East African context, particularly in Somalia. Existing local literature is largely confined to general discussions of urban development rather than actionable technological frameworks. This study addresses this gap by developing an affordable, multi-layered prototype built on open-source tools to demonstrate the operational feasibility of automated traffic enforcement under local constraints, guided by the following questions: how can an AI-based system improve the efficiency of traffic violation detection and monitoring in Mogadishu; how can the YOLOv8 architecture support stable, real-time vehicle detection; how can OCR accurately extract license plate information; how can the system distinguish between specific infractions such as driving in the wrong direction and illegal parking; and what infrastructural challenges arise when deploying such a prototype locally?

Accordingly, the general objective of this study was to design, implement, and evaluate an AI-driven traffic-monitoring ecosystem as a functional proof-of-concept for automated surveillance in Somalia. Specifically, this study (i) designed a real-time vehicle-detection pipeline built on the YOLOv8 framework, (ii) automated the identification of wrong-direction driving and illegal parking, (iii) implemented an OCR pipeline for automated license-plate extraction, and (iv) constructed a web-based administrative dashboard for real-time reporting, automated fine management, and digital notification delivery. By replacing continuous manual observation with algorithmic oversight, the system is intended to reduce fatigue-related enforcement gaps, introduce objective evidence-based penalty logs, and offer municipal authorities a low-cost blueprint free of capital-intensive smart-city investment, which can also serve as a software-engineering reference for future intelligent infrastructure projects in Somalia.

The present study was deliberately scoped as a controlled proof-of-concept: it focused on vehicle detection, tracking, license plate extraction, and two infraction classes—wrong-direction driving and illegal parking—within a laboratory simulation using toy vehicles, scaled road models, and consumer webcams, developed over a one-month period.

2. Related Work

2.1 Intelligent Traffic Monitoring

Intelligent Transportation Systems (ITS) have emerged as the primary alternative to manual traffic enforcement globally [1]. Traditional computer vision applications for urban traffic analysis rely heavily on stationary cameras to process the geometric variations and temporal frameworks [2]. Early systems used background subtraction and optical flow techniques to track movement trajectories. However, these methodologies exhibit significant degradation under low-illumination boundaries and complex multi-vehicle intersections, emphasizing the modern shift toward deep-learning implementations.

2.2 AI & Computer Vision

The integration of Convolutional Neural Networks (CNNs) has revolutionized scene-understanding capabilities within traffic surveillance infrastructure. Modern AI architectures facilitate contextual encoding, enabling computer vision pipelines to segment dense traffic lanes and identify anomalous behaviors in dynamic environments [3]. These models reduce the computational overhead while maintaining high execution speeds across standard surveillance streams.

2.3 YOLO-Based Detection

Real-time object detection has been significantly advanced by the "You Only Look Once" (YOLO) framework, which models detection as a single regression problem by evaluating full video frames in a single pass [4]. The deployment of YOLOv8 introduces optimized anchor-free backbone structures and enhanced feature extraction mechanisms [5], [6]. This architecture achieves localized vehicle tracking, with detection metrics frequently exceeding 90% accuracy in favorable environments [7], providing the necessary operational speed for immediate citation logging.

2.4 OCR & License Plate Recognition

Automatic Number Plate Recognition (ANPR) systems convert raw bounding box coordinates into identifiable legal metadata using Optical Character Recognition (OCR) [8], [9]. Frameworks such as the Tesseract engine apply adaptive thresholding and character segmentation patterns to extract textual arrays from localized license plate fields [10]. Deep learning extensions optimize this extraction pipeline, mitigating typical visual constraints, including spatial distortion, variable camera angles, and structural format discrepancies across diverse geographical contexts [11], [12].

2.5 Traffic Violation Detection

Vision-based trajectory learning models evaluate temporal vehicle paths to automate infraction identification [13]. By calculating the localized bounding box directions against the vector baselines, the algorithmic logic establishes distinct parameters for identifying wrong-direction driving and restricted lane crossing. Furthermore, stationary bounding box tracking over designated spatial thresholds enables automated illegal-parking classification without human intervention [14].

2.6 Web-Based Systems

Relational database management stacks utilizing PHP and MySQL serve as industry standards for constructing secure web platforms for enforcement authorities. Centralized dashboards automate the relationship between digital computer vision evidence logs and fine administration modules [15]. Transitioning from physical paper records to automated database schemas can streamline processing workflows while reducing manual transcription discrepancies.

2.7 Developing Countries

Developing urban centers, particularly in Sub-Saharan Africa, face significant operational barriers to smart city transitions owing to limited Internet networks, inconsistent power resources, and severe capital constraints [16]. While developed municipalities rely on expensive embedded sensors, recent research advocates locally adapted prototypes that implement open-source deep-learning layers over consumer-grade hardware components [17]. No published empirical research currently addresses automated vision architectures configured for the unique road conditions and license infrastructure in Mogadishu, Somalia.

2.8 Research Gap & System Synthesis

The foundational studies reviewed established the performance benchmarks of YOLOv8 and OCR engines independently. However, a systemic gap remains in unified, open-source architectures tailored for resource-constrained urban environments that lack centralized DMV networks. This study bridges this gap by combining real-time detection, instant character mapping, relational fine tracking, and low-cost digital notification layers into a single prototype. To illustrate the technical advantages of this integrated architecture, **Table 1** contextualizes the proposed framework with respect to traditional traffic monitoring infrastructures.

Table 1: Comparison Between Proposed System and Existing Infrastructures

Feature	**Proposed System**	**CCTV Manual**	**ANPR Systems**	**Fixed Speed Cameras**
Real-Time Detection	Yes	No	Partial	Partial
AI-Powered (YOLOv8)	Yes	No	No	No
License Plate Recognition	Yes (OCR)	No	Yes	Limited
Automated Fine Generation	Yes	No	Partial	Partial
Web-Based Dashboard	Yes	No	Partial	No
Digital Notification	Yes	No	No	No
Developing Country Suitability	Yes	Yes	No	No
Cost-Effectiveness	High	Moderate	Low	Low

3. Proposed System and Methodology

3.1 Research Design

This project uses a design and development research approach to build and evaluate a smart traffic prototype. By blending qualitative workflow modeling with quantitative performance metrics (such as processing latency and accuracy scores), we created a practical blueprint for automated traffic systems. Instead of testing on busy public streets, we conducted our research using a controlled, scaled-down simulation setup to isolate and measure the variables accurately.

3.2 Development Approach

The system engineering process followed the classic Systems Development Life Cycle (SDLC) framework. We began by analyzing functional needs, moved through the database and software design, and ended with programming and rigorous modular testing. The prototype was explicitly engineered to monitor two common traffic infractions: wrong-way driving and illegal parking.

3.3 System Requirements

To ensure that the system remains practical and affordable for developing regions, the setup avoids expensive infrastructure. Instead, it pairs consumer-grade hardware with reliable open-source software libraries as shown in Tables 2 and 3.

Table 3.1: Hardware Setup Configurations

Component	**Main Operational Purpose**
Logitech C270 Webcam	Captures real-time video feeds from the simulated street lanes.
Desktop Workstation	Runs primary deep learning models and system logic.
Miniature Road Models	Represents the local physical street layouts and intersections.
Scaled Toy Vehicles	They serve as moving targets for testing vehicle tracking and variety.

Table 3: Software Framework Dependencies

Tool / Library	**Role in the System Architecture**
Python 3.10	It serves as the primary language for AI scripts and core backend logic.
YOLOv8	It performs real-time object detection and maps vehicle bounding boxes.
OpenCV 4.x	It handles video frame streaming and initial image transformations.
EasyOCR	Extracts and converts plate text from localized vehicle regions.
PHP & MySQL	Powers the administrative dashboard, fine logging and relational data.

3.4 System Architecture

The platform is organized into six modular layers that keep image processing, data management, and user notifications separate, yet fully cooperative. The vision layer streams live video data from the intersection webcams, feeding an AI processing layer that pinpoints vehicle coordinates, tracks movement, and reads license plates. These outputs are passed to a backend layer, which evaluates whether a given movement constitutes a violation and triggers automatic fines accordingly. Confirmed violations are then handed to a database layer that securely stores vehicle profiles, driver identities, and citation logs, and a notification layer

connects to external APIs to deliver digital alerts via email and WhatsApp. Finally, a dashboard layer renders a web interface through which traffic officers can review live statistics.

3.5 Vehicle Detection Pipeline

When a raw video feed is received, OpenCV cleans and scales each frame. The data were then fed immediately into the YOLOv8 object detection model. The network calculates the bounding box coordinates, assigns a vehicle class (such as a car or truck), and outputs a confidence score to verify that a vehicle is actually present.

3.6 OCR and License Plate Mapping

Once a vehicle is locked in, the software crops a tight Region of Interest (ROI) around the license plate area. This cropped image was converted to grayscale and sharpened using adaptive threshold filters to make the letters stand out. The cleaned image is then read by the OCR engine, which turns the visual plate into an active text string linked directly to that specific vehicle.

3.7 Violation Detection Logic

To reduce human error, the system relies on strict coordinate geometry to automatically flag traffic infractions. The tracking pipeline monitored the center point (centroid) of each vehicle's bounding box across sequential video frames. The centroid coordinates (Xc, Yc) at any given frame are calculated using the bounding box edges as follows:

$$Xc = (Xmin + Xmax) / 2$$

$$Yc = (Ymin + Ymax) / 2$$

3.7.1 Wrong-Direction Driving Logic

The system detects wrong-direction driving by calculating the vertical displacement vector (Delta Y) of a vehicle's centroid between the entry frame (t0) and current frame (tn):

$$Delta\ Y = Yc(tn) - Yc(t0)$$

Violation Rule: If Delta Y < 0 and |Delta Y| > distance_threshold, then Violation State = 1 (True). Otherwise, Violation State = 0 (False).

(Note: the distance_threshold is the minimum pixel distance required to rule out minor camera vibrations).

3.7.2 Illegal Parking Logic

Illegal parking is determined by monitoring the velocity (V) of a vehicle while its centroid position (Xc, Yc) falls inside a predefined restricted coordinate boundary polygon (P_zone). The frame-by-frame velocity in pixels was calculated using the distance formula:

$$Velocity = \sqrt{[(Xc(tn) - Xc(tn-1))^2 + (Yc(tn) - Yc(tn-1))^2]}$$

Violation Rule: If Velocity = 0 continuously for an observed duration (T) that exceeds the maximum permitted parking time limit (time_threshold), then Violation State = 1 (True). Otherwise, Violation State = 0 (False).

3.8 Evidence Capture and Relational Database

When an infraction is confirmed, the system saves a high-quality snapshot of the frame to serve as digital evidence. This image, along with the extracted plate text, violation type, timestamp, and AI confidence percentage, was written directly to the MySQL database. By utilizing clear relational foreign keys, the system links the new violation directly back to the car profile and the owner's registration file for complete record integrity. To illustrate the underlying structural

setup of this backend platform, Table 4 details the core relational database schema architecture and table columns utilized by the system during the runtime execution.

Table 4: Relational Database Schema Architecture

Table Name	Field Name	Data Type	Key Type	Core Operational Purpose
users	id	INT	Primary Key	Unique internal identifiers for system administrators.
	username	VARCHAR(50)	Unique	Dashboard authentication login user credential string.
	password_hash	VARCHAR(255)	—	Hashed password strings for authentication.
	role	VARCHAR(20)	Enumerated	Enforces role-based dashboard control privileges.
vehicles	id	INT	Primary Key	Internal reference index tracking registered logs.
	plate_number	VARCHAR(50)	Unique	Standard clean alphanumeric string output by EasyOCR.
	vehicle_type	VARCHAR(50)	Class Tag	Direct category tags generated by the YOLOv8 model.
violations	id	INT	Primary Key	Main tracking matrix index for confirmed road infractions.
	vehicle_id	INT	Foreign Key	Relational link connecting directly back to vehicles.
	violation_type	VARCHAR(100)	Rule Type	Flags Wrong-Direction or Illegal Parking violation states.
	fine_amount	DECIMAL(10,2)	Currency	Calculated statutory penalty amount logged inside the system.
	ai_confidence	DECIMAL(5,2)	Percentage	Bounding box localization and plate probability scores.

4. System Implementation

Writing the code for this system required the integration of two completely different environments: a high-speed Python AI pipeline and a user-friendly PHP/MySQL web application. We addressed this by creating a clean, step-by-step data workflow that was triggered when a camera detected a vehicle. The core AI engine runs on Python, using OpenCV to grab and scale video frames at 30 fps. We integrated the YOLOv8 model directly into the script. When YOLOv8 spots a vehicle with a confidence score above the set threshold, it tracks the vehicle across the screen using its center coordinates. If the tracking logic flags a movement as a violation, such as a toy car driving backwards into a lane, the script immediately crops the license plate area.

To read the plate, we passed the cropped image to EasyOCR. Because raw text from cameras can sometimes be slightly off, we added quick text-cleaning functions to remove accidental spaces or strange characters. Once we have a clean text string, the Python script sends all the violation details (the license plate string, violation type, and timestamp) directly to our MySQL database using a secure database connector.

On the front end, we built an administrative dashboard using PHP, HTML5, CSS3, and Bootstrap. We chose PHP because it smoothly handles real-time database updates. The moment a new row hits the MySQL database, the web dashboard refreshes to show the traffic officers the violation, automated fine amount, and a direct link to the captured photo evidence.

5. Experimental Setup and Evaluation

Because deploying this system on the real streets of Mogadishu was not possible at this early stage, we built a physical miniature test environment in our university laboratory to replicate real-world traffic conditions as closely as possible.

5.1 Physical Simulation Setup

Our lab setup consisted of a high-definition Logitech C270 webcam mounted on a fixed stand, looking down at a scaled road model complete with painted driving lanes, an intersection, and designated "no parking" zones. We used a collection of toy vehicles of various scales (including different shapes, colors, and sizes) to represent cars, trucks, and buses. Each toy vehicle was fitted with a printed miniature license plate to give the OCR engine a realistic test.

5.2 Evaluation Metrics

The system performance was assessed through repeated simulation runs using three evaluation metrics. Detection accuracy captured YOLOv8's ability to successfully identify every vehicle entering the camera's field of view, whereas the OCR recognition rate measured the accuracy of EasyOCR in reading the characters on the miniature license plates under varying lighting conditions. The third metric, system latency, was defined as the elapsed time in milliseconds between the occurrence of a violation and its appearance on the PHP web dashboard. Test runs were designed to deliberately trigger a range of violations, including restricted zone parking and wrong-direction lane driving, generating the performance data needed to determine whether an open-source AI configuration can reliably support traffic enforcement in a budget-constrained environment.

6. Results and Discussion

After running our miniature traffic simulations multiple times in the laboratory, we compiled the data to determine the performance of our prototype. The results were encouraging, indicating that an affordable, open-source AI setup can handle real-time traffic monitoring under controlled conditions.

6.1 Bounding Box and Detection Success

The YOLOv8 model demonstrated strong detection performance. Across all test runs with the toy vehicles, the model achieved a vehicle detection accuracy of 95.0%. It consistently locked onto vehicles regardless of their color or shape, maintaining a stable tracking bounding box even when multiple toy cars crossed the intersection simultaneously. The system was also highly effective in detecting infractions. The trajectory-tracking logic successfully flagged 92.0% of wrong-direction driving incidents and 90.0% of illegal parking violations inside the mapped-out camera zones. Minor misses usually occurred when a vehicle was partially blocked from the camera's view by another toy car.

6.2 License Plate Reading Efficacy

Reading miniature text on our printed plates using EasyOCR yielded a successful recognition rate of 88.5%. When the toy cars stopped completely, as in the illegal parking simulation, the OCR accuracy exceeded 93.0% because the vehicle was stationary. However, we noticed that the speed and angles affected the results. When toy cars moved quickly through the wrong-direction lane, slight motion blur occasionally caused the OCR engine to misread similar-looking characters (e.g., confusing the letter O with the number 0 or I with 1). In a real-world setup, this can easily be minimized by using a camera with a faster shutter speed or by adding a software script that cross-checks plates with a registry database.

6.3 System Latency and Dashboard Performance

A key design objective was to ensure that the system responded with minimal delay. We measured the latency—the time delay from the exact moment a toy car violated a rule to the moment the fine appeared on the web dashboard. On average, the total system latency was just 450 milliseconds. This means that the entire pipeline—capturing the video, detecting the car, cropping the plate, running the OCR, writing to the MySQL database, and updating the PHP dashboard— occurs in less than half a second. For a traffic authority officer sitting behind a screen, the alerts appear virtually in real time, making it a practical tool for live law enforcement.

7. Conclusion

Modernizing traffic management does not always require millions of dollars for proprietary infrastructure. This study demonstrated that by combining smart, open-source AI frameworks, such as YOLOv8 and EasyOCR, with simple web technologies, we can build a highly responsive and cost-effective traffic enforcement system. With a total latency of less than half a second, our prototype demonstrates that automated violation logging and digital fine generation can be achieved almost instantly, even in resource-constrained environments. By reducing the human error and fatigue that come with manual traffic policing, this system offers a clear path toward making the streets of Mogadishu safer and less congested. Ultimately, this project serves as a functional and technological stepping stone, indicating that AI-driven smart city solutions are achievable and practical within the Somali context.

7.1 Limitations

While the prototype performed well in testing, it is important to acknowledge its limitations. The biggest limitation of this study is that the system was built and evaluated entirely within a controlled laboratory simulation. Although toy cars and scaled road layouts helped us verify our code and AI logic, they cannot fully replicate the unpredictable nature of real-world traffic. In a real city environment, such as Mogadishu, our AI pipeline would have to deal with much harsher conditions. For instance, heavy dust, bright midday glare, and poor street lighting at night can reduce video quality, which would naturally lower our OCR plate recognition scores. Furthermore, our system was trained on a standardized dataset, which means that it might struggle to accurately recognize the wide variety of vehicle shapes, modified license plates, and unpaved road conditions found locally. Finally, because this was a one-month project, we relied on local server hosting via XAMPP rather than cloud integration, which means we could not test how the system performs when handling dozens of cameras simultaneously.

7.2 Future Work

Looking ahead, we want to take this prototype out of the lab and prepare it for practical city-wide deployment. The next logical step is to collect and build a Somali-specific vehicle and license plate dataset. Training YOLOv8 and EasyOCR on local vehicles and actual Mogadishu license plates will drastically improve detection accuracy and prevent character misreadings. We also plan to expand the system's brain so that it can detect more than just two types of infractions. Future updates will include algorithms for detecting speed limit violations, sudden lane cuts, and running red lights. On the engineering side, moving the backend architecture from a local machine to a cloud-based multi-camera infrastructure will allow the system to scale smoothly, allowing the police to centralize traffic data from multiple intersections across the city onto a single server. Finally, we look forward to conducting usability testing with actual traffic police officers to gather feedback and make the dashboard interface even easier for them to use during daily shifts.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] M. A. Hannan, A. M. Ali, A. Mohamed, and A. Hussain, "Intelligent bus monitoring and management system using computer-based technologies," *European Journal of Scientific Research*, vol. 98, no. 2, pp. 234-241, 2013.

[2] N. Buch, S. A. Velastin, and J. Orwell, "A Review of Computer Vision Techniques for the Analysis of Urban Traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920-939, Sep. 2011.

[3] H. Zhang, K. Dana, J. Shi, et al., "Context encoding for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7151-7160.

[4] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263-7271.

[5] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," Ultralytics, Tech. Rep., 2023.

[6] W. Ke, T. Zhang, Z. Huang, Q. Ye, Z. Jian, and X. Liu, "Multiple anchor learning for visual object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10206-10215.

[7] M. Rezaei and R. Klette, "Performance of different object detection algorithms for vehicle detection in traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2145-2156, Oct. 2014.

[8] I. Slimani et al., "An automated license plate detection and recognition system based on wavelet decomposition and CNN," *Array*, vol. 8, art. no. 100041, Dec. 2020.

[9] C.-N. E. Anagnostopoulos et al., "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377-391, Sep. 2008.

[10] R. Smith, "An Overview of the Tesseract OCR Engine," in *Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 629-633.

[11] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311-325, Feb. 2013.

[12] C. Henry et al., "Automated number plate recognition systems for traffic law enforcement applications," *Forensic Science International*, vol. 312, art. no. 110311, Jul. 2020.

[13] B. T. Morris and M. M. Trivedi, "A survey of vision-based trajectory learning and analysis for surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1114-1127, Aug. 2008.
[14] S. Ahmed et al., "AI-powered traffic violation detection systems in urban environments," *Journal of Intelligent Transportation Systems*, vol. 25, no. 4, pp. 112-126, 2021.
[15] R. Hassan et al., "Automated fine management systems and web-based traffic platforms," *International Journal of Information Management*, vol. 64, art. no. 102471, Jun. 2022.
[16] L. Oyedele et al., "Smart city technologies in Sub-Saharan Africa: A review," *Sustainable Cities and Society*, vol. 72, art. no. 103042, Sep. 2021.
[17] M. Khan et al., "Effectiveness of intelligent traffic management systems in developing countries," *Transportation Research Part A: Policy and Practice*, vol. 159, pp. 45-58, 2022.