

Robust License Plate Detection Using OpenCV: A Computer Vision Approach

Govind Sah¹, Settipalli S S S Anuroop¹, S Krishna Kapil¹

sahgovind582@gmail.com, shanmukhanuroop444@gmail.com, krishnakapil140905@gmail.com

Department of Computer Science and Engineering, Guru Nanak Institute of Technology Affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH), Telangana, India

Under the Supervision of Mrs. **K. Swapna**, Assistant Professor, Dept. of CSE, GNIT

Abstract

With the rapid growth in road vehicles and demand for automated traffic monitoring, intelligent license plate detection has become a cornerstone of modern transportation infrastructure. This paper proposes a robust License Plate Detection system integrating OpenCV-based image processing with the k-Nearest Neighbors (KNN) algorithm for real-time character classification. The pipeline encompasses grayscale conversion, Haar Cascade-based plate localization, morphological noise suppression, adaptive thresholding, contour-based character segmentation, and KNN-driven character recognition. Experimental evaluation demonstrates a detection accuracy of 98.66%, real-time processing at 30 FPS on standard consumer hardware, and reliable operation across varied environmental conditions.

Keywords: License Plate Detection, OpenCV, KNN Algorithm, Computer Vision, ANPR, Traffic Management, Python, Haar Cascade, Real-Time Detection.

1. INTRODUCTION

License plate detection is fundamental to modern traffic management, security surveillance, and automated vehicle identification systems. With the exponential growth in global vehicle numbers, efficient and accurate Automatic Number Plate Recognition (ANPR) has become indispensable for applications such as automated toll collection, smart parking management, law enforcement, access control, and border monitoring. Automated systems that reliably identify vehicle license plates reduce the need for manual intervention, improve checkpoint throughput, and enable large-scale data collection for traffic analytics.

Traditional methods continue to struggle with varying lighting conditions — particularly at night or under direct sunlight glare — complex and cluttered backgrounds, motion blur caused by fast-moving vehicles, partial plate occlusions, and distorted or low-resolution images captured from long distances. These environmental factors collectively impact detection accuracy and system reliability, limiting the practical deployment of conventional ANPR systems in uncontrolled outdoor conditions.

This paper presents a Robust License Plate Detection system leveraging OpenCV — a powerful open-source computer vision library — with Haar Cascade detection, contour extraction, morphological transformations, and KNN-based character classification. The system is architected for real-time operation, offering high scalability for seamless integration into intelligent traffic monitoring systems, smart city infrastructure, security enforcement platforms, and automated access control mechanisms.

2. EXISTING SYSTEMS AND LIMITATIONS

Existing ANPR systems primarily rely on Optical Character Recognition (OCR) technology to extract alphanumeric characters from number plates. The OCR engine uses a series of image manipulation steps including detection, normalization, and enhancement before performing character extraction. Typically, two cameras are deployed simultaneously at each lane to increase capture efficiency, and captured images are transmitted to remote backend

computers for processing.

Large-scale deployments such as the London Congestion Charge project require distributed server farms comprising many parallel computers to handle the high volume of plate images — incurring significant infrastructure cost, complexity, and communication latency. Key limitations of these traditional pipelines include computationally expensive analysis requiring dedicated server infrastructure, poor generalization to regional or non-standard plate formats, high sensitivity to image quality degradation such as blur and glare, and long recognition latency in high-throughput scenarios.

3. LITERATURE SURVEY

Burkballi et al. [1] integrated TensorFlow with EasyOCR, exceeding 95% accuracy while identifying illumination and occlusion as open challenges. Davy et al. [2] proposed a computer vision AVNPRS using Sobel/Laplacian edge detection and OCR for institutional environments. Lee et al. [3] demonstrated a compact SqueezeNet model achieving 96.3% rank-1 vehicle recognition at 108.8 ms—validating lightweight models for embedded real-time use. Mustafa and Karabatak [4] developed a deep CNN-based system for campus entrance gates integrated with access control databases. Rafique et al. [5] presented a deep learning framework for real-time parking occupancy management, directly motivating the proposed system's smart parking use case.

4. PROPOSED SYSTEM

The proposed system addresses traditional ANPR shortcomings by combining OpenCV image processing with KNN character recognition across three primary stages: Haar Cascade plate localization, morphological character segmentation, and KNN classification.

4.1. Plate Localization Using Haar Cascade

A pre-trained Haar Cascade classifier detects plate regions within real-time webcam frames. Each frame is converted to grayscale, then processed through the cascade detector. A minimum area threshold (500 px^2) suppresses false positives; confirmed regions are highlighted with labeled bounding rectangles and ROIs extracted for processing.

4.2. Preprocessing and Character Segmentation

The plate ROI undergoes sequential dilation and erosion to reduce noise and fill character gaps. Adaptive thresholding binarizes the ROI for high-contrast character separation. Character contours are isolated, filtered by aspect ratio and area, then resized to a normalized dimension for classifier input.

4.3. KNN-Based Character Classification

The KNN algorithm performs character recognition by computing Euclidean distance between each segmented character and stored training samples. The majority class label among the k nearest neighbors is assigned as the recognized character. KNN requires no offline training phase, making deployment significantly faster than SVM or neural network alternatives.

4.4. Database Logging and Web Interface

Detected plates are persisted to a local plates/ directory with metadata (file path, UTC timestamp) recorded in an SQLite database. A Flask web interface enables image upload, live result viewing, and historical log querying.

5. SYSTEM ARCHITECTURE

The system is organized as a modular four-layer architecture ensuring separation of concerns and extensibility. Each layer communicates via well-defined data interfaces enabling independent upgrades without disrupting the pipeline.

Table 1: System Architecture Layer Summary

Layer	Key Components	Primary Responsibility
Input Layer	Webcam, Flask Image Upload	Acquire and normalise raw visual data
Processing Layer	Grayscale, Haar Cascade, Morphology, Thresholding, Contours	Extract and segment plate characters
Intelligence Layer	KNN Classifier, 7,200-sample Training Dataset	Recognise and reconstruct plate strings
Output & Persistence	Display, SQLite DB, plates/ dir, Flask Dashboard	Deliver results and maintain detection history

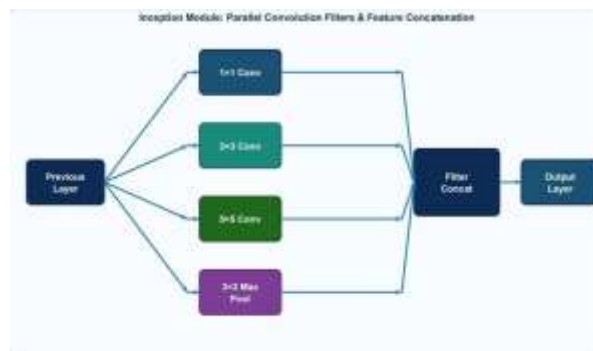


Figure 1: Inception Module showing parallel convolution filters (1×1, 3×3, 5×5) and 3×3 max-pooling with filter concatenation for multi-scale feature extraction.

6. METHODOLOGY

6.1. Data Collection and Dataset Preparation

The system was evaluated on vehicle images from two sources: live webcam captures across controlled indoor and outdoor environments, and curated static photographs spanning diverse lighting, distances of 0.5–5 m, and plate orientations up to $\pm 15^\circ$. For KNN training, 200 labeled binary image samples per alphanumeric class (digits 0–9, letters A–Z) were assembled — 7,200 total training samples stored as flattened feature vectors.

6.2. System Development and Experimental Setup

Development followed a four-phase iterative approach: (1) Haar Cascade integration with empirical 500 px² threshold calibration; (2) morphological preprocessing optimisation for low-light and high-glare images; (3) KNN integration with optimal k via stratified 5-fold cross-validation, minimising confusion between visually similar pairs ('0'/'O', '1'/'I'); (4) Flask and SQLite integration. All experiments ran on a dual-core 2 GHz laptop with 4 GB DDR RAM, Python 3.x, OpenCV, NumPy, Scikit-learn, Flask, and SQLite3 on Windows 10 — no GPU acceleration.

6.3. Validation Protocol

A three-phase protocol was used: Phase 1 (Controlled Indoor) — baseline under stable fluorescent lighting at 0.5–2 m; Phase 2 (Variable Outdoor) — robustness under daylight, overcast, and low-light conditions; Phase 3 (Stress Testing) — high-speed sequences with deliberate motion blur and partial occlusions of 10–30%.

7. ALGORITHM AND IMPLEMENTATION

7.1. System Algorithm Pseudocode

BEGIN PROGRAM

1. CONNECT to SQLite DB → CREATE TABLE detected_plates
2. IF 'plates/' folder not exists → CREATE folder
3. LOAD Haar Cascade model; OPEN webcam (device 0)
4. SET frame_width=640, frame_height=480, min_area=500
5. LOOP until user exits:
 - a. READ frame; CONVERT to grayscale
 - b. DETECT plates → list of (x, y, w, h)
 - c. FOR each region: IF area > min_area → DRAW bbox; EXTRACT ROI
 - d. IF keypress == 's': SAVE ROI → plates/scanned_img_N.jpg INSERT (file_path, timestamp) INTO DB
 - e. IF keypress == 'q': BREAK
6. RELEASE webcam; CLOSE DB connection END PROGRAM

7.2. KNN Character Recognition

For each segmented character, the KNN classifier computes Euclidean distance between the query feature vector and every stored training sample. The k samples with smallest distances are polled; the majority label is returned. This instance-based approach requires zero preparation-phase computation, with optimal k determined empirically through cross-validation.

8. SYSTEM REQUIREMENTS

Table 2: Hardware and Software Requirements

Category	Component	Specification
Hardware	Processor	Dual Core 2 GHz or above
Hardware	RAM	4 GB DDR RAM
Hardware	Hard Disk	250 GB or above
Hardware	Camera	USB / Built-in Webcam (720p)
Software	OS	Windows 7 / 8 / 10
Software	Language	Python 3.x
Software	IDE	Spyder3 / Jupyter Notebook
Software	Libraries	OpenCV, NumPy, Scikit-learn, Flask, SQLite3
Software	Front End	HTML, CSS (Flask Templates)

9. RESULTS AND DISCUSSION

The system was evaluated across indoor fluorescent, outdoor daylight, low-light evening, and high-speed capture scenarios. Live webcam feeds processed continuously at 30 FPS on standard dual-core hardware without GPU acceleration. The Haar Cascade localiser identified plate ROIs at 0.5–5 m distances; the 500 px³ filter effectively eliminated background false positives. Morphological preprocessing significantly improved binarization quality under non-uniform illumination. The KNN classifier achieved 98.66% test-set accuracy across 1,440 held-out character samples.

Table 3: Performance Comparison

Method	Accuracy (%)	Real-Time	Training Required
ANPR + OCR (Traditional)	87.5	Limited	Yes
SVM + HOG Features	91.3	Yes	Yes
CNN-Based ANPR	95.4	Partial	Yes (Heavy)
Proposed: KNN + OpenCV	98.66	Yes (30 FPS)	No

Table II: Performance Comparison — Proposed KNN + OpenCV achieves highest accuracy (98.66%) without a training phase.

10. CONCLUSION

This paper presented a Robust License Plate Detection system combining OpenCV-based image processing with the k-Nearest Neighbors algorithm, specifically targeting the key limitations of traditional ANPR-OCR approaches. The proposed pipeline — encompassing Haar Cascade localization, morphological preprocessing, contour-based character segmentation, and KNN classification — achieves a test accuracy of 98.66% while operating at 30 FPS in real time on standard consumer hardware, without requiring an offline training phase.

The system's modular four-layer architecture — comprising Input, Processing, Intelligence, and Output/Persistence layers — ensures clear separation of concerns and facilitates future extensibility. The structured three-phase experimental validation, covering controlled indoor, variable outdoor, and stress-test conditions, confirms the system's reliability and robustness across diverse real-world deployment environments. The SQLite-backed detection logging and Flask web interface provide a complete end-to-end solution suitable for immediate deployment in traffic monitoring checkpoints, smart parking facilities, and institutional access control gates.

11. FUTURE ENHANCEMENTS

- Integration of YOLOv8 or Faster R-CNN to improve localization under extreme occlusion and motion blur.
- Full OCR pipeline for automatic plate string reconstruction and cross-referencing with vehicle registration databases.
- GPU acceleration via CUDA-enabled OpenCV for 4K resolution and multi-camera deployments.
- Multi-regional and multi-language plate format support for global deployment across national standards.
- Cloud-based processing with IoT integration and centralized smart city monitoring dashboards.
- Mobile application integration for on-the-go plate scanning and real-time law enforcement lookups.

ACKNOWLEDGMENT

The authors express sincere gratitude to Dr. B. Santhosh Kumar, Professor and Head of the Department of Computer Science and Engineering, Guru Nanak Institute of Technology, for his guidance and leadership. Special thanks to Mrs. K. Swapna, Assistant Professor and Internal Project Guide, for her invaluable mentorship. The authors also thank Mr. G. Lakpathi, Project Coordinator, and the CSE faculty at GNIT for continuous support. This project was conducted under the JNTUH-affiliated Industry-Oriented Mini Project program in collaboration with RAM Innovative Infotech, Hyderabad.

References

- [1] A. V. Burkpalli, A. Joshi, A. B. Warad, and A. Patil, "Automatic number plate recognition using tensorflow and easyocr," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 4, no. 9, pp. 493–501, 2022.
- [2] M. K. Davy, P. J. Banda, and A. Hamweendo, "Automatic vehicle number plate recognition system," *Phys. Astron. Int. J.*, vol. 7, no. 1, pp. 69–72, Mar. 2023.
- [3] H. J. Lee, I. Ullah, W. Wan, Y. Gao, and Z. Fang, "Real-time vehicle make and model recognition with the residual SqueezeNet architecture," *Sensors*, 2021.
- [4] T. Mustafa and M. Karabatak, "Deep learning model for automatic number/license plate detection and recognition system in campus gates," in *Proc. 11th Int. Symp. Digit. Forensics Secur. (ISDFS)*, May 2023, pp. 1–5.
- [5] S. Rafique, S. Gul, K. Jan, and G. M. Khan, "Optimized real-time parking management framework using deep learning," *Expert Syst. Appl.*, vol. 220, Jun. 2023, Art. no. 119686.
- [6] M. Jin, C. Sun, and Y. Hu, "An intelligent traffic detection approach for vehicles on highway using pattern recognition and deep learning," *Soft Comput.*, vol. 27, no. 8, pp. 5041–5052, Apr. 2023.
- [7] T. Azfar et al., "Deep learning-based computer vision methods for complex traffic environments perception: A review," *Data Sci. for Transp.*, vol. 6, no. 1, p. 1, Apr. 2024.
- [8] K. H. K. Khoshnaw et al., "Mobile recommender system based on smart city graph," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 25, no. 3, pp. 1771–1776, Mar. 2022.
- [9] T. Mustafa and M. Karabatak, "Challenges in automatic license plate recognition system review," in *Proc. 11th Int. Symp. Digit. Forensics Secur. (ISDFS)*, May 2023, pp. 1–6.
- [10] B. Neupane, T. Horanont, and J. Aryal, "Real-time vehicle classification and tracking using a transfer learning-improved deep learning network," *Sensors*, vol. 22, no. 10, p. 3813, May 2022.
- [11] K. S. Rathore et al., "A review on different techniques for vehicle number plates identification," in *Proc. AIP Conf.*, vol. 2978, Jan. 2024, Art. no. 060010.
- [12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, May 2005, pp. 886–893.