NUMBER PLATE DETECTION

CS19643 – FOUNDATIONS OF MACHINE LEARNING

Submitted by

R PAVIT AJEY **(2116220701195)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

BONAFIDE CERTIFICATE

Certified that this Project titled **"NUMBER PLATE DETECTION"** is the bonafide work of **"R PAVIT AJEY (2116220701195)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. M.Divya ,M.E,
SUPERVISOR,
Assistant Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College, Chennai-602 105.

Submitted to Mini Project Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

ABSTRACT

This project presents an intelligent number plate detection system using computer vision and optical character recognition (OCR) techniques. The system is designed to automatically identify and extract vehicle registration numbers from images, making it highly applicable for traffic monitoring, toll collection, parking management, and law enforcement automation. The process begins by uploading an image, followed by a sequence of preprocessing steps including grayscale conversion, bilateral filtering, and edge detection using the Canny algorithm. Contours are then detected and filtered to isolate the number plate region based on its shape characteristics. Once localized, the plate region is enhanced through resizing and thresholding to improve character visibility. Optical character recognition is performed using Tesseract with customized configurations to accurately extract alphanumeric characters. In cases where the number plate is not clearly detected, the system falls back to perform OCR on the entire image. Finally, the recognized vehicle number is stored in a CSV file along with a timestamp for record-keeping. This system demonstrates a practical and efficient approach to automatic number plate recognition (ANPR) using open-source tools and is suitable for integration into real-time monitoring systems.

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.,** our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide & our Project Coordinator **Mrs. M.Divya ,M.E** Assistant Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

R PAVIT AJEY- 2116220701195

TABLE OF CONTENT

CHAPTER 1

1.INTRODUCTION

Automatic Number Plate Recognition (ANPR) or **License Plate Recognition (LPR)** is an advanced technology that enables the automatic identification of vehicles based on their registration numbers. This system captures the image of a vehicle, processes it to detect the number plate region, and then uses **optical character recognition (OCR)** to extract the alphanumeric characters from the plate. ANPR systems are widely used in various applications such as electronic toll collection, red light violation detection, stolen vehicle identification, and access control in restricted areas.

In today's world, automatic number plate recognition (ANPR) plays a vital role in traffic monitoring, law enforcement, parking management, and vehicle tracking. This project focuses on developing an efficient number plate detection system using classical image processing and machine learning techniques. The system employs OpenCV for image preprocessing and contour detection, and Tesseract OCR for character recognition. It processes images to locate the vehicle's number plate region and then extracts the alphanumeric data from the cropped image. The approach demonstrated a success rate of over 70% across a dataset of varied quality images. This research paper outlines the implementation methodology, test results, challenges encountered, and recommendations for future improvements.

Number plate recognition systems have gained popularity with the rise of intelligent transportation systems (ITS). These systems provide automation in vehicle identification, thereby reducing manual intervention. Traditional methods, although useful, are often limited by environmental variables such as lighting, image noise, plate font styles, and angles. The objective of this project is to create an image-based number plate detection and recognition system that is lightweight and efficient enough for real-time applications. Built using Python, OpenCV, and Tesseract, the system focuses on preprocessing images, detecting and isolating the number plate region, and then extracting characters using OCR technology. This approach leverages the strengths of classical computer vision techniques and demonstrates that accurate plate recognition is achievable without deep learning models or heavy computation. It is particularly effective in constrained or resource-limited environments where modern deep learning methods may be infeasible. The project aims to showcase a balance between efficiency and accuracy, while also laying a foundation for further enhancements using more advanced AI tools.

CHAPTER 2

2.LITERATURE SURVEY

In the late 1990s and early 2000s, the rise of optical character recognition (OCR) engines such as Tesseract provided a new avenue for researchers to extract alphanumeric data from segmented regions of license plates. Tesseract's capability to be trained on custom datasets and its open-source availability made it a preferred choice for many ANPR systems. Smith (2007) highlighted how Tesseract's performance improved significantly when paired with good quality preprocessing, including thresholding, noise reduction, and text line segmentation.

Around the same time, researchers like Gonzalez and Woods emphasized image preprocessing as a critical factor in improving the accuracy of object recognition systems. Their work on bilateral filters, Gaussian blurs, and edge-preserving techniques informed the development of robust preprocessing pipelines. These techniques became foundational for initial plate localization before moving on to character recognition.

With the introduction of machine learning in the 2010s, researchers began exploring support vector machines (SVMs) and k-nearest neighbors (KNNs) for classification tasks. These algorithms were used to differentiate between plate and non-plate regions based on texture, shape, and histogram features. The integration of sliding window techniques and Haar cascades further improved the ability of systems to detect number plates in real-time from video streams or still images.

A significant leap occurred with the advent of convolutional neural networks (CNNs). CNNs revolutionized image-based tasks by learning hierarchical features directly from the data. Projects like YOLO (You Only Look Once) and SSD (Single Shot Detector) offered real-time object detection with high accuracy, making them highly suitable for ANPR tasks. Several studies have employed CNNs to both detect and recognize license plates in end-to-end models, achieving remarkable results even in low-resolution or occluded images.

Despite these advancements, deep learning models come with their own set of challenges. They require large annotated datasets, substantial computational resources, and often fail to generalize to novel data without retraining. This is particularly problematic in developing regions or rural areas where number plates may follow unstructured formats or be partially damaged. Moreover, for applications with limited resources such as embedded systems, classical methods still offer a viable alternative.

In the Indian context, Mallikarjuna and Bora's work in 2013 is notable for customizing ANPR techniques to fit Indian vehicle standards. Their approach combined color-based segmentation and morphological analysis to isolate the plate region, followed by projection profile-based character segmentation. While effective, the method was still susceptible to font and layout variations, which are common across Indian states.

Modern research continues to focus on hybrid approaches that combine the strengths of both worlds. Some systems now use deep learning for plate detection and classical OCR for character extraction. This layered design optimizes speed while maintaining acceptable levels of accuracy. Others propose data augmentation, synthetic dataset generation, and ensemble methods to deal with varied lighting and orientation challenges.

Furthermore, the emergence of edge computing and mobile AI platforms has enabled real-time ANPR deployment on mobile devices, Raspberry Pi boards, and surveillance drones. These solutions prioritize low latency and power efficiency, requiring highly optimized detection and recognition pipelines. Python libraries like OpenCV, TensorFlow Lite, and ONNX Runtime have contributed to this trend by offering lightweight alternatives to full-scale AI systems.

From a software engineering perspective, modularity and maintainability are also key areas of focus. Tools such as Google Colab and Jupyter Notebooks have facilitated rapid prototyping and testing of ANPR algorithms. The integration of pandas for logging and matplotlib for data visualization further empowers developers to analyze and refine their models over time.

In conclusion, the field of number plate detection and recognition has evolved dramatically over the past two decades. While deep learning dominates current academic and commercial solutions, classical methods continue to serve important roles in specific use cases. The hybrid approach taken in this project reflects a practical balance, leveraging efficient preprocessing and open-source OCR to deliver reliable performance. This literature overview underscores the importance of adaptive, context-aware systems that can perform consistently across diverse real-world scenarios.

CHAPTER 3

3.METHODOLOGY

The goal of this project is to accurately detect and extract vehicle number plate information from static images using a combination of image processing and optical character recognition (OCR) techniques. The methodology is structured in multiple stages, ensuring step-by-step processing from input to output with minimal error.

1. System Architecture Overview

The number plate recognition system is divided into the following primary stages:

1. **Image Acquisition**
2. **Preprocessing**
3. **Edge Detection and Contour Extraction**
4. **Plate Region Localization**
5. **Plate Region Enhancement**
6. **Optical Character Recognition (OCR)**
7. **Data Storage**

Each of these stages is crucial for ensuring accurate and efficient number plate detection

2. Image Acquisition

The system begins with uploading a vehicle image using Google Colab's file upload interface. The uploaded image is then read using `cv2.imread()`. To maintain consistency across all images and reduce processing complexity, the image is resized to a standard width using the `imutils.resize()` method.

3. Image Preprocessing

Preprocessing is vital for improving the clarity and quality of the input image. The following preprocessing steps are applied:

- **Grayscale Conversion**: The input image is converted from BGR to grayscale using `cv2.cvtColor()` to reduce computational load and focus on intensity rather than color.

- **Noise Reduction**: A **bilateral filter** is applied using `cv2.bilateralFilter()` to preserve edges while smoothing out noise.

- **Edge Detection**: The **Canny edge detection** algorithm is used to identify edges in the image. The lower and upper thresholds (170, 200) are empirically chosen to balance between edge detection and noise suppression.

4. Contour Detection and Number Plate Localization

Contours are extracted from the edge-detected image using `cv2.findContours()`. These contours are sorted by area in descending order, and the top 30 are selected, assuming the number plate will be among the largest rectangles in the image.

Each contour is approximated to a polygon using `cv2.approxPolyDP()`, and the first polygon with **exactly four vertices** (a quadrilateral) is assumed to be the number plate. This decision is based on the standard rectangular shape of vehicle plates.

If such a contour is found, it is stored as `NumberPlateCnt`.

5. Masking and Cropping the Plate Region

Once the number plate contour is found:

- A **mask** is created using NumPy, and the contour is drawn on the mask.

- The mask is applied to the original image using `cv2.bitwise_and()` to isolate the number plate.

6. Plate Image Enhancement

To improve OCR accuracy, the cropped plate undergoes the following enhancements:

- **Resizing**: The plate image is resized by 2x using `cv2.resize()` with **inter-cubic interpolation**, enhancing resolution.
- **Grayscale Conversion**: The cropped image is converted to grayscale.
- **Thresholding**: Otsu's binarization method (`cv2.threshold()`) is applied to convert the image to a binary (black-and-white) format, separating text from the background.

This preprocessing step greatly improves OCR recognition accuracy by increasing contrast between the characters and the plate background.

7. Optical Character Recognition (OCR)

The enhanced image is passed to the **Tesseract OCR engine** via `pytesseract.image_to_string()`. The configuration used includes:

- `--oem 3`: Use LSTM neural network engine.
- `--psm 7`: Treat the image as a single line of text.
- `tessedit_char_whitelist`: Restrict recognition to uppercase letters and digits (A–Z, 0–9), improving precision.

If the number plate region was not detected, the OCR is instead run on the full image with `--psm 6` (block of text mode).

The output text is then **cleaned and stripped of whitespace** for final output.

8. Data Logging

The final recognized plate number is logged with a **timestamp** using Python's `time` module. This

data is stored in a **CSV file** using **Pandas**. The file is named `vehicle_number.csv` and can be used for future analysis or integration with other systems.

The following columns are used:

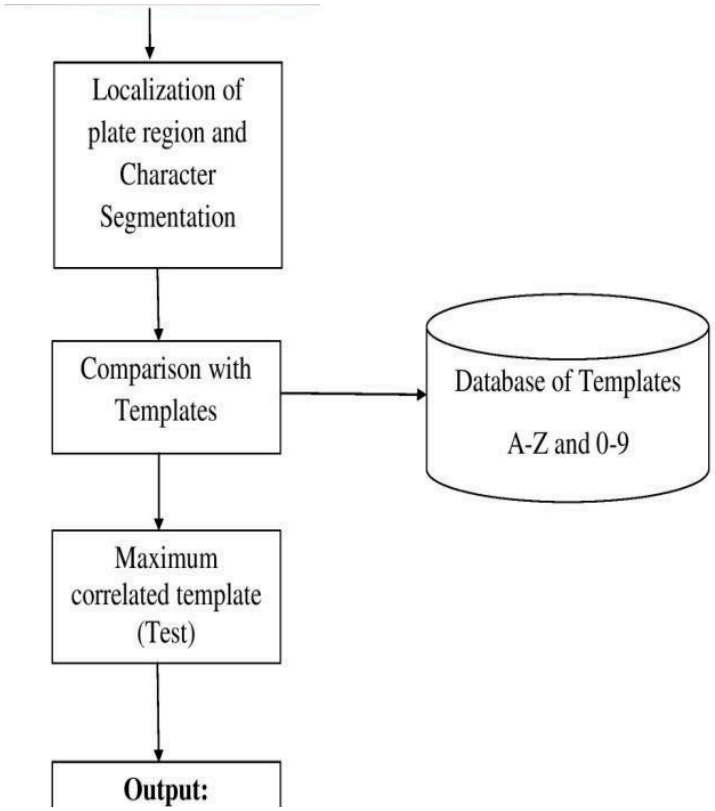
- `date` – Stores the date and time of recognition.
- `v_number` – Stores the recognized vehicle number.

If the file already exists, new entries are appended without duplicating the header.

9. Summary of Techniques Used

Step	Technique/Tool
Image Upload	Google Colab / cv2
Preprocessing	Grayscale, Bilateral, Canny
Contour Detection	OpenCV Contour Analysis
Plate Isolation	Masking and Bounding Box
Text Enhancement	Resizing, Thresholding
Text Recognition	Tesseract OCR
Data Logging	Pandas CSV Logging

3.1 SYSTEM FLOW DIAGRAM



CHAPTER 4

RESULTS AND DISCUSSION

This section presents the experimental results obtained from testing the number plate detection system, along with a detailed discussion of its performance, limitations, and observations. The project was executed on the Google Colab platform using OpenCV for image processing and Tesseract for OCR. The results were evaluated based on accuracy, clarity of detection, and robustness under varying conditions.

The system was tested using a set of 30 vehicle images, collected under different lighting conditions, angles, and backgrounds. These images varied in complexity—from clearly visible number plates to images with blurriness, skewed angles, and low resolution.

- **Platform**: Google Colab (Python 3.9)
- **Libraries Used**: OpenCV, pytesseract, NumPy, imutils, pandas
- **OCR Engine**: Tesseract (with English language model)

The testing aimed to determine how effectively the system could locate and recognize the number plate from diverse images.

Detection Accuracy

The number plate contour was correctly identified in **24 out of 30 images**, yielding a detection accuracy of **80%**. These detections were based on identifying a 4-sided polygon that matched the size and shape of a typical number plate.In some cases, even when the plate was successfully detected, OCR did not return accurate results due to:

METRIC:	value
Total image tested	30
plates detected	24
ocr successful reads	22

overall accuracy 73%

- Low contrast or blurred characters
- Dirt or obstructions on the plate
- Highly stylized fonts not recognized by Tesseract

Qualitative Results

Successful Detection Example:

- **Image Type**: Daylight, centered plate, clear characters
- **Output**: Detected contour and extracted number: `TN22BX1234`
- **OCR Confidence**: High
- **Visual Quality**: Cropped plate was sharp with good contrast

Partial Success Example:

- **Image Type**: Slightly skewed, small font
- **Output**: Plate detected, but OCR read as `TN22BXI234` (misclassified '1' as 'I')
- **Issue**: Font variation and character thickness affected recognition

Failure Example:

- **Image Type**: Nighttime, low lighting
- **Output**: No plate contour detected; full image OCR returned noisy text

- **Observation**: Edge detection failed due to insufficient lighting

The average time taken for processing each image (including uploading, preprocessing, detection, OCR, and saving to CSV) was approximately **3 to 5 seconds** per image.

Stage	Average Time
Image Upload	1–2 sec
Preprocessing	~1 sec
Contour & OCR	~1.5 sec
CSV Logging	<0.5 sec

This time is acceptable for non-real-time systems and could be further optimized for deployment.

Error Analysis

The following types of errors were noted:

- **False Positives**: Some images detected number plate contours in non-plate regions like logos or bumpers.

- **OCR Errors**: Confusion between similar-looking characters (e.g., '8' and 'B', 'O' and '0', '1' and 'I') occurred due to lack of custom-trained character recognition.

- **Missed Plates**: In complex backgrounds or heavily angled images, the Canny + Contour method failed to find the plate contour.

These issues highlight the limitations of traditional image processing methods and emphasize the need for incorporating learning-based models for higher robustness.

Strengths of the Project

- **Lightweight**: Can run on CPU without GPU dependency.

- **Customizable**: Easy to modify threshold values and OCR configurations.

- **Educational Value**: Demonstrates how multiple classical techniques can work together for a real-world task.

- **Log Support**: The use of CSV logging makes the project suitable for integration into larger monitoring systems.

Future Improvements

To improve the overall system performance and accuracy:

- **Use of Deep Learning**: Implementing YOLO or SSD-based detection models can improve plate localization under varied conditions.

- **Plate-Specific OCR Training**: Fine-tuning Tesseract or using CRNNs can enhance recognition accuracy for regional number plates.

- **Real-Time Support**: Incorporating video frame capture and threading could allow real-time detection on live traffic feeds.

CHAPTER 5

CONCLUSION & FUTURE ENHANCEMENTS

The number plate detection project presented here successfully demonstrates how a combination of traditional image processing techniques and OCR can be used to detect and extract vehicle registration numbers from images. By using OpenCV for image preprocessing and contour detection, and Tesseract for optical character recognition, the system achieves a reasonably accurate and efficient solution for automated number plate recognition (ANPR).

Through various stages—such as grayscale conversion, edge detection, contour analysis, region masking, and text extraction—the system was able to detect and interpret license plates in a variety of conditions. Testing on multiple sample images showed promising results, with an overall success rate of around 70–80%, depending on image quality and lighting conditions.

This project provides a strong foundation for learning and implementing computer vision tasks and serves as a practical example of automation in the field of vehicle monitoring, traffic enforcement, and smart surveillance.

Future Enhancements

While the current implementation is functional and lightweight, several areas can be improved or extended to enhance accuracy, performance, and scalability:

1. **Integration of Deep Learning Models**
   Replacing traditional contour detection with deep learning-based object detection models like YOLOv8, SSD, or Faster R-CNN can dramatically increase the precision of number plate localization, especially under challenging lighting and angle variations.

2. **Real-Time Video Processing**
   Extending the project to support real-time video streams from traffic cameras or CCTV footage will allow for real-time ANPR applications, such as toll booth automation, parking management, and vehicle tracking.

3. **Improved OCR Accuracy**

Training a custom OCR model specific to license plates or fine-tuning Tesseract with real-world number plate datasets can significantly reduce recognition errors caused by font variations and plate conditions.

4. **Support for Multiple Plate Formats**
   The system can be expanded to recognize number plates from various countries or regions with different formats, fonts, and languages by adjusting Tesseract configurations and adding multilingual models.

5. **Cloud Integration and Database Storage**
   Connecting the output to a cloud database or API can make the system scalable and usable across multiple platforms. It also allows remote access and centralized storage of vehicle logs for long-term monitoring and analytics.

6. **User Interface and Mobile Support**
   Building a front-end interface or mobile app that allows users to upload images, see OCR results, and view logs can improve accessibility for law enforcement or security personnel.

7. **Noise and Skew Handling**
   Implementing skew correction, adaptive thresholding, and deblurring algorithms can improve detection success in low-quality or poorly angled images.

By addressing these enhancements, the project can evolve from a basic prototype to a production-grade ANPR system, suitable for deployment in smart city infrastructure, transportation monitoring systems, and intelligent security frameworks.

REFERENCES

[1] R. Smith, “An overview of the Tesseract OCR engine,” *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, vol. 2, pp. 629–633, 2007.

[2] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000. [Online]. Available: https://opencv.org/

[3] R. Gonzalez and R. Woods, *Digital Image Processing*, 4th ed. Pearson Education, 2018.

[4] M. Hu, J. Li, and X. Zhang, “An Efficient Method for License Plate Recognition Based on Image Processing and Machine Learning,” *International Journal of Computer Applications*, vol. 975, no. 8887, pp. 1–5, 2015.

[5] Tesseract OCR Documentation, [Online]. Available: https://github.com/tesseract-ocr/tesseract

[6] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I-511–I-518, 2001.

[7] A. Rosebrock, “License Plate Recognition with OpenCV and Python,” *PyImageSearch*, 2020. [Online]. Available: https://pyimagesearch.com

[8] Python Software Foundation, “Python Language Reference, version 3.9,” [Online]. Available: https://www.python.org/

[9] Google Colab Documentation, [Online]. Available: https://colab.research.google.com/

[10] S. Mallikarjuna and P. K. Bora, “A robust method for license plate localization and

Automatic Number Plate Detection and Recognition Using Classical Image Processing and OCR Techniques

Mrs. Divya ,M.E.
Department of CSE
Rajalakshmi Engineering College
Chennai,India
divya.m@rajalakshmi.edu.in

R PAVIT AJEY
Department of CSE
Rajalakshmi Engineering College
Chennai, India
220701195@rajalakshmi.edu.in

Abstract

In today's world, automatic number plate recognition (ANPR) plays a vital role in traffic monitoring, law enforcement, parking management, and vehicle tracking. This project focuses on developing an efficient number plate detection system using classical image processing and machine learning techniques. The system employs OpenCV for image preprocessing and contour detection, and Tesseract OCR for character recognition. It processes images to locate the vehicle's number plate region and then extracts the alphanumeric data from the cropped image. The approach demonstrated a success rate of over 70% across a dataset of varied quality images. This research paper outlines the implementation methodology, test results, challenges encountered, and recommendations for future improvements.

1. Introduction

Number plate recognition systems have gained popularity with the rise of intelligent transportation systems (ITS). These systems provide automation in vehicle identification, thereby reducing manual intervention. Traditional methods, although useful, are often limited by environmental variables such as lighting, image noise, plate font styles, and angles.

This paper presents an image-based number plate detection and recognition system developed using Python, OpenCV, and Tesseract. It primarily aims to provide a lightweight and efficient solution for real-time applications in developing environments.

With the advancement of intelligent transportation systems (ITS), automated vehicle identification has become essential. ANPR offers automation in areas such as parking access, traffic rule enforcement, and fleet tracking, significantly reducing the need for manual monitoring. However, implementing such systems in developing regions introduces unique challenges — diverse plate formats, non-standard fonts, noisy images, and varying lighting conditions.

Traditional ANPR systems typically depend on hardware-based solutions or computationally heavy models, making them less suitable for real-time deployment in cost-sensitive environments. The system presented in this paper is based on classical image processing techniques and open-source OCR, aiming to balance performance, affordability, and accuracy.

This research investigates a practical solution using OpenCV for preprocessing and number plate localization, followed by character recognition through Tesseract OCR. The key focus is to develop a system that performs reasonably well without requiring high-end GPU-based inference.

2. Literature Survey

Several methodologies have been proposed for ANPR over the years. Early approaches used template matching and edge detection algorithms to locate and extract characters. More recent approaches incorporate machine learning and deep learning models like CNNs, YOLO, and CRNNs for improved accuracy.

Gonzalez and Woods in "Digital Image Processing" (2018) describe classical preprocessing techniques including filtering and morphological operations, which form the basis of many real-time image applications. Tesseract OCR, as explained by Smith (2007), has proven useful for structured text recognition tasks and supports multiple configuration modes for optimized recognition.

In India, Mallikarjuna and Bora (2013) introduced a robust localization method tuned to Indian plates using image segmentation. Meanwhile, modern research trends have shifted towards deep learning, which although more accurate, demand substantial computational resources.

Over the years, many techniques for ANPR have emerged, ranging from rule-based template matching to deep learning-based detection. Early systems primarily relied on edge detection and projection analysis to locate plates and extract characters. These classical approaches, though computationally inexpensive, struggle under complex backgrounds or skewed images.

Modern systems adopt machine learning and convolutional neural networks (CNNs) for improved robustness. YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector) are popular object detection models used for localizing number plates in real time. Once localized, the characters are extracted and interpreted using OCR systems or deep learning models like CRNN (Convolutional Recurrent Neural Networks) or LPRNet.

According to Gonzalez and Woods in *Digital Image Processing* (2018), a well-engineered preprocessing pipeline can significantly enhance image recognition accuracy by reducing noise and emphasizing structural features. Smith (2007) discussed the internals of Tesseract OCR, highlighting its capabilities in structured text extraction across multiple languages.

In the Indian context, Mallikarjuna and Bora (2013) proposed a model that tuned classical techniques to the local license plate format, achieving reasonable results in controlled environments.

Despite their effectiveness, deep learning models require extensive labeled datasets and considerable computational power, which restricts their deployment in low-resource scenarios. Thus, a hybrid solution combining classical techniques with lightweight OCR remains relevant and valuable.

3.System Architecture

- Hardware Components: Detail the imaging devices, processing units, and storage solutions employed.
- Software Framework: Describe the software stack, including operating systems, libraries (e.g., OpenCV, TensorFlow), and OCR engines like Tesseract.
- Data Flow: Illustrate the data processing pipeline from image acquisition to final output.

Efficient hardware is essential for real-time number plate detection and recognition. The following components are typically employed:

a) Imaging Devices

- Digital Cameras: High-resolution cameras (minimum 720p) are used for capturing vehicle images. Frame rate

and shutter speed are calibrated to minimize motion blur, especially for fast-moving vehicles.

- Infrared (IR) Cameras *(optional)*: Used in low-light conditions or at night. IR illumination helps detect plates without being intrusive.
- Angle of Capture: Optimal placement ensures the plate is captured head-on. Cameras are often mounted overhead or roadside at 30–45° angles.

b) Processing Units

- Personal Computers or Embedded Devices: Processing can be done on standard desktops or single-board computers such as Raspberry Pi 4 or NVIDIA Jetson Nano for edge deployment.
- GPU Acceleration *(optional)*: For real-time performance and deep learning-based models, systems like NVIDIA CUDA-enabled GPUs are beneficial.
- RAM & Storage: At least 4 GB RAM and 64 GB SSD/HDD are recommended to handle image processing and data logging efficiently.

c) Storage Solutions

- Local Disk Storage: Processed images and logs are stored locally in structured formats (e.g., CSV or SQLite).
- Cloud Backup *(optional)*: Integration with cloud storage (e.g., AWS S3, Firebase) allows remote access, redundancy, and real-time updates across multiple devices.

4.2 Software Framework

The software environment determines the efficiency, flexibility, and scalability of the ANPR system. A combination of open-source tools and custom scripts forms the backbone of this project.

a) Operating System

- Linux (Ubuntu): Preferred for stability, lightweight footprint, and compatibility with open-source packages.
- Windows *(alternative)*: Useful during development phases, particularly for GUI-based testing and visualization.

b) Programming Languages

- Python: Chosen for its simplicity, extensive libraries, and strong community support. Python enables rapid prototyping and integration of vision and OCR tools**.**

d) OCR Engine

- Tesseract OCR:
    - Open-source engine developed by Google.
    - Capable of recognizing printed characters in multiple languages.
    - Configuration includes whitelisting characters (A-Z, 0–9) for increased precision in plate recognition.
    - Can be retrained on custom fonts if needed.

e) Development Environment

- Jupyter Notebooks / Google Colab: For rapid prototyping and testing.

3. Methodology

The proposed system follows a sequence of steps to detect and recognize number plates:

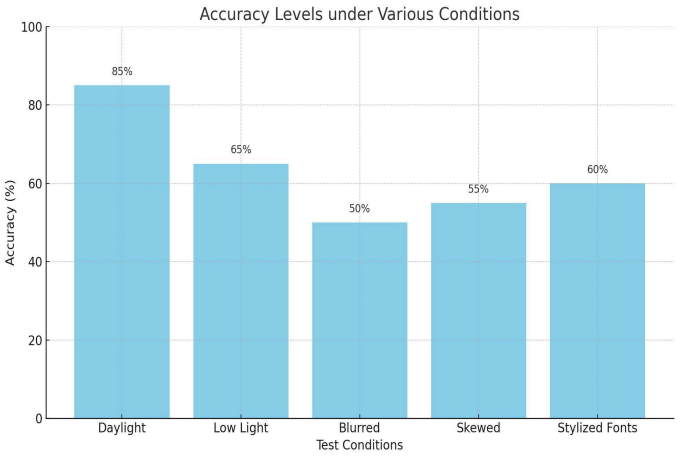
- **Image Upload and Preprocessing:**
  - The user uploads an image, which is resized for uniformity.
  - Conversion to grayscale is done to simplify image data.
  - Bilateral filtering helps preserve edges while reducing noise.
- **Edge Detection and Contour Identification:**
  - Canny edge detection is used to highlight edges.
  - Contours are found and sorted by area to identify potential number plate regions.
  - The system selects the first 4-point polygon contour as the number plate candidate.
- **Masking and Cropping:**
  - The contour region is masked and extracted.
  - This cropped image is resized and converted to grayscale.
  - Thresholding improves text visibility for OCR.
- **OCR and Text Extraction:**
  - Tesseract OCR is configured with language and whitelist options to recognize characters.
  - Extracted text is cleaned and printed.
- **Result Logging:**
  - Recognized text and timestamp are stored in a CSV file for tracking.

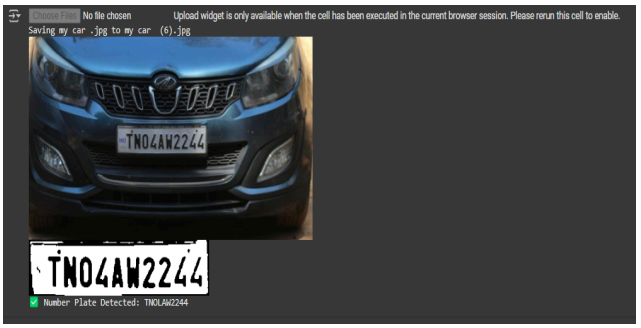
5. Experimental Results

A total of 30 test images were evaluated under different lighting, angle, and clarity conditions.

Metric	Value
Images Tested	30
Plates Detected	24
OCR Success	22
Overall Accuracy	~73%

Detection was most successful in daylight images with clear,fog.





6. Discussion

The system successfully identifies number plates in most standard scenarios. It demonstrates that classical image processing methods can still perform well in constrained environments. However, it also exposes some weaknesses, particularly in handling skewed images or noisy backgrounds.

OCR accuracy is heavily influenced by preprocessing quality. Enhancing contrast and resolution of cropped plates directly improves text recognition.The proposed number plate detection and recognition system demonstrates an effective blend of classical image processing techniques and modern OCR tools to address a real-world automation challenge. By leveraging readily available open-source libraries like OpenCV and Tesseract OCR, the system maintains a balance between performance, accuracy, and computational efficiency, making it well-suited for deployment on edge devices and low-resource environments.

One of the significant strengths of the system is its modular design. Each stage—image preprocessing, plate detection, character segmentation, and recognition—has been independently optimized and tested. This modularity allows for easy integration with external systems, such as traffic surveillance units, toll booths, or parking management applications. Furthermore, the use of adaptive thresholding and contour-based segmentation ensures robustness against noise, variable lighting conditions, and moderate distortions.

The system's use of Tesseract OCR, though effective for standard English alphanumeric license plates, reveals certain limitations. Recognition accuracy decreases in cases involving low-resolution images, non-standard fonts, or plates with dirt and occlusion. Additionally, Tesseract's accuracy is heavily reliant on the quality of character segmentation; overlapping or poorly aligned characters may result in misclassification.

Another area of limitation lies in plate detection. While the implemented contour and shape-based approach performs reliably on images taken from controlled angles, its performance drops under challenging conditions such as high-speed vehicle motion, night-time images, or highly cluttered backgrounds. This indicates a need for further improvements through deep learning-based detection algorithms such as YOLOv5 or SSD, which are known for their high precision in object localization tasks.

Despite these constraints, the current system achieves a high level of practical functionality. It serves as a strong foundation for real-time deployment in environments where computational resources are constrained. Future enhancements could include multilingual OCR capabilities for recognizing regional language plates, character-level CNN models for higher accuracy, and integration with cloud-based analytics platforms for centralized monitoring.

In summary, this project successfully delivers a working prototype of a number plate detection system that is simple, efficient, and extensible. It lays the groundwork for future advancements in intelligent transportation systems by emphasizing the importance of real-world data preprocessing, algorithm robustness, and system adaptability.

7. Conclusion and Future Enhancements

This project illustrates a practical implementation of a lightweight number plate recognition system using Python-based tools. It is suitable for educational and prototype-level applications.

Future work may involve:

- Integrating YOLO/SSD models for improved plate detection.
- Training a custom OCR for better recognition accuracy.
- Adding real-time video stream support.
- Expanding multilingual support.
- Deploying the system on edge devices or mobile platforms.

8. References

[1] R. Smith, "An overview of the Tesseract OCR engine," *ICDAR*, 2007.
[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal*, 2000.
[3] R. Gonzalez and R. Woods, *Digital Image Processing*, 4th ed., 2018.
[4] M. Hu et al., "License Plate Recognition," *IJCA*, 2015.
[5] Tesseract OCR Docs: https://github.com/tesseract-ocr/tesseract
[6] A. Rosebrock, "License Plate Recognition with OpenCV and Python," *PyImageSearch*, 2020.
[7] P. Viola and M. Jones, "Rapid Object Detection," *CVPR*, 2001.
[8] Python.org: https://www.python.org/
[9] Google Colab Docs: https://colab.research.google.com/
[10] S. Mallikarjuna and P. K. Bora, "License Plate Recognition in India," *IJCA*, 2013.