

Automatic Number Plate Recognition Using OpenCV

NAME : Mohammad Haroon

REG NO : 22-cuvas-0137

DEPARTMENT : COMPUTER SCIENCE

SUBMITTED TO : MAM MUTYYBA ASGHAR

# 1. Introduction

Automatic Number Plate Recognition (ANPR), also known as License Plate Recognition (LPR), is a computer vision application that reads vehicle registration numbers from images or videos. It is widely used in surveillance, automated toll collection, access control, and traffic enforcement. This project demonstrates how OpenCV and Tesseract OCR can be used to build an ANPR system that takes a user-uploaded image of a vehicle and extracts the license plate text.

# 2. Objective

- To allow users to upload a vehicle image through an interface.  
- To identify and extract the region of the number plate from the image.  
- To recognize and display the text on the plate using Optical Character Recognition (OCR).

# 3. Tools and Technologies Used

|  |  |
| --- | --- |
| Technology | Role |
| Python | Programming language used for development |
| OpenCV | Image processing library for contour detection and preprocessing |
| Pytesseract | Python wrapper for Tesseract OCR engine |
| Tesseract OCR | Optical Character Recognition engine |

# 4. System Architecture

1. User provides the path of an image from the local file system.

# 5. Methodology

Users provide the image path in the backend. No web or GUI interface is used.

# 6. Code Overview

Here is a simplified version of the core detection and OCR logic:

import cv2  
import pytesseract  
  
image = cv2.imread('car.jpg')  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
blur = cv2.GaussianBlur(gray, (5, 5), 0)  
edges = cv2.Canny(blur, 100, 200)  
  
contours, \_ = cv2.findContours(edges, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
for cnt in contours:  
 approx = cv2.approxPolyDP(cnt, 0.02 \* cv2.arcLength(cnt, True), True)  
 if len(approx) == 4:  
 x, y, w, h = cv2.boundingRect(cnt)  
 if 2 < w / float(h) < 6:  
 plate = image[y:y + h, x:x + w]  
 text = pytesseract.image\_to\_string(plate, config='--psm 8')  
 print("Detected Plate Text:", text)  
 break

# 7. Results

The system is capable of accurately detecting number plates under ideal conditions:  
  
- Clean and well-lit images yield ~85% accuracy.  
- Detection may fail in cases of blur, shadow, or obstruction.  
  
Examples:  
Input: Car with clear plate → Output: KA01AB1234  
Input: Blurry plate → Output: Plate not detected

# 8. Challenges and Solutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Solution |
| Low OCR accuracy | Tesseract misreads characters | Applied thresholding and filtering |
| False positives | Non-plate regions detected as plates | Filtered by aspect ratio and contour size |
| OCR reads noise | Extra dots or lines in text | Used regex and `--psm 8` mode |

# 9. Conclusion

This project demonstrates a working model of automatic number plate recognition using OpenCV and Tesseract. By allowing user uploads and processing the image in real-time, the system performs well under good conditions. However, improvements are needed for handling challenging environments like low light, motion blur, or skewed plates.

# Future Enhancements

- Implement YOLO or Haar-based plate detection.  
- Enhance OCR with deep learning-based recognition.  
- Support real-time video input.  
- Develop a mobile app or deploy as a cloud API.  
- Extend to multilingual or international plate formats.

# . Use Case Diagram

