 ****

**Loyola INSTITUTE OF TECHNOLOGY & SCIENCE**

**LOYOLA NAGAR, THOVALAI – RAJAVOOR ROAD, KANYAKUMARI – 629302**

**Approved by AICTE, New Delhi and Affiliated to Anna University.**

DEPARTMENT OF INFORMATION TECHNOLOGY

Completed the Project named as

**Number Plate Detection**

*Submitted by,*

**TEAM MEMBERS**

1. **N.ANU SANTIYA-(961222205007)**
2. **S.AVITHA-(961222205008)**
3. **M.BOOMALINI-(961222205009)**
4. **S.KIRUTHIGA-(961222205024)**

**INTRODUCTION:**

The increasing number of vehicles worldwide has led to the need for more advanced and automated systems to monitor and manage traffic. Manual methods of recording and verifying vehicle number plates are often inefficient, error-prone, and not scalable. Automatic Number Plate Recognition (ANPR) is a modern solution that uses computer vision and artificial intelligence to identify vehicles by reading their license plates from images or video footage.

ANPR systems typically involve multiple stages, such as vehicle detection, license plate localization, character segmentation, and character recognition. These systems are widely used in applications like traffic monitoring, toll collection, parking access, and law enforcement.

With advancements in image processing and machine learning technologies, ANPR has become more accurate, real-time, and cost-effective. This project aims to implement a simple yet robust ANPR system using Python and OpenCV, demonstrating the core functionality of detecting and recognizing license plates from real-time or static images.

**ABSTRACT:**

Automatic Number Plate Recognition (ANPR) is an intelligent system that enables the automatic identification of vehicle registration numbers using image processing and OCR (Optical Character Recognition). It is an essential component in modern Intelligent Transportation Systems (ITS) and finds applications in traffic enforcement, toll collection, and vehicle tracking systems.

This project focuses on the development of an ANPR system capable of detecting a vehicle’s license plate and recognizing the alphanumeric characters present on it. The system utilizes OpenCV for image processing, and OCR (using Tesseract) to recognize the characters. The main contribution of this project is to present an efficient and affordable approach that can be used in real-time scenarios with decent accuracy.

**OBJECTIVES:**

1. **To develop a functional ANPR system** that can automatically detect and read license plates from images or video streams.
2. **To implement image pre-processing techniques** to enhance license plate detection and character clarity.
3. **To apply Optical Character Recognition (OCR)** for recognizing the characters from the segmented license plate.
4. **To reduce human effort** in monitoring vehicle movements and improve operational efficiency in various applications.
5. **To test and validate** the system across different conditions like varying lighting, plate sizes, and camera angles.
6. **To ensure modularity and flexibility**, so the system can be integrated into real-world applications like toll booths, smart parking, and security gates.

**TOOLS & TECHNOLOGIES:**

1. **Python** – High-level programming language used to implement the entire system due to its simplicity and powerful libraries.
2. **OpenCV** – An open-source computer vision library used for image processing, contour detection, and manipulation.
3. **Tesseract OCR** – Optical Character Recognition engine used to read text (characters) from license plates.
4. **NumPy** – Library for numerical operations, especially useful for image arrays.
5. **Matplotlib/Imshow** – Optional tools for visualizing images during testing or debugging.
6. **Hardware (optional)** – Webcam or IP camera for real-time detection.
7. **Jupyter Notebook / VS Code / PyCharm** – Development environments for writing and testing code.

**IMPLEMENTATION CODE:**

import cv2

import pytesseract

# Path to tesseract executable

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe' # Change path if needed

# Load the image

image = cv2.imread("car.jpg") # Replace with your image file

# Resize the image (optional)

image = cv2.resize(image, (600, 400))

# Convert to grayscale

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Apply edge detection

edged = cv2.Canny(gray, 170, 200)

# Find contours based on edges

contours, \_ = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

# Sort contours by area and keep the largest ones

contours = sorted(contours, key=cv2.contourArea, reverse=True)[:30]

plate = None

for c in contours:

# Approximate the contour

peri = cv2.arcLength(c, True)

approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

if len(approx) == 4: # Check for rectangular contour

x, y, w, h = cv2.boundingRect(c)

plate = gray[y:y + h, x:x + w]

cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 3)

break

if plate is not None:

# OCR on extracted plate region

text = pytesseract.image\_to\_string(plate, config='--psm 8')

print("Detected Number Plate Text:", text.strip())

cv2.imshow("Number Plate", plate)

else:

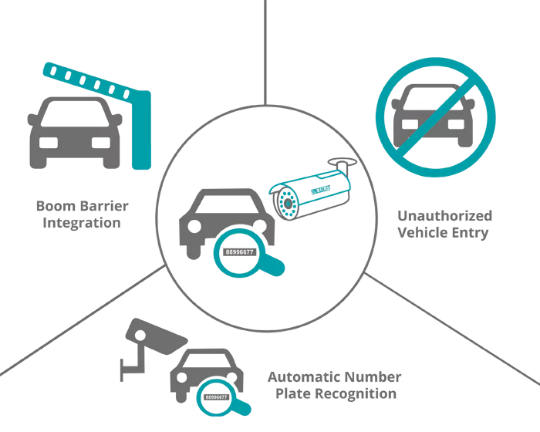
print("Number plate not detected.")

cv2.imshow("Original Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**EXAMPLE REAL-TIME DETECTION SNAPSHOT**:



**✅ ADVANTAGES OF NUMBER PLATE DETECTION:**

* **Automation** – Reduces the need for manual vehicle monitoring.
* **Efficiency** – Processes large volumes of vehicles quickly and accurately.
* **Scalability** – Easily integrated into large traffic systems and smart cities.
* **Security** – Helps in identifying stolen vehicles and unauthorized access.
* **Cost-effective** – Once implemented, it reduces long-term operational costs.

**⚠️ DISADVANTAGES OF NUMBER PLATE DETECTION:**

* **Lighting and Weather Sensitivity** – Accuracy may drop in poor lighting or weather conditions.
* **Plate Style Variations** – Non-standard plates (fonts, colors) can affect recognition.
* **Image Quality Dependency** – Low-resolution or blurry images can reduce OCR accuracy.
* **Language Limitation** – Mostly designed for alphanumeric plates; regional characters may need customization.

**🚗 APPLICATIONS OF NUMBER PLATE DETECTION:**

* **Toll Booths** – Automatic billing by reading number plates.
* **Traffic Enforcement** – Detecting red light violations or over-speeding vehicles.
* **Parking Management** – Entry/exit control and billing in parking lots.
* **Border Control & Security** – Monitoring vehicle movement across sensitive areas.
* **Stolen Vehicle Detection** – Alert systems for unauthorized or blacklisted vehicles.
* **Smart Cities** – Used in urban planning and real-time traffic management.

**⭐ KEY FEATURES OF NUMBER PLATE DETECTION:**

* **Real-Time Detection** – Capable of processing live video feeds.
* **High Accuracy OCR** – Converts images to readable text using machine learning.
* **Modular Design** – Easily extendable with databases or cloud services.
* **Language Independent (customizable)** – Can be trained for local scripts or custom fonts.
* **Lightweight Implementation** – Works even on mid-range systems.

**🎯 KEY OBJECTIVES OF NUMBER PLATE DETECTION:**

1. **Automate license plate detection and recognition** from image/video.
2. **Enhance accuracy** under different lighting and camera conditions.
3. **Minimize processing time** to support real-time usage.
4. **Support regional plate formats** (if needed).
5. **Enable integration** with existing security or traffic systems.

**CONCLUSION:**

Automatic Number Plate Recognition (ANPR) is a crucial advancement in the fields of traffic automation and surveillance. This project demonstrates a basic yet effective ANPR system using OpenCV and Tesseract OCR. Despite some limitations under variable conditions, the model shows promising results in controlled environments. With further enhancements like deep learning-based detection (e.g., YOLO or CNNs) and dataset-driven training, this project can evolve into a robust solution for smart city and industrial needs. ANPR represents a step forward in building safer, more efficient, and intelligent transportation systems.

