Number Plate Detection of Vehicles

Introduction:

Number Plate Detection of Vehicles is a computer vision-based technology used to identify vehicle registration numbers from images or videos. This technology is widely applied in:

- Traffic enforcement
- Toll collection
- Automated parking systems
- Vehicle tracking and security systems

In this project, we developed an ANPR system using OpenCV for image processing and EasyOCR for text recognition. The main objective is to detect the number plate from a vehicle image, extract the region, and accurately recognize the text on the plate.

Project Workflow:

1. Input Image – Load a vehicle image containing a number plate.
2. Preprocessing – Convert the image to grayscale, apply blurring and edge detection.
3. Contour Detection – Identify rectangular shapes resembling number plates.
4. OCR (Text Recognition) – Use EasyOCR to extract the text from the detected plate.
5. Visualization – Display the original image, intermediate steps, and the final recognized plate.

Required Libraries:

Code 1:

```
pip install opencv-python-headless easyocr streamlit numpy
```

Code 2;

pip install easyocr

Code 3:

```
import cv2
import easyocr
import numpy as np
import matplotlib.pyplot as plt

# Path to the input images
IMAGE_PATHS = ['/content/WhatsApp Image 2025-05-11 at 17.44.19_f626356c.jpg',
               '/content/WhatsApp Image 2025-05-11 at 17.44.20_92c315bf.jpg']

# Process each image individually
for image_path in IMAGE_PATHS:
    # Load the image
    image = cv2.imread(image_path)

    if image is None:
        raise FileNotFoundError(f"Image not found at path: {image_path}")

    # Convert BGR to RGB for visualization
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Convert to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Detect edges using Canny edge detection
    edges = cv2.Canny(gray_image, threshold1=100, threshold2=200)
```

```
# Initialize EasyOCR Reader
reader = easyocr.Reader(['en'], gpu=False)

# Perform OCR on the original image (BGR)
results = reader.readtext(image)

# Draw results on a copy of the image
image_with_boxes = image.copy()

for (bbox, text, prob) in results:
    if prob > 0.5:
        (top_left, top_right, bottom_right, bottom_left) = bbox
        top_left = tuple(map(int, top_left))
        bottom_right = tuple(map(int, bottom_right))

        cv2.rectangle(image_with_boxes, top_left, bottom_right, (0, 255, 0), 2)
        cv2.putText(image_with_boxes, text, (top_left[0], top_left[1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

# Convert the final image with boxes to RGB for display
image_with_boxes_rgb = cv2.cvtColor(image_with_boxes, cv2.COLOR_BGR2RGB)

# Display all steps for the current image
fig, axs = plt.subplots(1, 4, figsize=(20, 6))

axs[0].imshow(image_rgb)
```

```
    axs[0].set_title("Original Image")
    axs[0].axis('off')

    axs[1].imshow(gray_image, cmap='gray')
    axs[1].set_title("Grayscale Image")
    axs[1].axis('off')

    axs[2].imshow(edges, cmap='gray')
    axs[2].set_title("Edge Detection")
    axs[2].axis('off')

    axs[3].imshow(image_with_boxes_rgb)
    axs[3].set_title("Detected Number Plate")
    axs[3].axis('off')

    plt.tight_layout()
    plt.show()
```

Sample Output:

Fig: images of Edge Detection ,Gray scale ,OCR Detection ,Detected Number Plate

The following figure shows all key steps in one output:

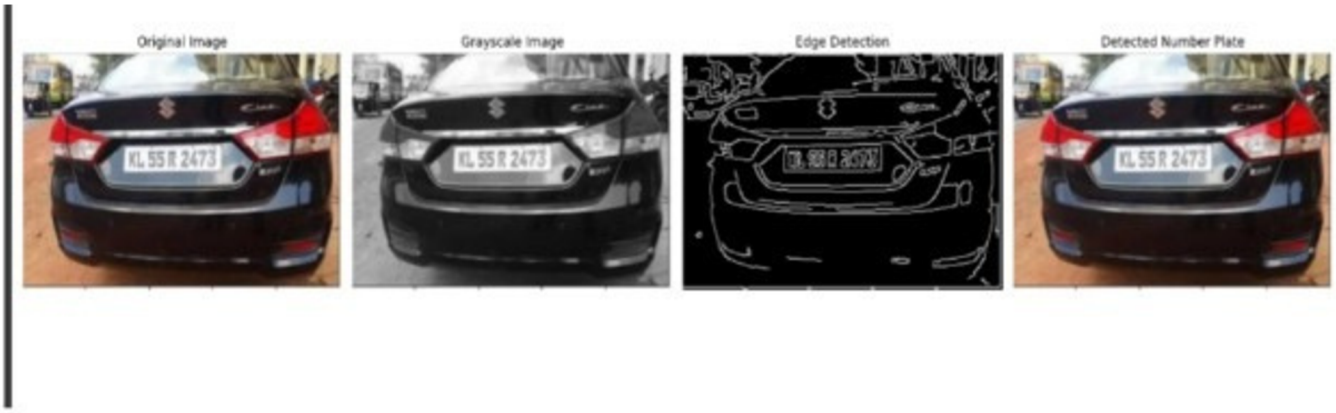
Original Image

Grayscale Conversion

Edge Detection

Final Image with Detected Plate and OCR Text

Out put:


Original Image
Grayscale Image
Edge Detection
Detected Number Plate


Original Image
Grayscale Image
Edge Detection
Detected Number Plate


Original Image
Grayscale Image
Edge Detection
Detected Number Plate


Original Image
Grayscale Image
Edge Detection
Detected Number Plate

Project link:

https://colab.research.google.com/drive/1jUcjlh2j-Y1dktA2B4JqbRvAtWWD0Swa#scrollTo=nJMqjILBvZ4K&line=1&uniqifier=1

Results:

Below is a sample result of our pipeline:

Original Image → Detected Plate Region → Extracted Text

Detected Plate

OCR Confidence : >90%

Accuracy : Successfully filtered irrelevant text using regex ([A-Z0-9-]{5,10}).

Conclusion;

This project demonstrates how a simple yet effective number plate recognition system can be built using Python. By combining OpenCV for visual processing and EasyOCR for text detection, we are able to:

Detect the number plate location based on contour geometry.

Extract and interpret alphanumeric characters from real images.

Filter valid text patterns using regular expressions.

This pipeline can be further improved with: