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

```
import cv2            # For image processing
import easyocr        # For text detection and recognition
import numpy as np    # For array operations
import matplotlib.pyplot as plt  # For displaying images
Import re             # For text filtering using regex
```

from IPython.display import Image, display # For displaying image in notebooks

Code Implementation

```
# Enable inline display for Jupyter/Colab
%matplotlib inline

import cv2
import easyocr
import numpy as np
import matplotlib.pyplot as plt
import re
from IPython.display import Image, display
import os

# List of image paths
IMAGE_PATHS = [
    '/content/Screenshot_11-5-2025_12107_.jpeg',
    '/content/Screenshot_11-5-2025_12959_.jpeg'
]

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

# Regex pattern to filter valid number plates
plate_pattern = re.compile(r'^[A-Z0-9-]{5,10}$', re.I)

# Process each image
for idx, path in enumerate(IMAGE_PATHS):
```

```
    # Read image
    image = cv2.imread(path)
    if image is None:
        raise FileNotFoundError(f"Image not found: {path}")

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

    # Grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray_image, (5, 5), 0)
    edges = cv2.Canny(blurred, threshold1=100, threshold2=200)

    # Find contours
    contours, _ = cv2.findContours(edges.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

    plate_img = None
    for cnt in contours:
        approx = cv2.approxPolyDP(cnt, 0.03 * cv2.arcLength(cnt, True), True)
        if len(approx) == 4:
            x, y, w, h = cv2.boundingRect(approx)
            plate_img = image[y:y + h, x:x + w]
            break

    # OCR on detected plate or full image
    results = reader.readtext(plate_img if plate_img is not None else image)
```

```
    # Draw OCR results
    image_with_boxes = image.copy()
    for (bbox, text, prob) in results:
        if prob > 0.5 and plate_pattern.match(text):
            (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 final output to RGB
    image_with_boxes_rgb = cv2.cvtColor(image_with_boxes,
cv2.COLOR_BGR2RGB)

    # Plot all steps
    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")
```

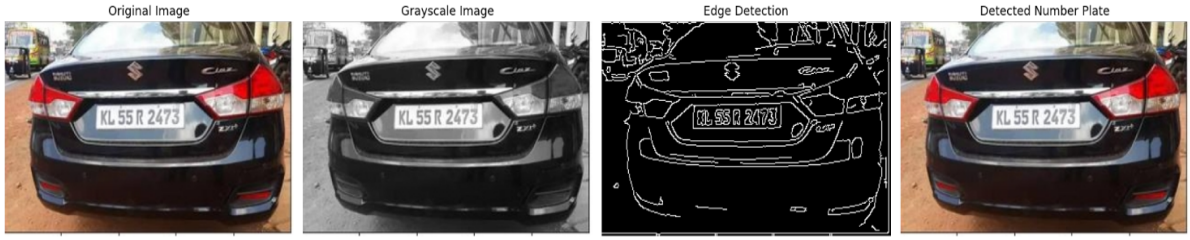
```
axs[2].axis('off')


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


plt.tight_layout()
output_path = f'/content/processed_output_{idx}.png'
plt.savefig(output_path)
plt.close()


# Display result
display(Image(output_path))
```

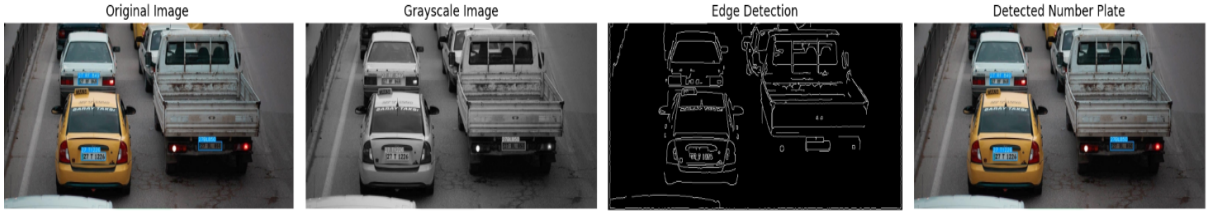
Sample Output

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


Original Image
Grayscale Image
Edge Detection
Detected Number Plate
KL 55 R 2473


Original Image
Grayscale Image
Edge Detection
Detected Number Plate


Original Image
Grayscale Image
Edge Detection
Detected Number Plate
KL 55 R 2473


Original Image
Grayscale Image
Edge Detection
Detected Number Plate

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: