الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong
Department of Electrical and Electronic Engineering

PROJECT REPORT

A Lightweight Automatic Number Plate Recognition (ANPR) System

COURSE CODE : EEE-3604

COURSE TITLE : Digital Signal Processing I Sessional

SUBMITTED TO : Engr. Md. Abdul Kader
Associate Professor
Dept. of EEE, IIUC.

SUBMITTED BY :

Matric ID	Names
ET231033	Md. Abdullah Al Muntasir
ET231034	Md. Minhaj Uddin Raiyan
ET231035	Imran Yakub

SEMESTER : 6th (Autumn-25)

SECTION : 6A

DATE OF SUBMISSION : 15 / 12 / 2025

Remark:

Project Title: A Lightweight Automatic Number Plate Recognition (ANPR) System

1. Objectives

a) To understand the fundamental concepts of image processing used in object detection.
b) To learn the application of edge detection and contour analysis for number plate localization.
c) To detect and extract the vehicle number plate region from an input image
d) To recognize alphanumeric characters from the extracted plate using Optical Character Recognition (OCR).
e) To analyze the performance and limitations of the developed system under different conditions.

2. Formulation of Problem

2.1. Project Description

This project focuses on developing an automatic vehicle number plate detection and recognition system using digital image processing techniques and OCR. The system takes a vehicle image as input, preprocesses it to enhance relevant features, localizes the number plate region, and extracts the alphanumeric characters.

Specifications:

- Input: Vehicle image (RGB)
- Processing: Grayscale conversion, noise reduction, edge detection, contour detection
- Output: Detected license plate text (uppercase)

Constraints:

- Works best under adequate lighting conditions
- Limited performance for rotated, blurred, or heavily stylized plates
- Designed primarily for standard English alphanumeric plates

2.2. Engineering P's Addressed:

2.2.1. WP1 – Knowledge Profile (WK3-WK8):

a) **WK3:** Understanding of digital image processing and filtering.

b) **WK4:** Application of pattern recognition and contour analysis.

c) **WK5:** Knowledge of OCR techniques and text recognition.

d) **WK6:** Practical programming skills for implementing algorithms in Python.

e) **WK7:** Use of computer vision libraries and mathematical models for edge detection.

f) **WK8:** Analytical thinking to select appropriate preprocessing and detection methods.

2.2.2. WP2 – Technical Requirements and Conflicts:

a) Must detect rectangular plate regions (size, aspect ratio constraints).

b) High accuracy in text recognition (conflict: speed vs. OCR accuracy).

c) Ability to handle noisy or low-quality images (conflict: denoising may blur edges).

2.2.3. WP3 – Solution Methods and Selection:

a) Methods considered: Morphological operations, Machine Learning-based detection, Contour detection.

b) Selected method: Contour detection + OCR using EasyOCR, as it is simple, robust, and effective for most standard plates.

2.2.4. WP4 – Infrequently Encountered Issues:

a) Plates partially occluded by objects or dirt.

b) Poorly lit images or extreme angle distortions.

c) Variations in plate font or background patterns that reduce OCR confidence.

3. Methodology

The methodology describes the detailed step-by-step procedure followed to design and implement the automatic vehicle number plate detection and recognition system.

3.1. Image Acquisition

- A digital image of a vehicle is provided as input to the system.
- The image is uploaded using the Google Colab file upload interface.
- OpenCV is used to read the uploaded image into the program.
- Input validation is performed to ensure the image is correctly loaded before processing.

3.2. Preprocessing

3.2.1. Grayscale Conversion:

- The uploaded RGB image is converted into a grayscale image.
- Grayscale conversion reduces computational complexity by eliminating color information.
- Structural details such as edges and shapes remain preserved after conversion.
- This step prepares the image for effective filtering and edge detection.

3.2.2. Noise Reduction Using Bilateral Filter

- A bilateral filter is applied to the grayscale image to reduce noise.
- Unlike simple smoothing filters, the bilateral filter preserves edges while removing noise.
- Noise reduction helps prevent false edge detection in later stages.
- Filter parameters are selected to balance noise suppression and edge preservation.

3.2.3. Edge Detection Using Canny Operator

- The Canny edge detection algorithm is applied to the denoised image.
- The image is first smoothed internally using a Gaussian filter.
- Gradient magnitude and direction are computed to identify intensity changes.
- Non-maximum suppression removes weak and irrelevant edges.
- Double thresholding and edge tracking ensure only meaningful edges are retained.
- This step highlights strong boundaries such as number plate edges.

3.3. Contour Detection and Selection

- Contours are extracted from the edge-detected image using OpenCV.
- Each contour represents a closed boundary of connected edge pixels.
- Contours are sorted based on their area in descending order.
- The largest contours are examined as potential plate candidates.
- Polygon approximation is applied to simplify contours.
- Contours with four vertices are selected, assuming rectangular number plates.

3.4. Masking and Cropping of Plate Region

- A binary mask is created using the selected contour.
- The mask isolates the number plate region from the background.
- Bitwise operations extract only the plate area from the original image.
- The bounding region of the mask is computed.
- The number plate region is cropped and prepared for OCR processing.

3.5. Fallback Strategy for Plate Localization Failure

- In some cases, a valid rectangular contour may not be detected.
- To handle such situations, a fallback strategy is implemented.
- When contour detection fails, OCR is applied directly to the full grayscale image.
- This prevents complete system failure under challenging conditions.
- The fallback improves overall robustness and reliability.

3.6. Optical Character Recognition Using EasyOCR

- EasyOCR is used to recognize characters from the cropped plate region.
- The OCR model automatically detects text regions.
- Alphanumeric characters are recognized using deep learning models.
- Detected characters are combined to form the plate number.
- All recognized letters are converted to uppercase for consistency.

3.7. Visualization of Results

- The detected number plate text is displayed on the original image.
- The bounding region of the detected plate is highlighted.
- Visual output helps verify system performance and correctness.
- Intermediate outputs are also analyzed for debugging and validation.

4. Investigation (Result & Analysis)

To evaluate the performance of the developed system, experiments were conducted using sample vehicle images. The behavior of the system was observed at each processing stage using a single test vehicle image. Additionally, a failure case was analyzed to verify the fallback strategy.

4.1. Step-by-Step Output for a Test Vehicle Image

The following figures show every intermediate and final outputs obtained during number plate detection and recognition for a sample vehicle image.

4.1.1. Original Input Image

Figure 4.1.1: Original Input Image

4.1.2. Grayscale Image

Figure 4.1.2: Grayscale Image

4.1.3. Noise Reduced Image (Bilateral Filter)

Figure 4.1.3: Noise Reduced Image

4.1.4. Edge Detected Image (Canny)

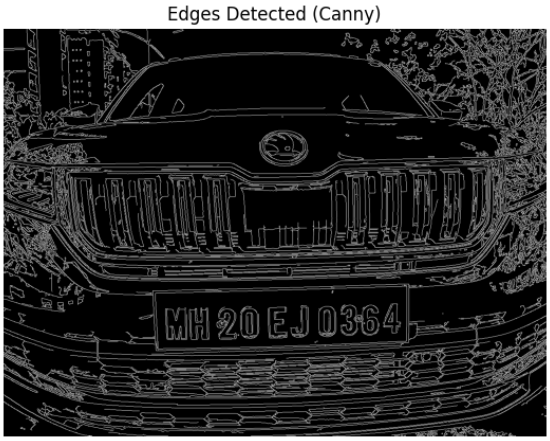

Figure 4.1.4: Edge Detected Image

4.1.5. Contour Detection Output

Figure 4.1.5: Contour Detection Result

4.1.6. Masked Plate Region

Figure 4.1.6: Masked Plate Region

4.1.7. Cropped Number Plate Region

Figure 4.1.7: Original Input Image

4.1.8. Final OCR Output with Detected Text

Figure 4.1.8: Final OCR Output

A few more samples;

4.2. Fallback Strategy Evaluation (Contour Detection Failure Case)

In certain cases, the system fails to detect a valid rectangular contour due to complex backgrounds or low contrast. To handle this situation, a fallback strategy is employed.

4.2.1. Failure in Contour Detection

- No suitable four-sided contour is detected in the edge image.
- Plate localization cannot be performed using contour analysis.

```
[INFO] No number plate contour detected. Using full image for OCR.

[INFO] Detected Text: VNA 453
[INFO] No contour found for visualization.
```

Figure 4.2.1: Terminal Output Figure showing fallback OCR activation

4.2.2. OCR Applied to Full Image (Fallback Mode)

- The system attempts to extract readable text without localization.
- Although accuracy may be reduced, partial or full plate text can still be recovered.

Figure 4.2.2: Original Input Image

5. Conclusion

5.1. Summary:

This project successfully demonstrates an automatic vehicle number plate detection and recognition system using classical image processing and OCR techniques. The system effectively detects number plate regions, extracts relevant features, and recognizes characters with reasonable accuracy.

5.2. Comparison with Other Solutions:

Compared to deep-learning-based detection systems, the proposed approach is computationally efficient and easier to implement, though it sacrifices robustness in challenging conditions. The system can be applied in parking management, traffic monitoring, and access control systems.

5.3. Applications:

- Automatic parking management systems
- Traffic monitoring and law enforcement
- Vehicle access control and security systems
- Toll collection and campus surveillance systems

5.4. Limitations:

- Reduced accuracy in low-light or night images

Figure 5.4.1: Low Light

- OCR errors for blurred or partially visible plates

Figure 5.4.2: Low Quality - Blurry

- No correction for rotated or tilted plates

Figure 5.4.3: Angular Distortion

- Performance degrades with complex backgrounds

Figure 5.4.4: Complex Background

- Not optimized for Bangla plates or stylized fonts

Figure 5.4.5: Language

5.5. Future Development:

- Integration of rotation and skew correction techniques
- Enhancement of performance under low-light conditions
- Support for Bangla license plates and multilingual OCR
- Adoption of deep-learning-based plate detection models
- Real-time implementation using video input
- Accuracy improvement through adaptive preprocessing methods

6. Appendix – Google Colab

```
# @title A Lightweight ANPR System
import cv2
import easyocr
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files

print("[INFO] Please upload an image containing a vehicle number plate:")
uploaded = files.upload()

for fn in uploaded.keys():
    image_path = fn
    print(f"[INFO] Image uploaded: {image_path}")

image = cv2.imread(image_path)
if image is None:
    raise ValueError("[INFO] Error reading image. Please upload a valid image
file.")

plt.figure(figsize=(8,5))
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')
plt.show()

# Grayscaling
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(8,5))
plt.imshow(gray, cmap='gray')
plt.title("Grayscale Image")
plt.axis('off')
plt.show()
```

```
# Bilateral Filter
gray = cv2.bilateralFilter(gray, 11, 17, 17)
plt.figure(figsize=(8,5))
plt.imshow(gray, cmap='gray')
plt.title("After Bilateral Filter (Noise Reduction)")
plt.axis('off')
plt.show()

# Canny Edge Detection
edged = cv2.Canny(gray, 30, 200)
plt.figure(figsize=(8,5))
plt.imshow(edged, cmap='gray')
plt.title("Edges Detected (Canny)")
plt.axis('off')
plt.show()

# Contour Detection
cnts, _ = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:10]

image_contours = image.copy()
cv2.drawContours(image_contours, cnts, -1, (0, 255, 0), 2)
plt.figure(figsize=(8,5))
plt.imshow(cv2.cvtColor(image_contours, cv2.COLOR_BGR2RGB))
plt.title("Contours Detected")
plt.axis('off')
plt.show()

screenCnt = None
for c in cnts:
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.018 * peri, True)
    if len(approx) == 4:
        screenCnt = approx
        break

# Masking & Cropping Number Plate
if screenCnt is not None:
    mask = np.zeros(gray.shape, np.uint8)
    new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1)
    new_image = cv2.bitwise_and(image, image, mask=mask)

    (x, y) = np.where(mask == 255)
    (topx, topy) = (np.min(x), np.min(y))
    (bottomx, bottomy) = (np.max(x), np.max(y))
    cropped = gray[topx:bottomx+1, topy:bottomy+1]

    plt.figure(figsize=(8,5))
```

```
    plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
    plt.title("Detected Plate Region (Masked)")
    plt.axis('off')
    plt.show()

    plt.figure(figsize=(8,5))
    plt.imshow(cropped, cmap='gray')
    plt.title("Cropped Number Plate (for OCR)")
    plt.axis('off')
    plt.show()
else:
    print("[INFO] No number plate contour detected. Using full image for OCR.")
    cropped = gray

# OCR Recognition
reader = easyocr.Reader(['en'])
result = reader.readtext(cropped)

text = " ".join([d[1] for d in result]).strip()
text = text.upper()
print(f"\n[INFO] Detected Text: {text}")

# Visualizing OCR Results
if screenCnt is not None:
    cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)
    x, y, w, h = cv2.boundingRect(screenCnt)
    cv2.putText(image, text, (x, y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2)
else:
    print("[INFO] No contour found for visualization.")

plt.figure(figsize=(10,6))
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title(f"[INFO] Final Detected Number: {text}")
plt.axis('off')
plt.show()
```