is059rfbo

April 24, 2025

[9]:
```
!pip install opencv-python
!pip install pytesseract
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-
packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-
packages (from opencv-python) (2.0.2)
Requirement already satisfied: pytesseract in /usr/local/lib/python3.11/dist-
packages (0.3.13)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.11/dist-
packages (from pytesseract) (11.1.0)
```

[10]:
```
import cv2
import pytesseract
import os
import numpy as np
import matplotlib.pyplot as plt
```

[11]:
```
def image_read(image1, image2, title1="", title2=""):
    fig = plt.figure(figsize=(15, 15))
    ax1 = fig.add_subplot(121)
    ax1.imshow(image1, cmap="gray")
    ax1.set(xticks=[], yticks=[], title=title1)
    ax2 = fig.add_subplot(122)
    ax2.imshow(image2, cmap="gray")
    ax2.set(xticks=[], yticks=[], title=title2)
```

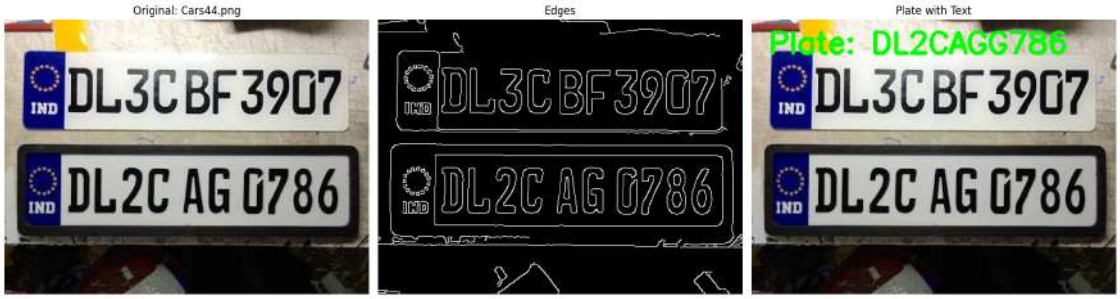
[12]:
```
# Initialize results list
results = []
```

[13]:
```
# Loop through all images in the 'dataset' folder
for filename in os.listdir("dataset"):
    if filename.lower().endswith((".jpg", ".jpeg", ".png")):
        path = os.path.join("dataset", filename)
        image = cv2.imread(path)
        if image is None:
```

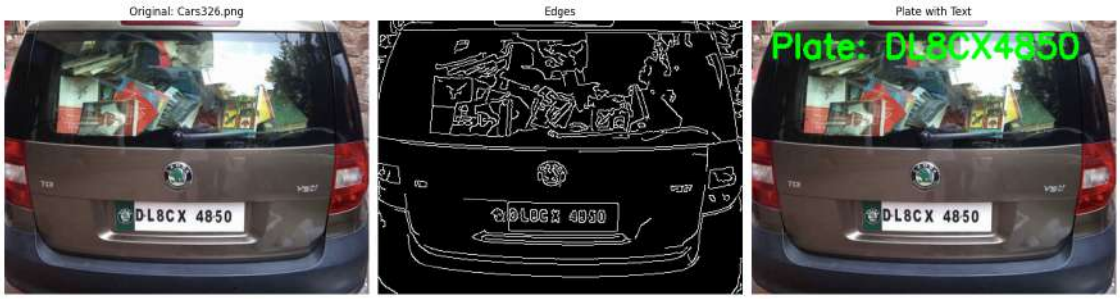
```
            continue



        # Convert the image to grayscale
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)


        # Perform edge detection
        edges = cv2.Canny(gray, 50, 150)
        # image_read(gray, edges, title1="Gray and Blur(Smooth)",␣
↪title2="Edges")

        # Find contours
        contours, _ = cv2.findContours(edges.copy(), cv2.RETR_LIST, cv2.
↪CHAIN_APPROX_SIMPLE)
        image_copy = image.copy()
        _ = cv2.drawContours(image_copy, contours, -1, (255, 0, 0), 2)
        # image_read(edges, image_copy, title1="Edges", title2="Contours")

        contours = sorted(contours, key=cv2.contourArea, reverse=True)[:5]
        image_reduced = edges.copy()
        _ = cv2.drawContours(image_reduced, contours, -1, (255, 0, 0), 2)
        # image_read(image_copy, image_reduced, title1="Original",␣
↪title2="Reduced")

        # Initialize license plate variable
        license_plate = None
        for contour in contours:
            perimeter = cv2.arcLength(contour, True)
            approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)
            if len(approx) == 4:
                license_plate = approx
                break

        # If a license plate is found, extract it and perform OCR
        if license_plate is not None:
            mask = np.zeros(gray.shape, np.uint8)
            cv2.drawContours(mask, [license_plate], 0, 255, -1)
            plate_image = cv2.bitwise_and(image, image, mask=mask)

            # Convert to grayscale and apply OCR
            plate_image_gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY)
            config = r'--oem 3 --psm 7 -c␣
↪tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
```

```
            plate_number = pytesseract.image_to_string(plate_image_gray,␣
↪config=config)
            print(f"License Plate Number for {filename} ==========>␣
↪{plate_number.strip()}")

            # Save the visual output with original, edges, and license plate␣
↪text
            fig, axs = plt.subplots(1, 3, figsize=(18, 6))

            # Original image
            axs[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
            axs[0].set_title(f"Original: {filename}")
            axs[0].axis("off")

            # Edges detected image
            axs[1].imshow(cv2.cvtColor(edges, cv2.COLOR_BGR2RGB))
            axs[1].set_title("Edges")
            axs[1].axis("off")

            # Plate image with OCR text
            annotated = image.copy()
            cv2.putText(annotated, f"Plate: {plate_number.strip()}", (20, 40),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.2, (0, 255, 0), 3)
            axs[2].imshow(cv2.cvtColor(annotated, cv2.COLOR_BGR2RGB))
            axs[2].set_title("Plate with Text")
            axs[2].axis("off")

            plt.tight_layout()
            plt.show()

            # Save plate image
            plate_path = f"plates/{os.path.splitext(filename)[0]}_plate.png"
            cv2.imwrite(plate_path, plate_image)

            results.append({
                "filename": filename,
                "plate_number": plate_number.strip(),
                "plate_image": plate_path
            })

        else:
            print(f"License plate not found in {filename}")
            results.append({
                "filename": filename,
                "plate_number": "Not found",
                "plate_image": "None"
            })
```

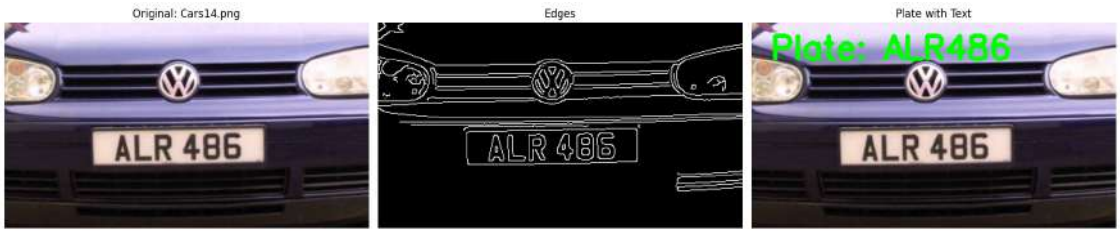
License Plate Number for Cars44.png ==========> DL2CAGG786



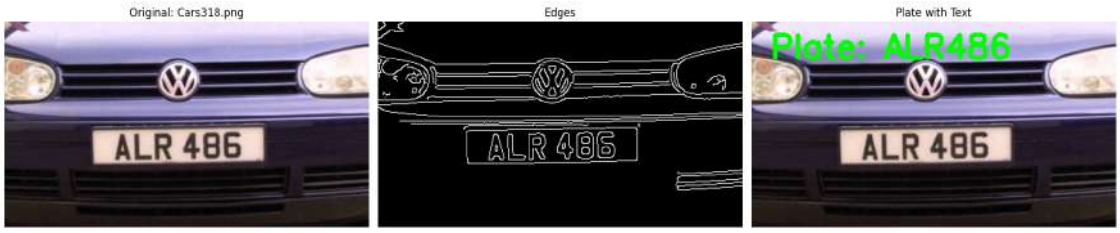
License Plate Number for Cars326.png ==========> DL8CX4850



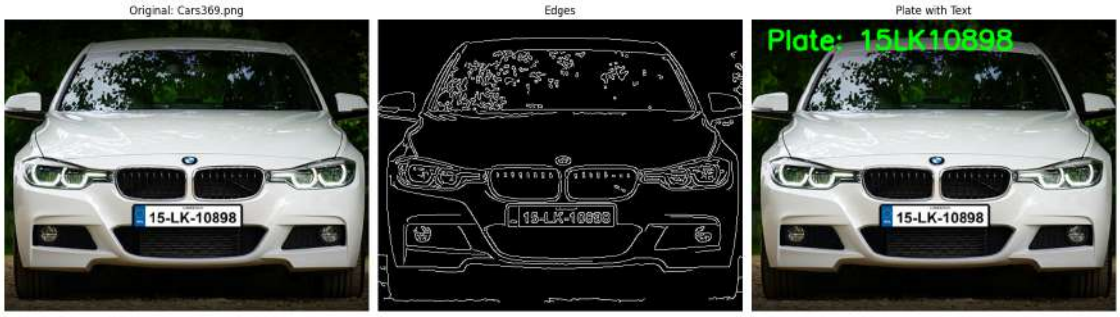
License Plate Number for Cars14.png ==========> ALR486



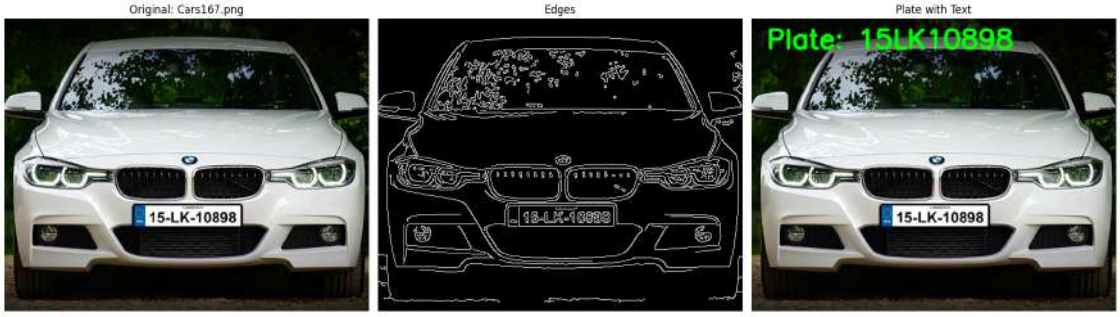
License Plate Number for Cars318.png ==========> ALR486



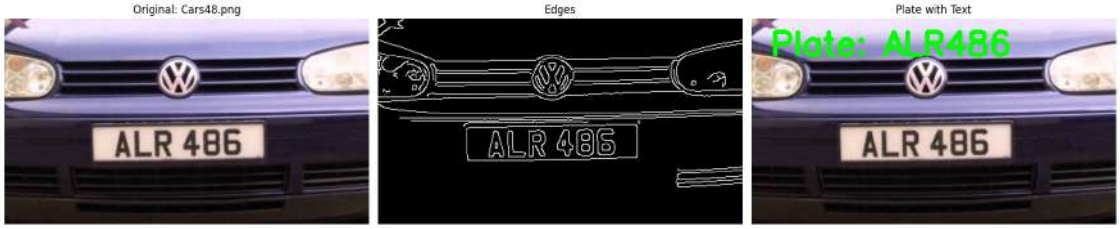
License Plate Number for Cars369.png ==========> 15LK10898



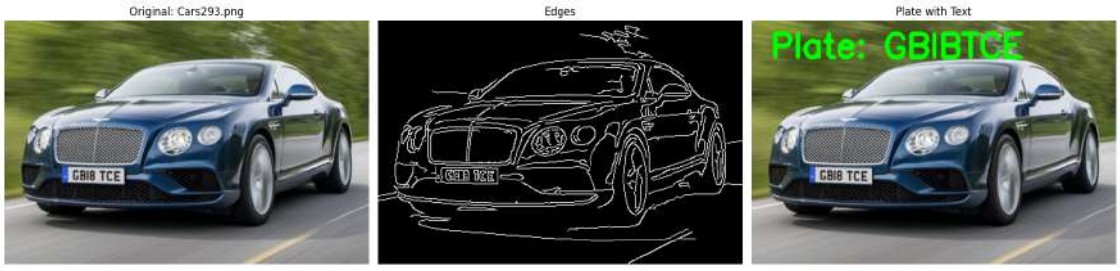
License Plate Number for Cars167.png ==========> 15LK10898



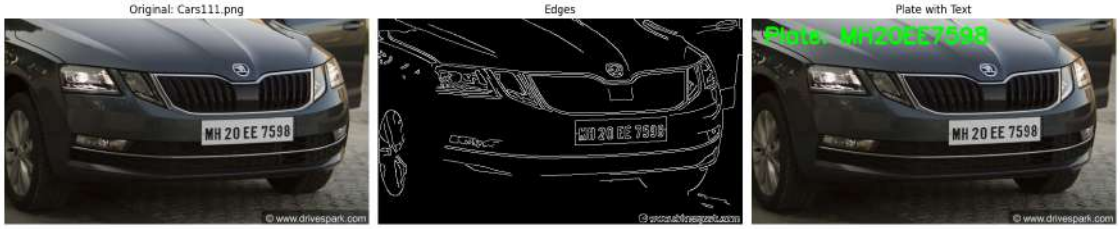
License Plate Number for Cars48.png ==========> ALR486



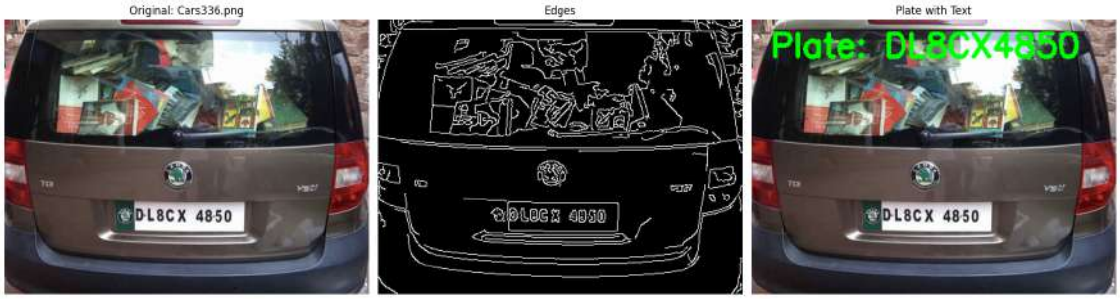
License Plate Number for Cars293.png ==========> GBIBTCE



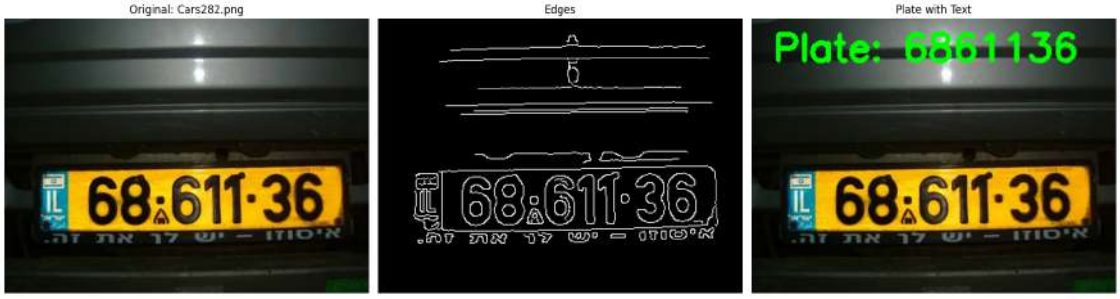
License Plate Number for Cars111.png ==========> MH20EE7598



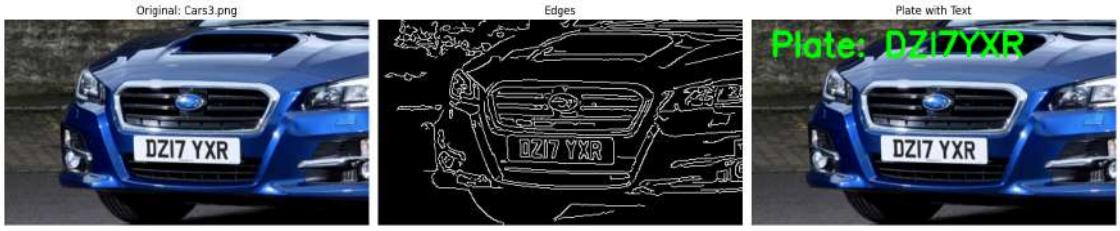
License Plate Number for Cars336.png ==========> DL8CX4850



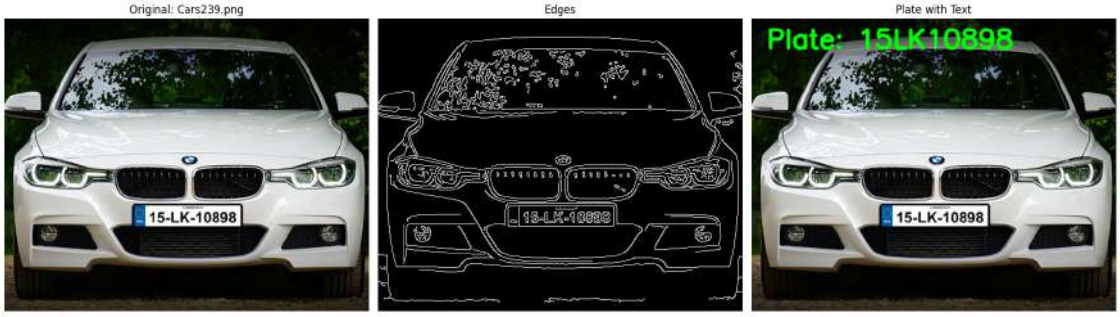
License Plate Number for Cars282.png ==========> 6861136



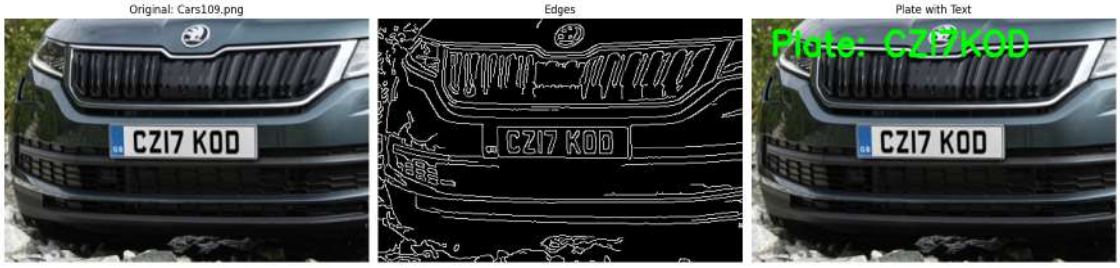
License Plate Number for Cars3.png ==========> DZI7YXR



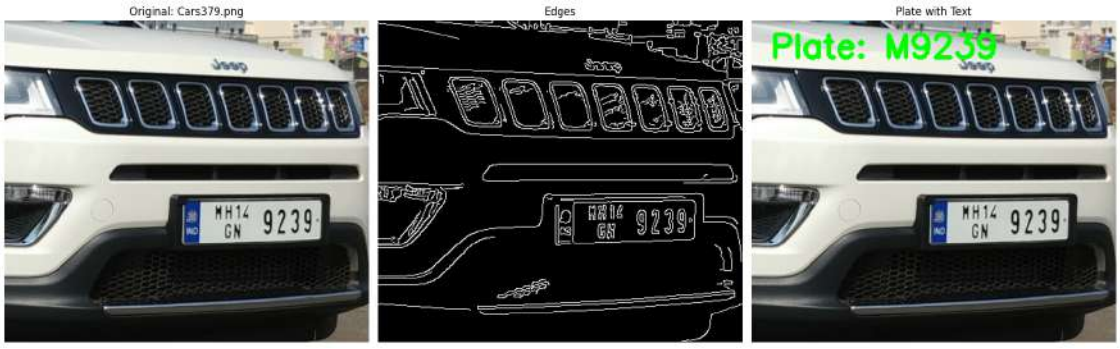
License Plate Number for Cars239.png ==========> 15LK10898



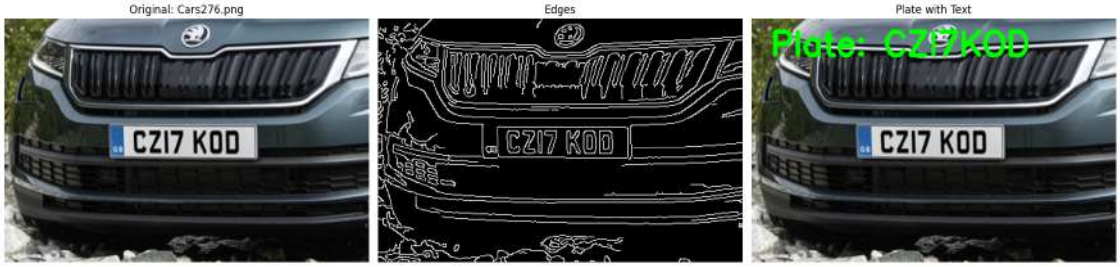
License Plate Number for Cars109.png ==========> CZI7KOD



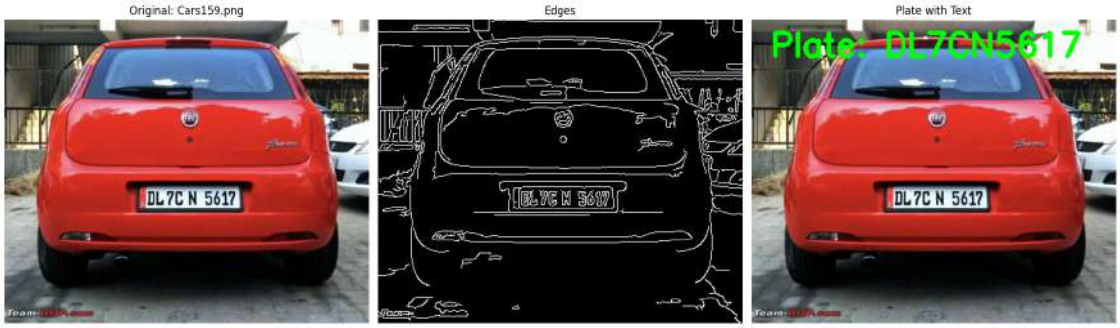
License Plate Number for Cars379.png ==========> M9239



License Plate Number for Cars276.png ==========> CZI7KOD

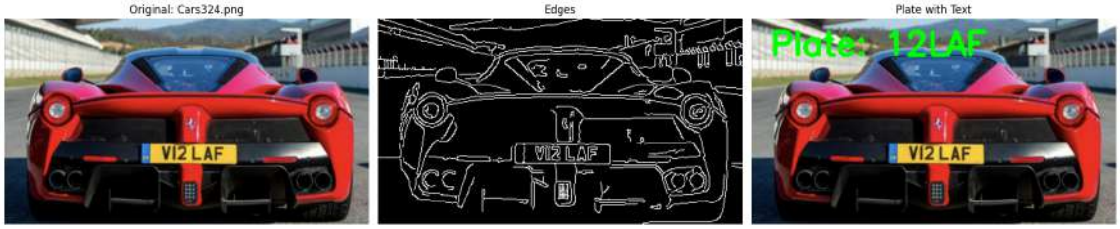


License Plate Number for Cars159.png ==========> DL7CN5617

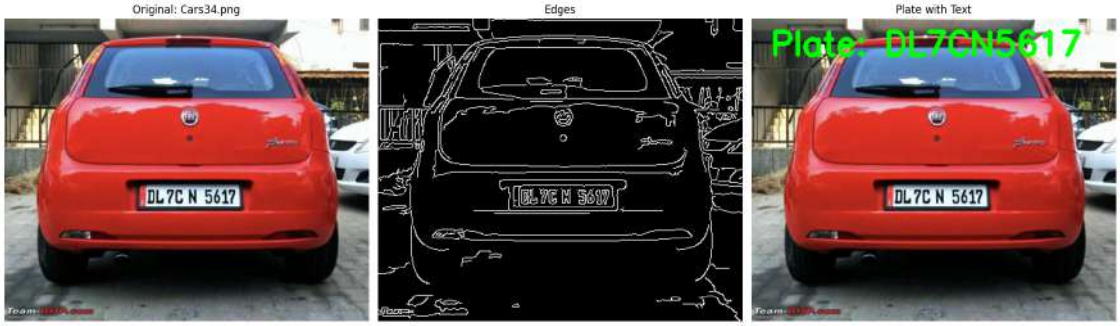


License Plate Number for Cars204.png ==========> NBYOND

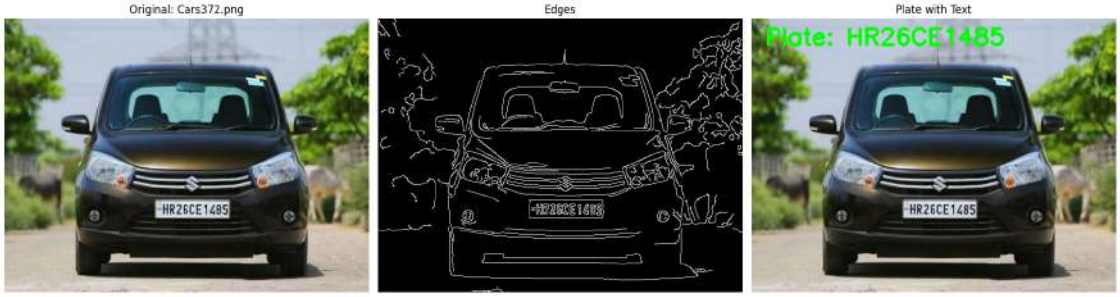
License Plate Number for Cars324.png ==========> 12LAF



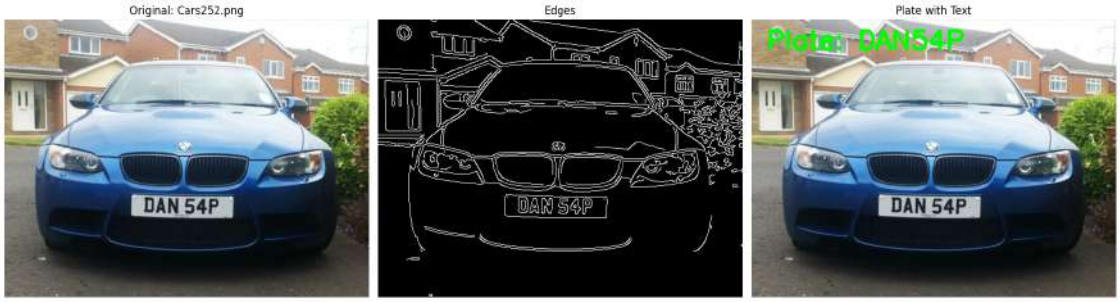
License Plate Number for Cars34.png ==========> DL7CN5617



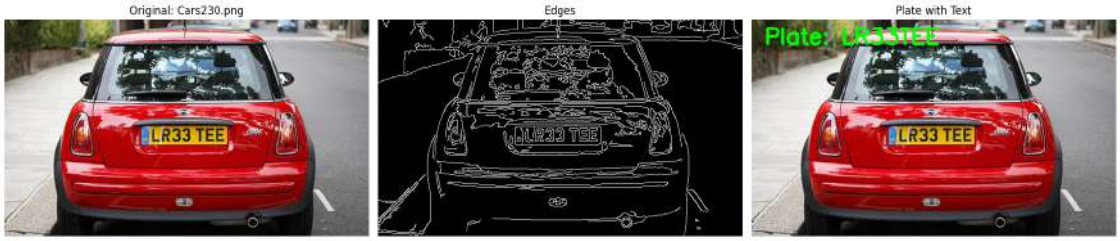
License Plate Number for Cars372.png ==========> HR26CE1485



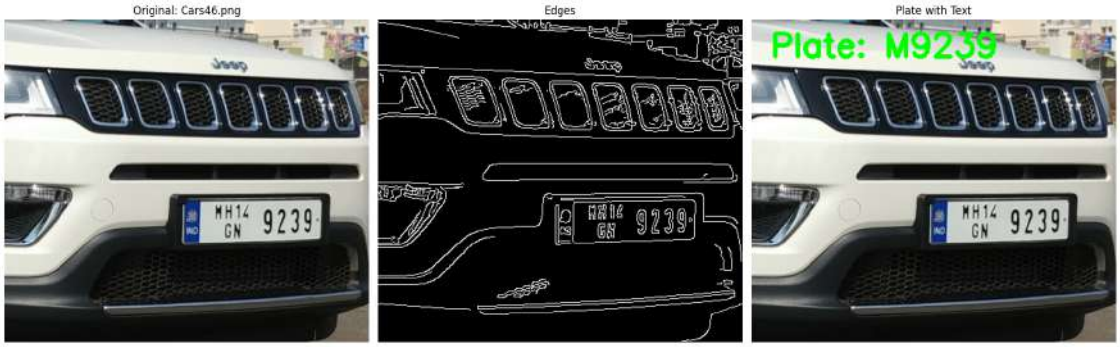
License Plate Number for Cars252.png ==========> DAN54P



License Plate Number for Cars230.png ==========> LR33TEE

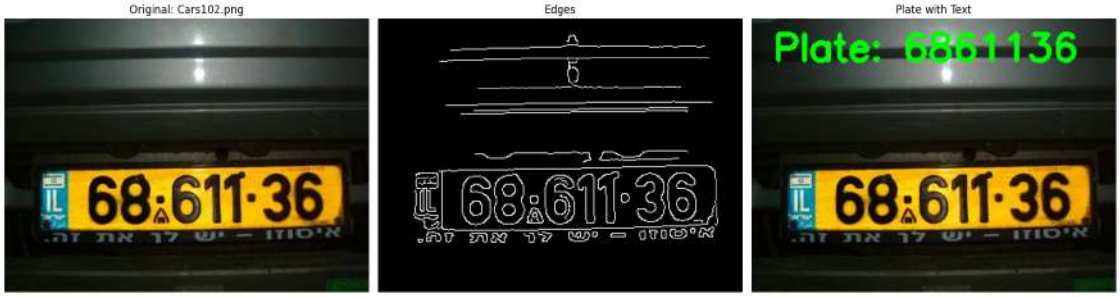


License Plate Number for Cars46.png ==========> M9239

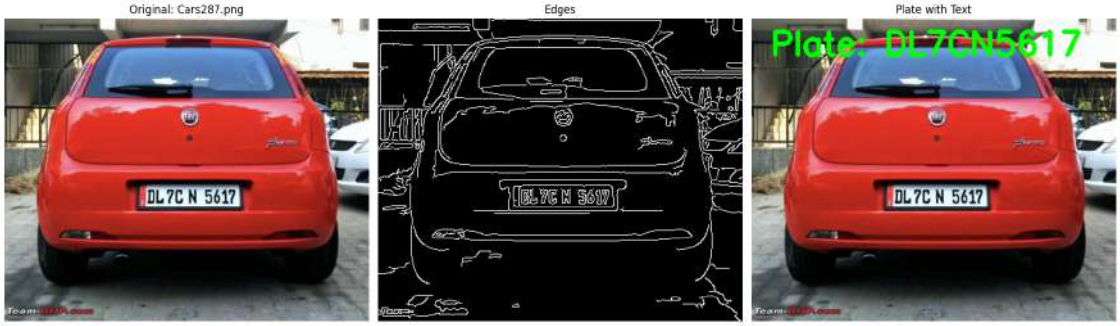


License Plate Number for Cars27.png ==========> DZI7YXR

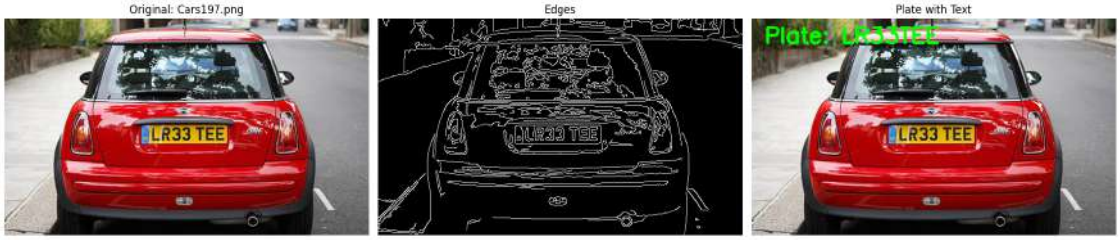
License Plate Number for Cars102.png ==========> 6861136



License Plate Number for Cars287.png ==========> DL7CN5617

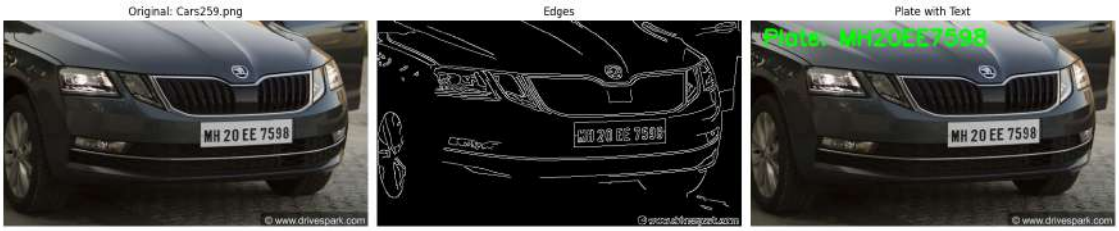


License Plate Number for Cars197.png ==========> LR33TEE



License Plate Number for Cars412.png ==========> LR33TEE

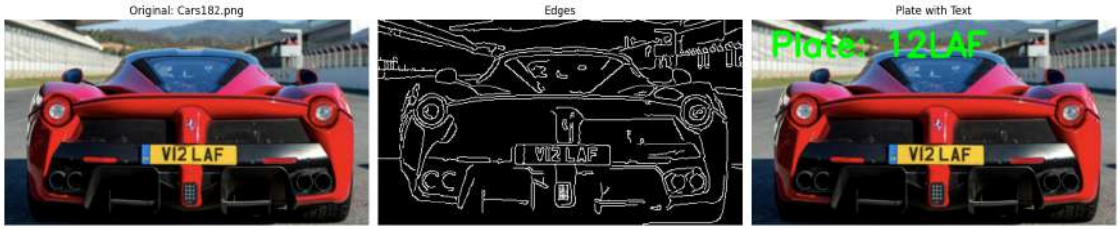
License Plate Number for Cars259.png ==========> MH20EE7598



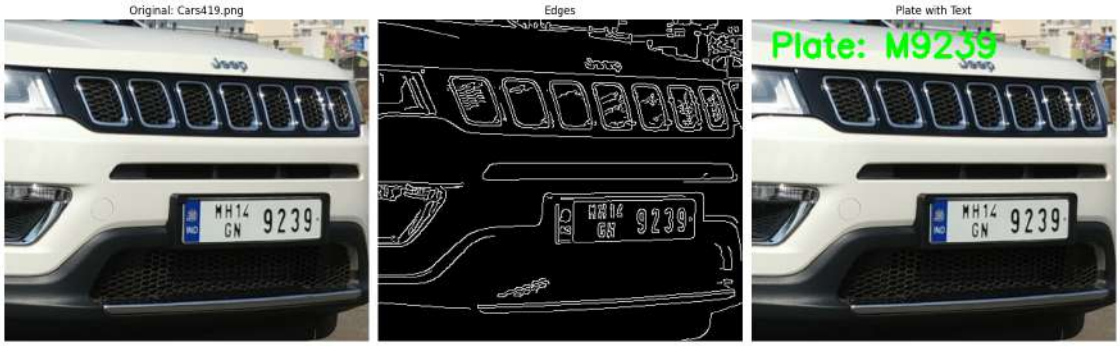
License Plate Number for Cars212.png ==========> KL65H 4383

License Plate Number for Cars417.png ==========> BAD231

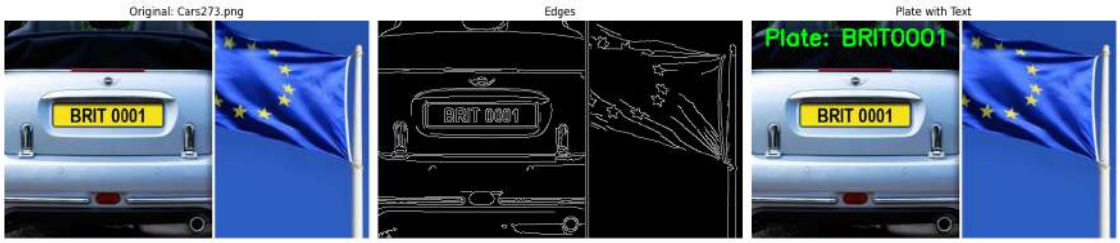
License Plate Number for Cars182.png ==========> 12LAF



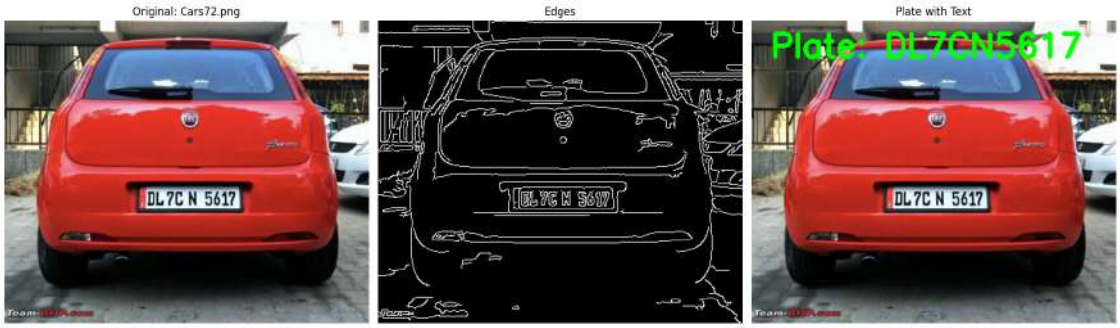
License Plate Number for Cars419.png ==========> M9239



License Plate Number for Cars273.png ==========> BRIT0001

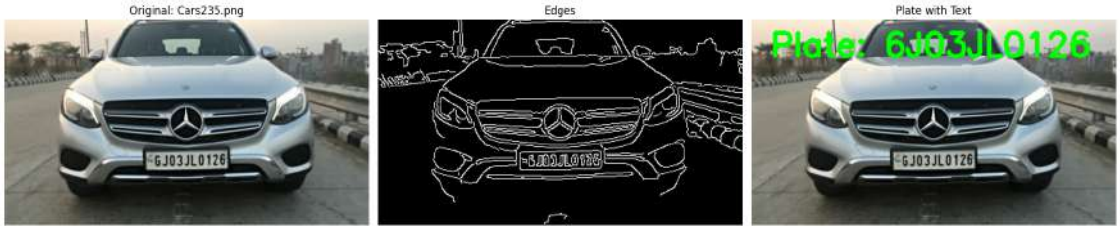


License Plate Number for Cars72.png ==========> DL7CN5617

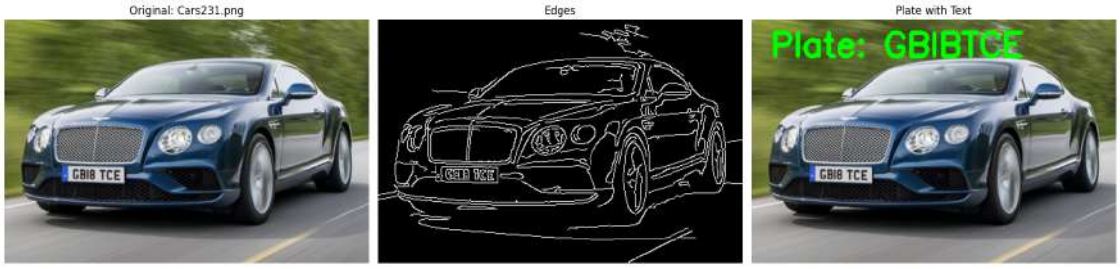


License Plate Number for Cars6.png ==========> 80210

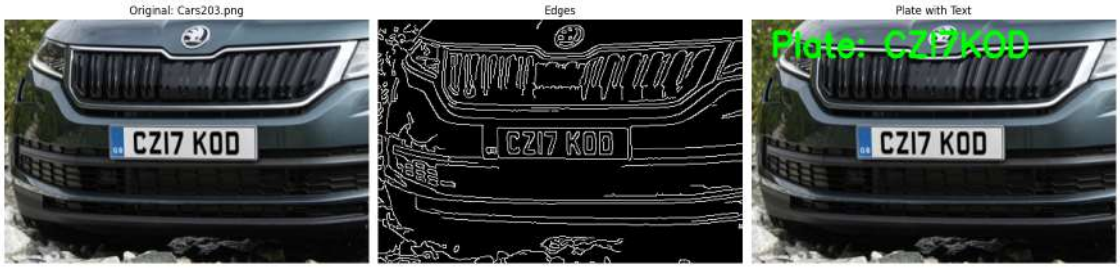
License Plate Number for Cars235.png ==========> 6J03JL0126



License Plate Number for Cars231.png ==========> GBIBTCE

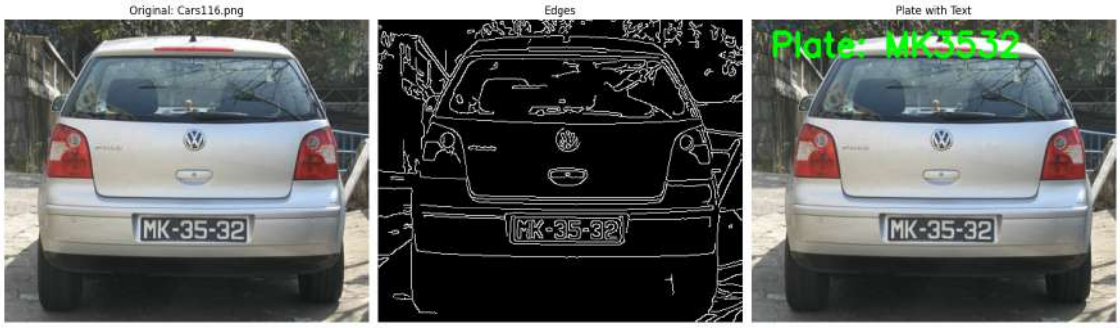


License Plate Number for Cars203.png ==========> CZI7KOD

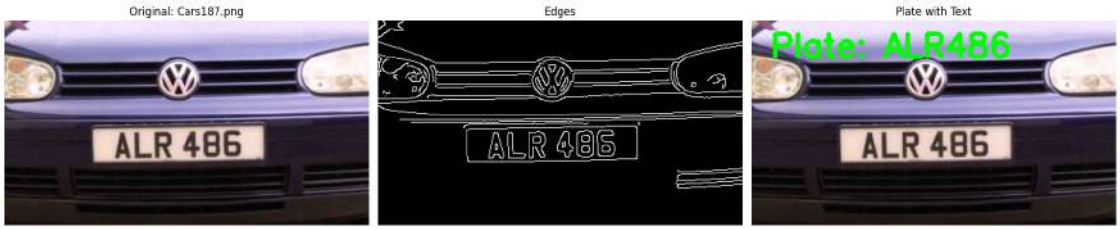


License Plate Number for Cars211.png ==========> NS

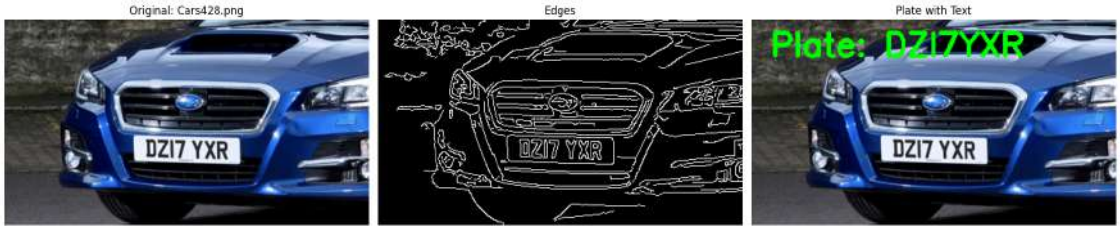
License Plate Number for Cars116.png ==========> MK3532



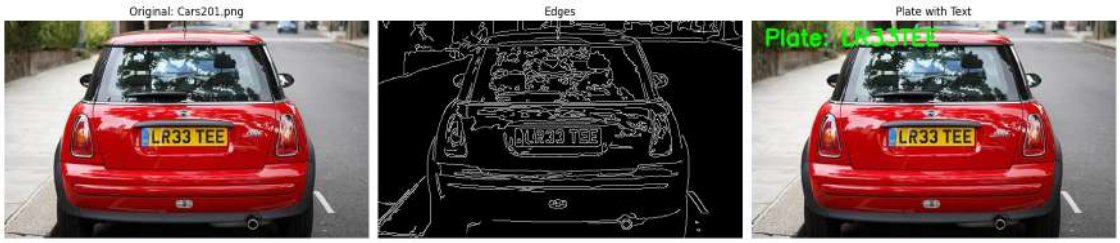
License Plate Number for Cars187.png ==========> ALR486



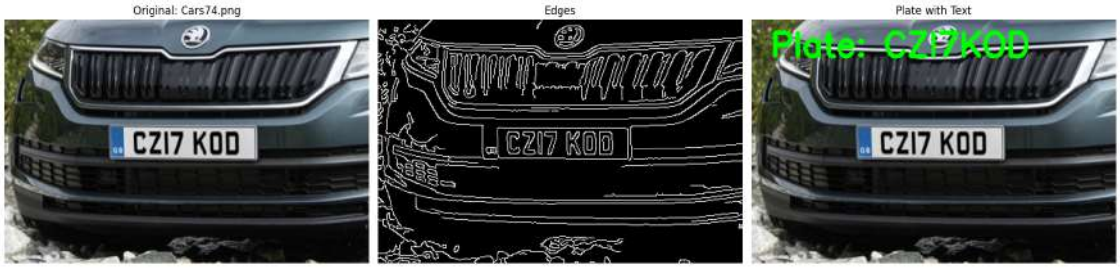
License Plate Number for Cars428.png ==========> DZI7YXR



License Plate Number for Cars201.png ==========> LR33TEE

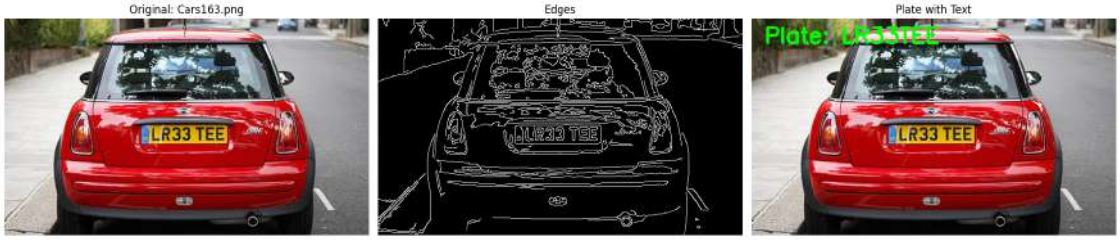


License Plate Number for Cars74.png ==========> CZI7KOD

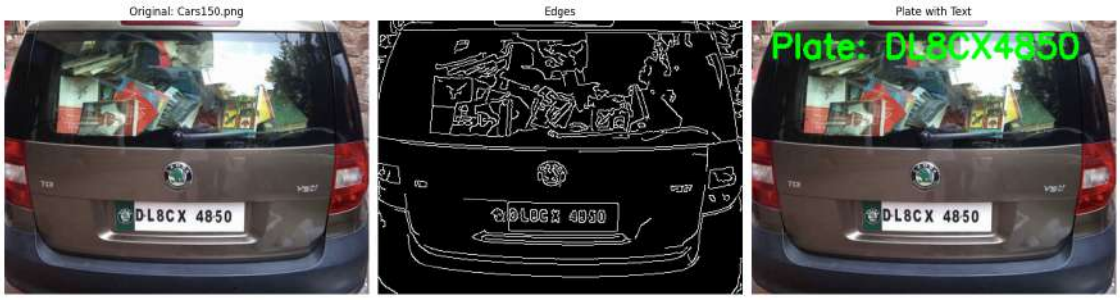


License Plate Number for Cars80.png ==========> BAD231

License Plate Number for Cars163.png ==========> LR33TEE

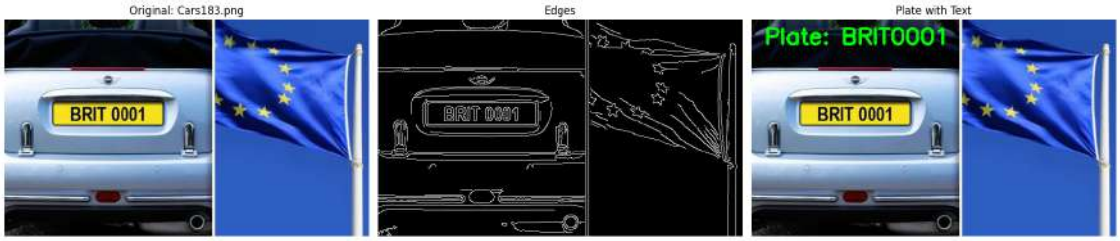


License Plate Number for Cars150.png ==========> DL8CX4850



License Plate Number for Cars332.png ==========> DZI7YXR

```
License Plate Number for Cars183.png ==========> BRIT0001
```



Automated Vehicle License Plate Recognition System

1st Pochimireddy Harika
Lovely Professional University,
Punjab, India
harika@gmail.com

2nd *T K Santhosh Rao*
Lovely Professional University
Punjab, India
tksanthoshrao@gmail.com

***Abstract*— With the increasing volume of vehicles on the road, efficient and automated vehicle identification systems have become crucial for smart traffic management and law enforcement. This project presents a classical image processing-based approach for Indian vehicle number plate recognition using OpenCV and Tesseract OCR. The system processes real-world vehicle images and handles moderate variations in noise, illumination, and angle. It follows a structured pipeline comprising grayscale conversion, Gaussian smoothing, Canny edge detection, and contour analysis to localize the license plate region. After isolating the plate, Optical Character Recognition (OCR) is applied to extract alphanumeric characters. To enhance accessibility and user interaction, the system is integrated with a Flask web application. This lightweight Python framework allows users to upload vehicle images directly through a browser, trigger the detection pipeline, and instantly view the recognized number plate. The recognized plate numbers, along with the corresponding images, are stored and exported into a CSV format for record-keeping and further analysis. While the system does not currently use deep learning techniques, it offers a lightweight, explainable, and easy-to-deploy solution well-suited for controlled environments and resource-constrained setups. Future improvements could include the integration of models like YOLO or Faster R-CNN to enhance recognition robustness in real-time and complex scenarios.**

Keywords*—*Number Plate Recognition (NPR), Optical Character Recognition (OCR), Contour Detection, Edge Detection, OpenCV.

I. INTRODUCTION

The growing demand for automation in public infrastructure has led to the widespread integration of intelligent systems across various sectors. One such area is intelligent transportation systems (ITS), where technologies are being developed to address challenges in traffic management, vehicle monitoring, and law enforcement. With the rapid increase in the number of vehicles worldwide, manual monitoring and record-keeping systems have become inefficient and unreliable. To mitigate these issues, the deployment of automatic vehicle identification systems has gained significant traction, especially in smart city initiatives. Among the key technologies supporting ITS, Number Plate Recognition (NPR) also referred to as License Plate Recognition (LPR) plays a pivotal role. It involves the automatic identification of a vehicle by capturing and processing the alphanumeric characters present on its license plate. NPR systems are widely used for various applications, including automated toll collection, parking management, entry/exit control in private or restricted zones, detection of traffic violations, and real-time tracking of stolen or unauthorized vehicles. Despite their widespread adoption, building an effective NPR system remains a technical challenge due to several real-world issues. Variability in lighting conditions, plate orientations, image noise, camera resolutions, and environmental factors such as rain or glare can significantly affect system performance. Furthermore, in countries like India, non-standard license plate formats, custom fonts, and poor plate maintenance further complicate automated recognition tasks. Unlike standardized plates in certain countries, Indian vehicle plates often deviate in structure, size, and clarity, necessitating a robust and adaptable recognition framework. While deep learning-based approaches have shown impressive results in recent years, they typically require large labelled datasets and significant computational resources for training and inference. This makes them unsuitable for real-time or edge-level deployments in cost-sensitive environments. Hence, a reliable and lightweight solution based on traditional image processing techniques is both practical and necessary for widespread implementation. In this study, we propose a classical computer vision-based number plate recognition system using OpenCV for image pre-processing and contour analysis, and Tesseract OCR for text extraction. The proposed methodology is designed to operate efficiently under low resource constraints while maintaining acceptable recognition accuracy. The system follows a multi-step process that includes:

- **Image Acquisition and Pre-processing**: Conversion to grayscale, Gaussian smoothing, and noise reduction to enhance relevant features.
- **Edge Detection and Segmentation**: Use of Sobel and Canny edge detectors to isolate potential regions of interest, followed by contour analysis to identify the number plate area.
- **Plate Localization**: Selection of contours based on geometric filtering and approximation to locate rectangular plate-like structures.
- **Character Recognition**: Use of Tesseract OCR on the extracted plate regions to convert image-based text into machine-readable format.
- **Result Logging**: The recognized text and plate images are stored and exported to a CSV file for database integration or further analysis.

Our system is tested on a dataset of Indian vehicles with diverse challenges including skewed angles, low illumination, and

noisy environments. The goal is not to outperform deep learning-based solutions but to provide a lightweight, interpretable, and easily deployable alternative that still delivers reliable performance for many real-world scenarios.

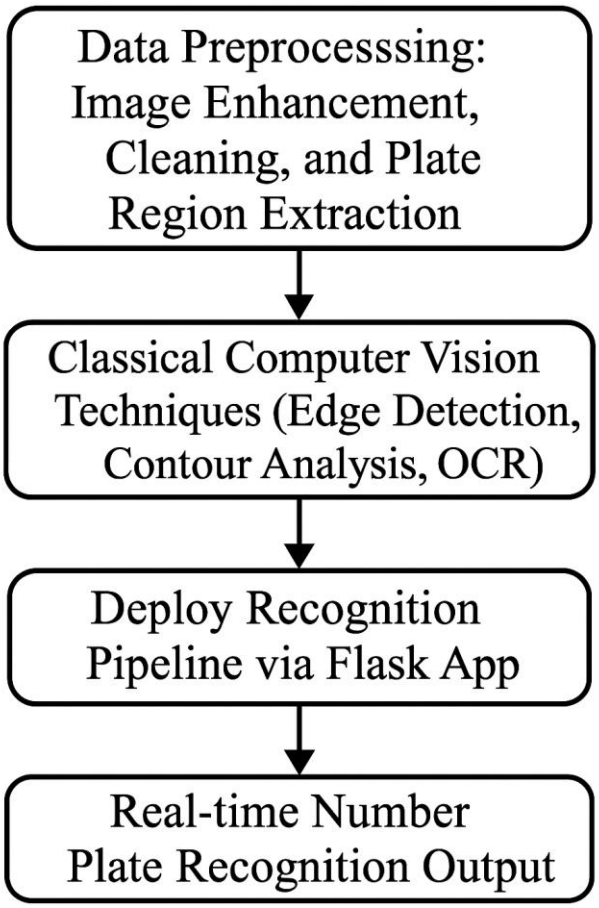


Fig 1. System Architecture for License Plate Detection and Recognition

II. Literature Review

Patel et al. [1] conducted an extensive survey on Automatic Number Plate Recognition (ANPR) systems, outlining the standard processes, methodologies, and challenges associated with developing reliable ANPR solutions. The study classified ANPR systems into three core stages: number plate localization, character segmentation, and character recognition. In their review, the authors explored various techniques implemented at each stage, including edge detection, morphological operations, and region-based approaches for locating number plates, followed by template matching and artificial neural networks for character recognition.[2] Badr et al. [2] proposed a complete Automatic Number Plate Recognition (ANPR) system focused on detecting and recognizing vehicle number plates from images captured under various real-world conditions. The authors structured their system into several sequential stages: image acquisition, pre-processing, license plate detection, character segmentation, and character recognition. In their methodology, Badr et al. utilized Sobel edge detection for locating license plate regions by identifying high-contrast areas, followed by morphological operations to refine the detected regions. For character segmentation, a projection-based approach was adopted to isolate individual characters from the extracted license plate region. The final recognition stage employed template matching techniques to identify alphanumeric characters. The study evaluated the system using a custom dataset of vehicle images taken under diverse lighting and environmental conditions. The results demonstrated satisfactory recognition rates, particularly in controlled environments, while also acknowledging challenges such as poor lighting, image blurriness, and complex backgrounds affecting detection accuracy. Duan et al. [3] presented the development of a complete automatic vehicle license plate recognition system, focusing on practical implementation in real-time traffic surveillance scenarios. The proposed system was organized into three primary modules: plate localization, character segmentation, and character recognition. For license plate localization, the authors employed a combination of edge detection and morphological filtering techniques to accurately identify rectangular regions likely to contain number plates. In the character segmentation phase, they applied a vertical projection method to isolate individual characters from the detected license plate region, ensuring accurate separation even in cases of closely spaced or slightly tilted characters. [4] Qadri and Asif proposed an Automatic Number Plate Recognition (ANPR) system designed for vehicle identification using optical character recognition (OCR) techniques. Their system was structured into four main stages: image acquisition, license plate extraction, character segmentation, and character recognition. The authors utilized edge detection algorithms combined with morphological operations to accurately localize the license plate region within a captured vehicle image. Once the plate region was isolated, a pixel projection method was applied to segment individual characters, ensuring clean separation for the OCR process. For the character recognition phase, the system employed a template matching-based OCR approach, where each segmented character was compared against a predefined set of templates to determine the final output. Chang et al. [5] presented a comprehensive study on the development of an Automatic License Plate Recognition (ALPR) system, specifically designed for intelligent transportation systems applications. The proposed system was structured into three core phases: license plate detection, character segmentation, and character recognition. For the detection phase, the authors employed a combination of edge detection techniques and region-based image processing to accurately locate license plates under varied environmental conditions. The segmentation process utilized adaptive thresholding and connected component analysis to isolate individual characters from the detected plate region. In the recognition stage, a multi-layer perceptron (MLP) neural

network was implemented to classify segmented characters, offering improved robustness over traditional template matching methods. Kaur [6] proposed an Automatic Number Plate Recognition (ANPR) system utilizing classical image processing techniques to accurately detect and recognize vehicle license plates from still images. The system was divided into sequential stages, including image acquisition, preprocessing, license plate detection, character segmentation, and optical character recognition (OCR). In the preprocessing phase, median filtering was applied to remove noise, while edge detection using the Sobel operator facilitated the localization of high-contrast plate regions. For character segmentation, Kaur employed a bounding box technique to isolate each alphanumeric character from the extracted license plate area. The final recognition stage incorporated template matching-based OCR to interpret the segmented characters.

III. Proposed Prediction Model

The proposed Automated Number Plate Recognition (ANPR) system is designed to provide an efficient, lightweight, and explainable solution using classical computer vision techniques and the Tesseract OCR engine. The system is particularly well-suited for controlled or semi-structured environments where resource efficiency and ease of deployment are key priorities. The framework consists of the following modules:

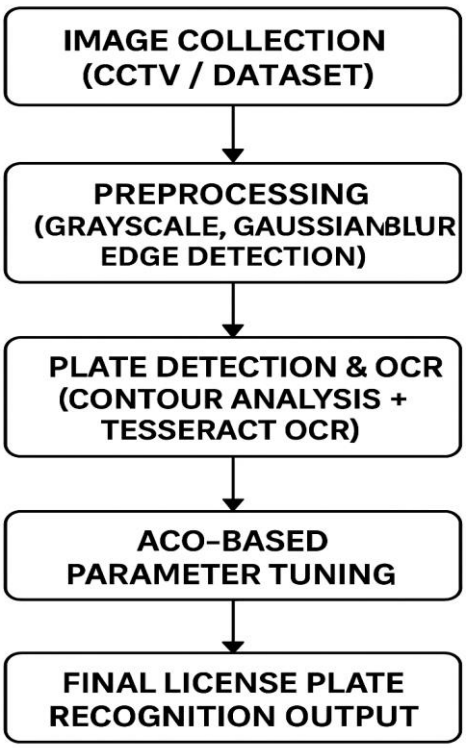


Fig 2. Framework of Proposed Prediction Model.

A. Image Collection

Input Source: CCTV footage or a curated vehicle image dataset.

This module handles the acquisition of vehicle images which serve as the raw input for the ANPR system. The images can be collected from real-time surveillance cameras or publicly available datasets containing various vehicle orientations and environmental conditions. The primary aim is to ensure the system can function across a variety of scenarios such as parking lots, highways, or gated premises.

B. Image Preprocessing

Techniques Used:

- Grayscale Conversion
- Gaussian Blurring
- Edge Detection (Canny)

To prepare the images for license plate detection, the preprocessing stage enhances relevant features while reducing noise and irrelevant details. Images are first converted to grayscale to reduce complexity and then smoothed using Gaussian blur to suppress high-frequency noise. Finally, Canny edge detection is applied to highlight strong edges in the image—edges that are likely to represent license plate boundaries.

C. License Plate Detection & OCR

Techniques Used:

- Contour Detection
- Shape Approximation (Rectangular filtering)
- Tesseract OCR

This module is responsible for locating and reading the license plate. After edge detection, the system finds contours in the image and filters them based on geometric properties like aspect ratio and rectangular shape to localize the plate region. Once a candidate region is identified, it is cropped and passed to **Tesseract OCR**. Tesseract processes the plate image and extracts alphanumeric characters as machine-readable text.

D. ACO-based Parameter Optimization:

Improve performance under varying conditions.

Optimization Focus:

- Edge detection thresholds
- Morphological kernel sizes
- Contour filtering parameters

This optional but powerful module uses the **Ant Colony Optimization (ACO)** algorithm to fine-tune system parameters. Inspired by the behavior of ants searching for optimal paths, ACO dynamically explores various parameter combinations and identifies the best set that leads to improved recognition outcomes. It is especially beneficial for handling images affected by low lighting, noise, or skewed angles by optimizing the detection pipeline without the need for deep learning.

E Final Output – License Plate Recognition Result:

Output Format:

- Alphanumeric text (license plate number)
- Optionally stored in a CSV file with timestamps/image references.

After successful detection and OCR processing, the system produces the final recognized license plate number. These results can be exported in structured formats like CSV for integration with traffic monitoring systems, parking access controls, or law enforcement databases.

IV. Ant Colony Optimization Algorithm

1.Role of Ant Colony Optimization (ACO) in ANPR

Ant Colony Optimization (ACO) is a bio-inspired metaheuristic algorithm that emulates the foraging behavior of ants to solve complex optimization problems. In the context of Automated Number Plate Recognition (ANPR), ACO can be effectively utilized to enhance various stages of the recognition process:

- **Edge Detection and Segmentation**: ACO improves the accuracy of edge detection by simulating the pheromone-laying behavior of ants. Ants traverse the image, depositing pheromones on high-gradient areas, which helps in precisely identifying the boundaries of number plates .
- **Parameter Optimization**: ACO optimizes the hyperparameters of machine learning models used in ANPR, leading to improved recognition accuracy and reduced error rates. By exploring various parameter combinations, ACO identifies the optimal set that enhances system performance .

2. Application in Image Processing for ANPR

In image processing, ACO can be applied to:

- **Edge Detection**: By simulating ant movement over an image, ACO identifies edges based on intensity variations. Ants are dispatched to move on the image, and their movements are driven by local variations in the image's intensity values. This process establishes a pheromone matrix that represents the edge information at each pixel position .
- **Segmentation**: ACO assists in segmenting the number plate from the background by optimizing the selection of pixel clusters that represent the plate region. The pheromone matrix guides the segmentation process, enhancing the accuracy of number plate localization .

3. Benefits of Using ACO in ANPR

Incorporating ACO into the ANPR system offers several advantages:

- **Improved Accuracy**: Enhanced edge detection and segmentation lead to better recognition rates.
- **Robustness**: ACO's adaptability makes the system more resilient to variations in lighting, angle, and image quality.
- **Optimization**: ACO effectively tunes system parameters, reducing errors and improving overall performance.
- For instance, a study demonstrated that an ACO-based number plate recognition system achieved a 94% accuracy rate in character extraction, showcasing its effectiveness in real-world applications

V. Experimental Results

This figure demonstrates the complete pipeline of license plate detection and recognition on multiple vehicle images. The process includes three primary stages:

1. **Original Images** (Left Column):
   The raw input images containing vehicles with visible license plates. These serve as the input for further image processing.
2. **Edge Detection Output** (Middle Column):
   The edge-detected versions of the original images, highlighting prominent contours and features. This step is crucial for isolating the license plate region from the background and is typically implemented using algorithms like Canny edge detection.
3. **Detected Plates with OCR Text** (Right Column):
   The final output of the system showing the localized license plate with the corresponding recognized text overlaid. Optical Character Recognition (OCR) has been applied to extract the alphanumeric content from the plate region.

Fig 1.

Fig 2.

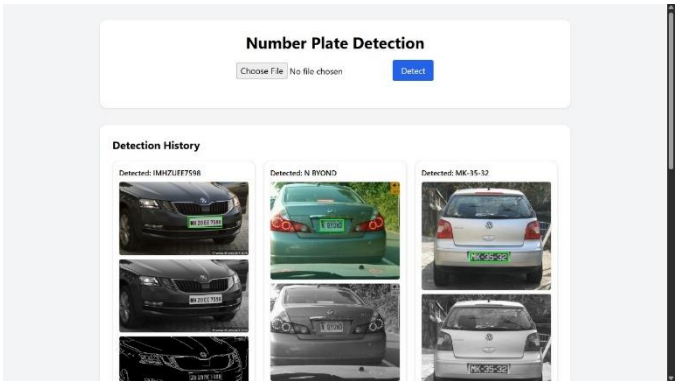


Fig 3.

VI. Conclusion

In this study, an Automatic Number Plate Recognition (ANPR) system was developed and tested using a combination of classical image processing techniques and Optical Character Recognition (OCR) through the Tesseract engine. The implemented system followed a structured pipeline comprising image preprocessing, edge detection, contour extraction, license plate localization, character segmentation, and character recognition. The preprocessing stage converted vehicle images to grayscale and applied Gaussian blur to minimize noise, followed by Canny edge detection for boundary extraction. The system then utilized contour detection and shape approximation to identify potential license plate regions based on their geometric properties. Once a suitable license plate region was localized, a mask was created to isolate the plate area, and OCR was applied to extract the alphanumeric characters. The recognized plate numbers, along with corresponding images, were stored and exported in a structured CSV format for record-keeping and evaluation. The experimental results demonstrated that the system effectively detected and recognized license plates from images taken under favourable conditions, such as clear lighting and standard plate formats. However, challenges persisted in cases involving poor lighting, skewed images, or obscured plates, affecting detection reliability and OCR accuracy. The system's strength lies in its simplicity, lightweight implementation, and ease of deployment for static image analysis and controlled access applications. Future enhancements could involve integrating advanced deep learning-based object detection models (e.g., YOLO, Faster R-CNN) for more robust plate localization and character recognition in complex, real-time environments. This would improve accuracy and adaptability, particularly in dynamic traffic surveillance and smart city systems.

References

[1] Patel, Chirag, Dipti Shah, and Atul Patel. "Automatic number plate recognition system (anpr): A survey." *International Journal of Computer Applications* 69.9 (2013).

[2] Badr, A., Abdelwahab, M. M., Thabet, A. M., & Abdelsadek, A. M. (2011). Automatic number plate recognition system. *Annals of the University of Craiova-Mathematics and Computer Science Series*, *38*(1), 62-71.

[3] Duan, T. D., Du, T. H., Phuoc, T. V., & Hoang, N. V. (2005, February). Building an automatic vehicle license plate recognition system. In *Proc. Int. Conf. Comput. Sci. RIVF* (Vol. 1, pp. 59-63).

[4] Qadri, Muhammad Tahir, and Muhammad Asif. "Automatic number plate recognition system for vehicle identification using optical character recognition." *2009 international conference on education technology and computer*. IEEE, 2009.

[5] Chang, S. L., Chen, L. S., Chung, Y. C., & Chen, S. W. (2004). Automatic license plate recognition. *IEEE transactions on intelligent transportation systems*, *5*(1), 42-53.

[6] Kaur, S. (2016). An automatic number plate recognition system under image processing. *International Journal of Intelligent Systems and Applications*, *8*(3), 14.

[7] Pechiammal, B., & Renjith, J. A. (2017, March). An efficient approach for automatic license plate recognition system. In *2017 third international conference on science technology engineering & management (ICONSTEM)* (pp. 121-129). IEEE.

[8] Mutua, S. M. (2016). *An automatic number plate recognition system for car park management* (Doctoral dissertation, Strathmore University).

[9] Rajput, Hitesh, Tanmoy Som, and Soumitra Kar. "An automated vehicle license plate recognition system." *Computer* 48.8 (2015): 56-61.

[10] Ahmed, Mohammed Jameel, et al. "License plate recognition system." *10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003*. Vol. 2. IEEE, 2003.

[11] Parvin, Shahnaj, Liton Jude Rozario, and Md Ezharul Islam. "Vehicle number plate detection and recognition

techniques: a review." *Advances in Science, Technology and Engineering Systems Journal* 6.2 (2021): 423-438.

[12] Singh, B., Kaur, M., Singh, D., & Singh, G. (2016). Automatic number plate recognition system by character position method. *International Journal of Computational Vision and Robotics*, *6*(1-2), 94-112.

[13] Agbemenu, Andrew S., Jepthah Yankey, and Ernest O. Addo. "An automatic number plate recognition system using opencv and tesseract ocr engine." *International Journal of Computer Applications* 180.43 (2018): 1-5.

[14] Laroca, R., Zanlorensi, L. A., Gonçalves, G. R., Todt, E., Schwartz, W. R., & Menotti, D. (2021). An efficient and layout-independent automatic license plate recognition system based on the YOLO detector. *IET Intelligent Transport Systems*, *15*(4), 483-503.

[15] Arora, M., Jain, A., Rustagi, S., & Yadav, T. (2019). Automatic number plate recognition system using optical character recognition. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *5*(2), 986-992.

[16] Luo, S., & Liu, J. (2022). Research on car license plate recognition based on improved YOLOv5m and LPRNet. *IEEE Access*, *10*, 93692-93700.

[17] Kaur, S., & Kaur, S. (2014). An Efficient Approach for Automatic Number Plate Recognition System under Image Processing. *International Journal of Advanced Research in Computer Science*, *5*(6).

[18] Rashid, M. M., Musa, A., Rahman, M. A., Farahana, N., & Farhana, A. (2012). Automatic parking management system and parking fee collection based on number plate recognition. *International Journal of Machine Learning and Computing*, *2*(2), 94.

[19] Raj, S., Gupta, Y., & Malhotra, R. (2022, March). License plate recognition system using yolov5 and cnn. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 372-377). IEEE.

[20] Tang, J., Wan, L., Schooling, J., Zhao, P., Chen, J., & Wei, S. (2022). Automatic number plate recognition (ANPR) in smart cities: A systematic review on technological advancements and application cases. *Cities*, *129*, 103833.