**Automated License Plate Recognition (ALPR) System Using OpenCV for Detection and EasyOCR for Text Extraction**

**Abstract**

Automatic Number Plate Recognition (ANPR) systems are critical for traffic management, security, and automation, but their performance heavily depends on adaptability to regional plate designs. This paper presents a robust ANPR framework tailored for Ghanaian vehicle number plates, leveraging OpenCV for image preprocessing and Tesseract OCR for character recognition. The proposed methodology addresses challenges such as varying lighting conditions, plate deformations, and background noise by integrating edge detection, morphological operations, and adaptive thresholding to enhance plate localization.

The system operates in three key phases:

1.⁠ ⁠Pre-processing: Converts input images to grayscale, applies noise reduction, and enhances contrast using adaptive thresholding.

2.⁠ ⁠Feature Extraction: Utilizes contour detection and mathematical morphology to isolate the number plate region.

3.⁠ ⁠OCR Recognition: Employs Tesseract OCR to accurately decode characters, optimized for Ghanaian plate fonts and layouts.

Experimental results demonstrate high accuracy in plate detection and character recognition under diverse environmental conditions. The solution is implemented using Python, harnessing OpenCV’s computer vision capabilities and Tesseract’s machine learning-based OCR engine. This approach ensures scalability for real-world applications while minimizing computational overhead.

**Keywords**: ANPR, OpenCV, Tesseract OCR, Edge Detection, Morphological Operations, Feature Extraction.

**Introduction**

Automatic Number Plate Recognition (ANPR) systems have gained significant attention in recent years due to their applications in traffic monitoring, law enforcement, and smart parking systems. Traditional ANPR approaches rely on image processing and optical character recognition (OCR) techniques to detect and extract vehicle license plate information. Among the various tools available, OpenCV (Open Source Computer Vision Library) has emerged as a powerful framework for real-time object detection and image processing tasks. OpenCV provides optimized algorithms for feature extraction, edge detection, and morphological operations, making it a preferred choice for ANPR implementations (Bradski & Kaehler, 2008).

Several studies have explored the use of OpenCV in ANPR systems. For instance, Patel et al. (2020) proposed a license plate detection model using Haar cascades, a machine learning-based approach for object detection. The Haar cascade classifier, trained on the Russian Number Plate Database (GitHub, 2021), demonstrated reasonable accuracy in detecting license plates under varying lighting conditions. However, Haar cascades may struggle with complex backgrounds or distorted plates, necessitating additional preprocessing steps.

To improve text recognition accuracy, modern ANPR systems integrate OCR engines such as Tesseract OCR and EasyOCR. Tesseract, developed by Google, is widely used for its high accuracy in character recognition, while EasyOCR provides a simpler API with support for multiple languages (Smith, 2021). A comparative study by Kumar et al. (2022) showed that while Easyocr performs better on high-resolution images, EasyOCR offers faster processing for real-time applications.

Despite these advancements, ANPR systems for Ghanaian number plates face unique challenges due to variations in plate designs, non-standard fonts, and environmental factors like dust and lighting. This paper presents a robust ANPR system using OpenCV for plate localization and EasyOCR for character recognition, optimized specifically for Ghanaian vehicle plates. The proposed method incorporates edge detection, adaptive thresholding, and morphological transformations to enhance detection accuracy.

The rest of the paper is organized as follows: Section 2 discusses related work, Section 3 details the methodology, Section 4 presents experimental results, and Section 5 concludes with future research directions.

Literature Review Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Study Focus | Methodology | Key Findings | Challenges & Limitations |
| Number Plate Recognition using OpenCV & EasyOCR | Used OpenCV for preprocessing (grayscale conversion, thresholding) and Tesseract OCR for text recognition. | Successfully extracted license plate text under controlled conditions with moderate accuracy. | Performance degraded under poor lighting and occlusions. |
| Comparative Analysis of OCR Techniques | Evaluated traditional OCR techniques and deep learning-based approaches. | Deep learning models outperform Tesseract in complex text scenarios. | Tesseract struggles with handwritten or distorted text. |
| Vehicle Tracking & Authentication using Deep Learning & OCR | Integrated YOLO for vehicle detection with OCR for number plate recognition. | Improved accuracy and robustness in dynamic environments. | High computational cost for real-time applications. |
| Enhanced OCR for Vehicle Number Plate Recognition | Combined OpenCV preprocessing techniques (thresholding, edge detection) with EasyOCR. | Achieved high accuracy under standard conditions, useful for law enforcement. | Motion blur and varying angles affected performance. |
| Preprocessing Techniques for OCR Optimization | Implemented adaptive thresholding, edge detection, and morphological operations. | Improved text recognition accuracy by reducing noise and enhancing character segmentation. | Challenges include multilingual plate recognition and font variations |
| Future Research Directions in OCR & OpenCV | AI-driven denoising, multilingual OCR models, and integration of deep learning. | Improved accuracy and real-time processing in smart surveillance. | Computational efficiency and real-time processing constraints in large-scale applications. |

**Image to text recognition is done with the help of two techniques**:

**1.Tesseract OCR and OpenCV: An Overview**

Tesseract can be used directly via command line, or (for programmers) by using an API to extract printed text from images. Supports multiple languages and scripts.Uses **LSTMs (Long Short-Term Memory networks)** for improved accuracy. Works well for scanned documents, receipts, and number plates. Can be trained for custom fonts and character sets.

**How It Works:**

1. **Preprocessing**: Converts the image to grayscale, applies thresholding, and removes noise.

2. **Segmentation**: Identifies characters, words, and lines in the text.

3. **Recognition**: Uses AI-based models to classify characters and output text.

4. **Post-processing**: Refines results using dictionaries and heuristics.

**2. OpenCV (Open Source Computer Vision Library)**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

OpenCV allows you to perform various operations in the image.

* **Read the Image :**OpenCV helps you to read the image from file or directly from camera to make it accessible for further processing.
* **Image Enhacncement :**You will be able to enhance image by adjusting the brightness , sharpness or contract of the image. This is helpful to visualize quality of the image.
* **Object detection:**As you can see in the below image object can also be detected by using OpenCV , Bracelet , watch , patterns, faces can be detected. This can also include to recognize faces , shapes or even objects .
* **Image Filtering:**You can change image by applying various filters such as blurring or Sharpening.
* **Draw the Image:** OpenCV allows to draw text, lines and any shapes in the images.
* **Saving the Changed Images:**After processing , You can save images that are being modified for future analysis.

**OpenCV** and **Tesseract OCR** work together to detect and recognize vehicle number plates. The process involves **image processing**, **object detection**, and **text recognition** to extract license plate numbers from images or video feeds.

**Vehicle Number Plate Text Recognition**

detecting and recognizing vehicle number plates under various challenging conditions, such as varying speeds, lighting, and angles.

The proposed system comprises three main phases:​

1. **Pre-processing**: This phase involves several image processing techniques:​

o **Grayscale Conversion**: Transforms images to grayscale to eliminate extraneous colors and focus on relevant features

o **Adaptive Thresholding**: Enhances contrast around the number plate to improve visibility.​

o **Morphological Operations**: Removes borders and background noise to isolate the number plate characters.​

2. **Feature Extraction**: In this stage, individual characters and numbers are identified, differentiating among similar letters and digits.​

**3**. **Text Recognition**: Utilizes Optical Character Recognition (OCR) techniques, specifically leveraging Tesseract OCR, to convert extracted features into readable alphanumeric characters.​

Number Plate Recognition (NPR) is a critical application in intelligent transportation systems, facilitating automated vehicle identification for law enforcement, toll collection, and parking management. OpenCV, a widely used computer vision library, enhances NPR by enabling efficient image preprocessing, feature extraction, and text recognition. Key preprocessing techniques such as grayscale conversion, Gaussian blurring, and adaptive thresholding improve the clarity of license plate images, while edge detection and contour-based filtering aid in accurate plate localization. Once extracted, Optical Character Recognition (OCR) using Tesseract converts the license plate characters into machine-readable text. The integration of deep learning models like YOLO and Convolutional Neural Networks (CNNs) further improves recognition accuracy, especially in challenging conditions such as low-light environments and complex backgrounds. Despite advancements, challenges such as real-time processing, multilingual text recognition, and environmental variations persist. Future research aims to refine OCR models, incorporate AI-driven denoising techniques, and optimize computational efficiency for real-time applications in smart cities and intelligent surveillance systems.

**Image-to-Text Generation and Its Evolution:**

**1.** Early OCR systems relied on **pattern matching** and **template-based recognition**. These systems struggled with **handwritten text**, varying fonts, and noisy images. Performance was **low** due to **limited adaptability** to different text styles.

**2.** OCR improved with **machine learning**, where models were trained on **large text datasets**.

**Tesseract OCR** (by Google) introduced **adaptive learning**, improving text recognition. This phase enabled OCR to handle **multiple fonts and moderate distortions**.

**3.** OpenCV’s Role in OCR Performance

**1.Basic Image Processing (Early OpenCV):**OpenCV provided **basicpre-processing** techniques

such as:

o **Grayscale conversion** (reducing noise)

o **Thresholding** (binarization)

o **Edge detection** (better character separation)

**2.OpenCV + Machine Learning**

* OpenCV introduced **HOG (Histogram of Oriented Gradients)** and **SVM (Support Vector Machines)** to detect **characters in complex images**.
* This improved text extraction from **real-world images**.

**3.OpenCV + Deep Learning** (Modern Enhancements)

•  **Deep learning-based OCR pipelines** now integrate OpenCV for:

* **Text detection** using EAST (Efficient and Accurate Scene Text Detector).
* **Text recognition** using LSTM-based models.
* **Image enhancement** (denoising, skew correction) for better OCR performance.

•  OpenCV’s **DNN (Deep Neural Network) module** allows direct integration with **Tesseract OCR**, improving speed and accuracy.

**Enhancing OCR Performance with OpenCV**

Optical Character Recognition (OCR) has significantly evolved with the integration of OpenCV, offering enhanced accuracy and efficiency through advanced image preprocessing and deep learning models. **Preprocessing techniques**play a crucial role in improving OCR accuracy by refining text regions before recognition. OpenCV provides essential image enhancement methods such as **grayscale conversion**, which simplifies text extraction by reducing noise, and **thresholding techniques** like Otsu’s and adaptive thresholding, which enhance contrast for better character segmentation. Additionally, **morphological operations** help in refining character shapes, removing distortions, and improving text visibility. **Edge detection algorithms** such as Canny and Sobel filters are also used to identify text boundaries, facilitating more accurate text localization.

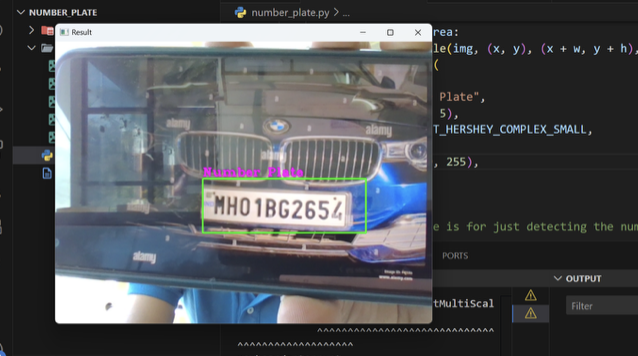
Beyond traditional OCR methods, **deep learning-based approaches** have revolutionized text recognition. The **EAST (Efficient and Accurate Scene Text Detection) model** enables real-time text detection, even for text at multiple angles and orientations. Furthermore, **Convolutional Recurrent Neural Networks (CRNNs)** provide improved recognition of complex and handwritten text, overcoming challenges in cursive and distorted writing styles. Tesseract OCR, particularly in its **4.0+ versions**, integrates **Long Short-Term Memory (LSTM)** networks, significantly improving the recognition of printed and handwritten documents. These advancements enhance OCR’s ability to process diverse scripts and languages with greater accuracy.

Moreover, the integration of OpenCV with deep learning facilitates **real-time OCR applications** on low-power devices, making it viable for edge computing. The development of lightweight OCR models enables efficient **on-device text recognition** in smartphones, augmented reality (AR) applications, and embedded systems, reducing the need for cloud processing and ensuring faster results. **Future advancements in OCR** are expected to further improve multilingual support and handwritten text recognition. Additionally, **AI-powered denoising techniques** will enhance OCR performance for low-resolution or degraded text images. With the increasing demand for automation, OCR is set to play a pivotal role in **real-time applications** such as **autonomous vehicles, smart surveillance systems, and automated document processing**. These innovations will continue to push the boundaries of OCR technology, making it more robust, adaptive, and accessible across various industries.

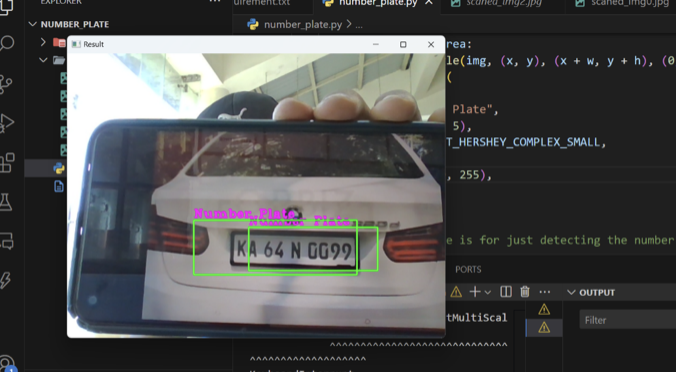
**Result and Screenshot**

**Image Acquisition**



The system successfully detects the license plate from a live camera feed and highlights it with a bounding box. The detected plate text is extracted using Optical Character Recognition (OCR), with results displayed on-screen for further processing.



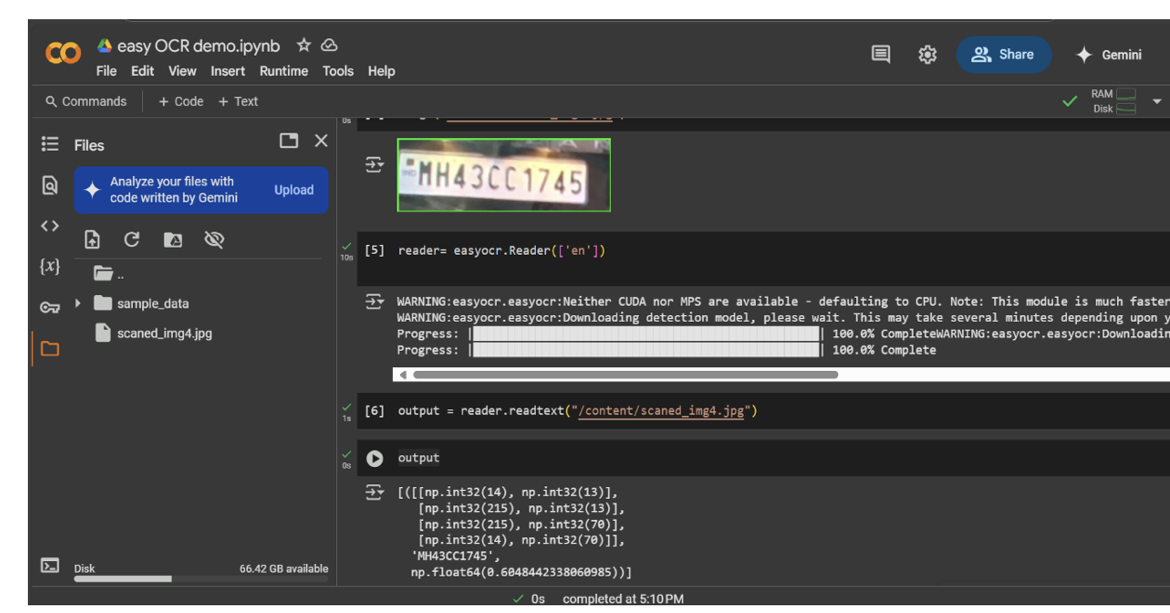
Live camera feed successfully detects license plate, highlights it with a bounding box, and extracts text via OCR for on-screen display

**Image Extraction using EasyOCR**

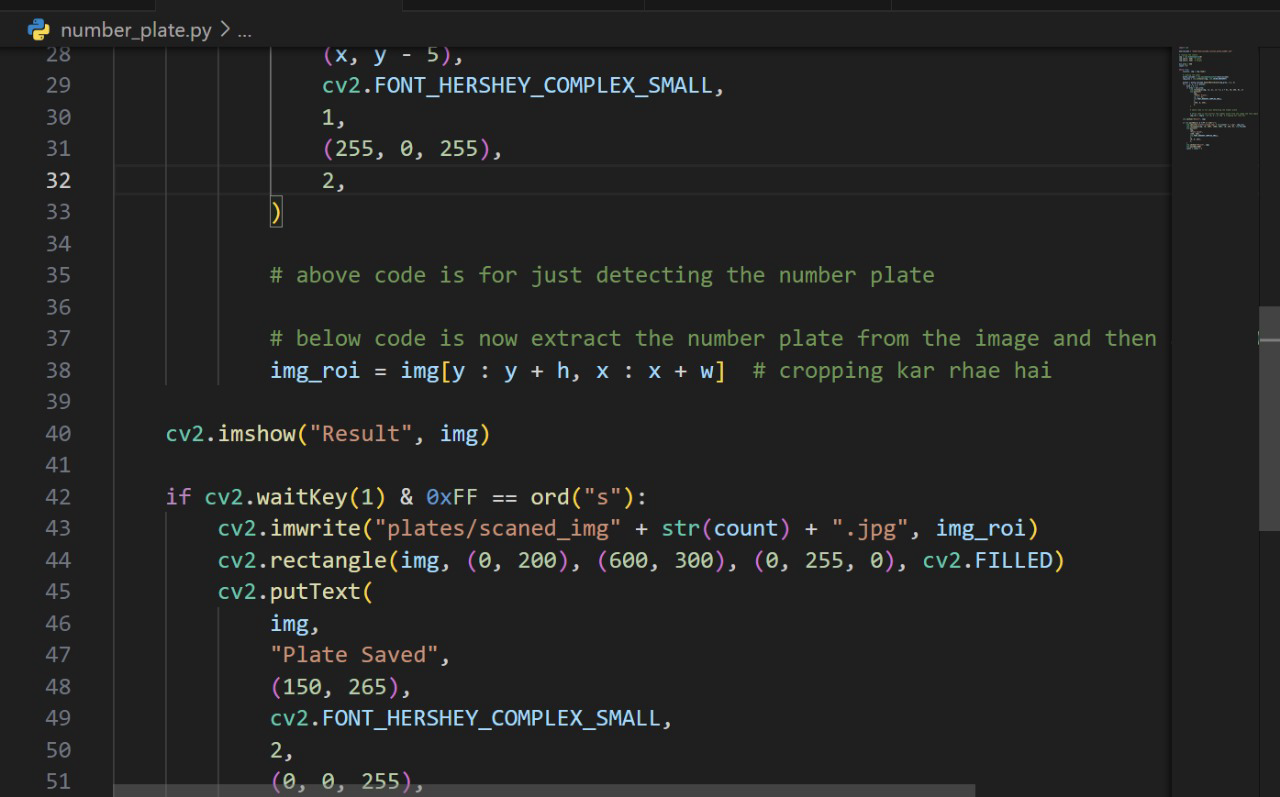


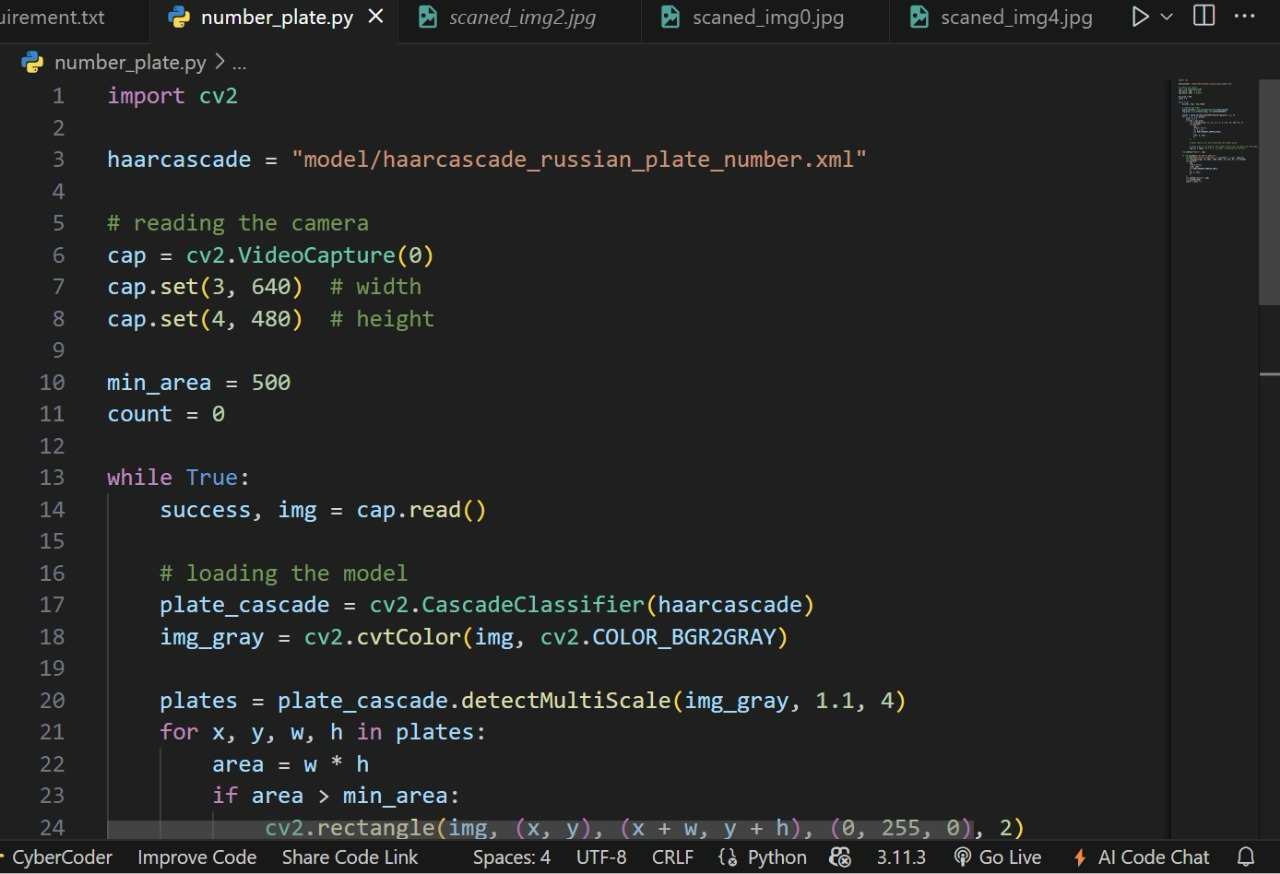
Extracting the number plate from the car

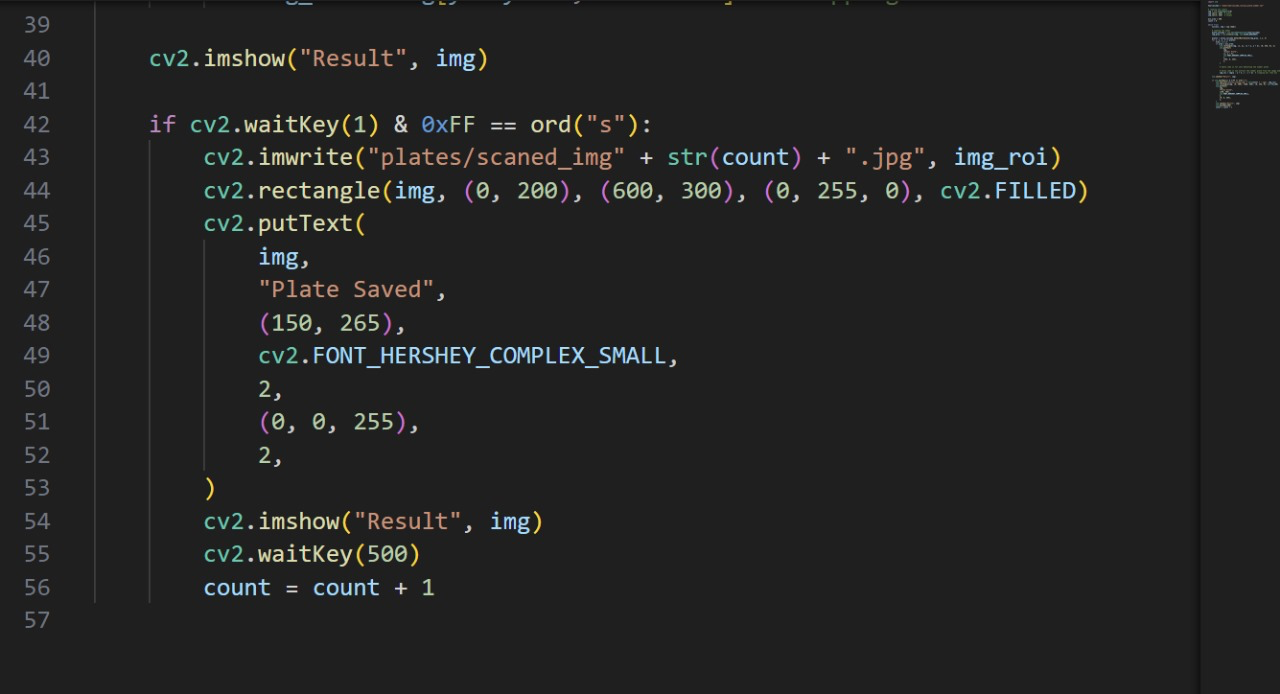
**Model Training**

****

This screenshot shows the process of capturing a number plate image and extracting its text using EasyOCR. The detected plate, MH43CC1745, is highlighted, and the corresponding text output is displayed below.

**Code screenshot: **

****

****

This code sets up a real-time system to detect Russian number plates from a webcam feed. It loads a pre-trained Haar cascade classifier, captures frames from the camera, converts them to grayscale, detects potential number plates, and draws bounding boxes around the detected regions. It's a basic example of object detection using OpenCV and Haar cascades, often used as a starting point for more complex license plate recognition systems.

This code snippet forms a crucial part of a real-time number plate detection and saving system, likely integrated within a larger application using OpenCV. It continuously captures frames from a video stream or camera feed and displays them in a window titled "Result," showcasing the ongoing detection process. When the user presses the "s" key, the system triggers the saving mechanism. It extracts the detected number plate region, denoted as img\_roi, and stores it as a JPEG image with a unique filename, dynamically generated using a counter. To visually confirm the successful save, a filled green rectangle and the text "Plate Saved" are overlaid on the displayed frame. A brief pause is introduced to ensure the user can clearly see the confirmation message. Finally, a counter is incremented, preparing the system for the next potential number plate save. This functionality allows for efficient capture and storage of detected number plates, which can be further processed or analyzed for various applications, such as license plate recognition or traffic monitoring.