Cognifront

Project Report

Date: 20/01/2024

Declaration

Student name : Saiesh Agre

Project Title : License Plate Detection

Internal Guide Name : Prof. Namrata D. Ghuse

External Guide Name (Cognifront) : Mr. Suchit Tiwari

College Name : K. K. Wagh Institute of Engineering Education and Research

Branch and Year : Artificial Intelligence & Data Science (B) - T.E.

Roll Number : 01

Date of Submission : 20/01/2024

Signature :

Note by the Author:

I'm thrilled to provide this project report, which is the result of extensive research and hard work. This project has been a journey full of obstacles, discoveries, and valuable lessons from the start to the finish. There were anticipated and unforeseen challenges during the implementation phase, but each presented an opportunity for growth and problem-solving. Presenting the results brings me great pleasure, as it has been a fulfilling experience.

Warm regards,

Saiesh Agre

Certificate

This is to certify that the internship project report entitled **License Plate Detection** submitted by **Saiesh Agre** in partial fulfillment of the requirement for Internship completion for Third Year Engineering (T.E.) for the Academic year 2023 - 2024 as prescribed in the **Savitribai Phule Pune University (SPPU)** curriculum

Place: Nashik

Date:

Project Guide (Internal)	Project Guide (External) Cognifront	Head of Department
(Name)	(Name)	(Name)
(Sign)	(Sign)	(Sign)

Acknowledgements

I wish to express my sincere gratitude to **Mr. Suchit Tiwari** of **Cognifront** for his unwavering support, guidance, and mentorship throughout the entire duration of this project. His profound expertise and invaluable insights significantly contributed to the successful completion of this endeavor.

I extend my deep appreciation to **Prof. Namrata D. Ghuse**, my esteemed internal guide, for their indispensable advice, constructive feedback, and continuous encouragement. Their mentorship has been pivotal in providing the necessary direction, shaping the project into its ultimate form.

Additionally, I am grateful for the steadfast support from my family and friends. Their unwavering encouragement and understanding were instrumental in overcoming challenges and maintaining unwavering focus.

The accomplishment of this project owes much to the collaborative efforts of all individuals mentioned above. I extend my heartfelt thanks for being integral contributors to this journey of achievement.

Abstract

This project revolves around the development of a sophisticated license plate detection and character recognition system, leveraging the synergistic capabilities of OpenCV and EasyOCR. The primary objective is to create an adaptable solution capable of accurately identifying license plates within images. Utilizing OpenCV for robust image processing, the system systematically detects license plates, addressing challenges like varying lighting conditions and plate sizes. Subsequently, EasyOCR is employed for character recognition, extracting alphanumeric information. The collaborative integration of these libraries aims to yield a precise and versatile automated license plate recognition system, with potential applications in vehicle monitoring, security, and broader automation contexts.

Table of Contents

Introduction to the Project

Within the scope of this project, my focus was on the development of a comprehensive license plate recognition system, encompassing three distinct yet interconnected tasks. The initial task centered on the adept utilization of OpenCV, a powerful computer vision library, to achieve efficient and accurate number plate extraction from a diverse array of images. This involved overcoming challenges posed by varying lighting conditions and plate sizes, ensuring the adaptability of the system.

Subsequently, the project transitioned into the realm of character recognition, employing the capabilities of EasyOCR. The objective here was to precisely decipher alphanumeric information embedded within the extracted license plates. This task aimed at enhancing the system's accuracy and efficiency, making it adept at handling a myriad of characters and fonts.

The final task introduced a novel approach to character segmentation by implementing a vertical split technique. This step further refined the system's ability to interpret individual characters, contributing to its overall precision. By breaking down the characters in a license plate, this approach aimed to optimize the recognition process, particularly when dealing with complex or stylized fonts.

Project Specification

1. Objective

Develop a robust license plate recognition system using OpenCV and EasyOCR. Specific goals include efficient number plate **extraction**, accurate character **recognition**, innovative character **segmentation** through vertical splitting, system adaptability, and integration for practical applications in vehicle monitoring, security, and automation.

2. Tasks

- **Data Collection**
  Collect diverse set of images to test the algorithm

- **Image Preprocessing**
  Implement basic image preprocessing techniques using OpenCV to enhance clarity and correct lighting variations.

- **Number Plate Extraction**
  Use contour detection technique in OpenCV to locate and extract potential license plates from images.

- **Character Recognition Integration**
  Integrate EasyOCR to recognize characters on the extracted license plates and output the alphanumeric information.

- **Character Segmentation**
  Implement a straightforward vertical split technique for character segmentation to enhance recognition accuracy.

- **Testing**
  Conduct initial testing using a small dataset to evaluate the system's basic functionality.

3. Technology Stack

- **Programming Language**
  Python: Widely used for its extensive libraries and frameworks in computer vision and image processing.

- **Computer Vision Library:**
  OpenCV: Essential for image processing tasks, including license plate extraction and manipulation.

- **Optical Character Recognition (OCR) Library:**
  EasyOCR: A powerful OCR library capable of recognizing text in images, suitable for extracting alphanumeric characters from license plates.

- **Numerical Computing Library:**
  Numpy: Utilized for character segmentation.

4. Deliverables

- Fully functional License Plate Recognition System
- Python source code
- Presentation
- Detailed final project report

Implementation Details

1. Image Preprocessing

1.1. Resizing the image

Utilized the *cv2.resize()* function to resize images, to standardize input.

1.2. Grayscaling

Applied grayscaling to simplify image processing and reduce computational complexities.

1.3. Thresholding

Employed thresholding technique to separate *foreground* (white region) and *background* (black region) aiding in accurate edge detection.

1.4. Edge Detection

Incorporated edge detection to enhance contour detection accuracy by pinpointing significant changes in pixel intensity values.

2. License Plate Extraction

Utilized *contour detection algorithm* on the preprocessed image to identify license plate region. The detected region of interest is *masked*, *extracted* and *saved* as a separate image file for further task of Optical Character Recognition.

3. Character Recognition

Integrated EasyOCR library to accurately identify and interpret text within the saved image file.

Lessons Learned

1. **Library Integration:** Incorporating the OpenCV library for detection and extraction of regions of interest and EasyOCR library for streamlined character recognition.

2. **Preprocessing Significance:** The importance of robust preprocessing, including grayscaling, thresholding, and contour detection, became evident.

3. **Parameter Tuning Iterations:** Iterative tuning of parameters for each processing step proved essential. Adjustments in preprocessing and EasyOCR and contour detection settings played a key role in achieving optimal performance.

4. **Real-world Variability:** Acknowledging and addressing variations in real-world scenarios, such as diverse lighting conditions or plate types, is crucial for creating a robust and adaptable system.

5. **Continuous Testing and Validation:** Regularly testing the system with diverse images and scenarios is essential. Continuous validation helps identify potential issues and ensures the reliability of the license plate recognition system.

Learning Outcomes

1. **OpenCV2 & EasyOCR Proficiency:**
   Attained proficiency in utilizing OpenCV2 for image processing tasks, and EasyOCR for character recognition.

2. **Understanding Image Variability:**
   Developed an understanding of handling real-world image variability, acknowledging factors like diverse lighting conditions and plate types.

3. **Image Preprocessing Skills:**
   Developed expertise in image preprocessing techniques, including grayscaling, thresholding, and edge detection.

4. **License Plate Extraction Competence:**
   Demonstrated competence in using contour information to accurately extract license plate regions, showcasing an understanding of how to leverage contours for effective object boundary delineation.

5. **Optimization Techniques:**
   Explored and applied optimization techniques within OpenCV2, refining the preprocessing steps and contour extraction to enhance efficiency and accuracy in license plate region identification.

Conclusion

In concluding my internship at **Cognifront**, I extend my heartfelt gratitude to **Mr. Suchit Tiwari**, whose unwavering support and insightful guidance significantly enriched my learning experience. His expertise has been instrumental in navigating the intricacies of the project.

I am equally thankful to my internal guide, **Prof. Namrata D. Ghuse**, for her scholarly guidance and continuous encouragement throughout this project. Prof. Ghuse's mentorship provided a strong academic foundation, contributing to the successful execution of the internship.

The project aimed to leverage OpenCV2 and EasyOCR for the extraction of license plate regions from images. The integration of these technologies not only showcased technical proficiency but also emphasized the importance of adaptability in handling diverse image variations. The systematic application of image preprocessing, thresholding, edge detection, and contour extraction demonstrated a comprehensive approach to solving real-world challenges in license plate recognition.

By successfully implementing character recognition within the extracted regions, the project has not only enhanced my technical skills but also highlighted the potential applications of such systems in fields ranging from security to smart transportation.

This internship has been a transformative journey, providing practical insights into the dynamic realm of computer vision and OCR technologies. As I express my gratitude to Mr. Suchit Tiwari and Prof. Namrata D. Ghuse, I am eager to apply the knowledge gained from this experience to future endeavors, contributing to the ever-evolving landscape of technology.

Bibliography

1. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html
2. https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html
3. https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
4. https://www.geeksforgeeks.org/image-thresholding-in-python-opencv/
5. https://blog.roboflow.com/license-plate-detection-and-ocr/
6. https://universe.roboflow.com/browse/transportation/anpr
7. https://circuitdigest.com/microcontroller-projects/license-plate-recognition-using-raspberry-pi-and-opencv
8. https://github.com/nicknochnack/ANPRwithPython
9. https://github.com/JaidedAI/EasyOCR

Appendices

Appendix A: Terminologies

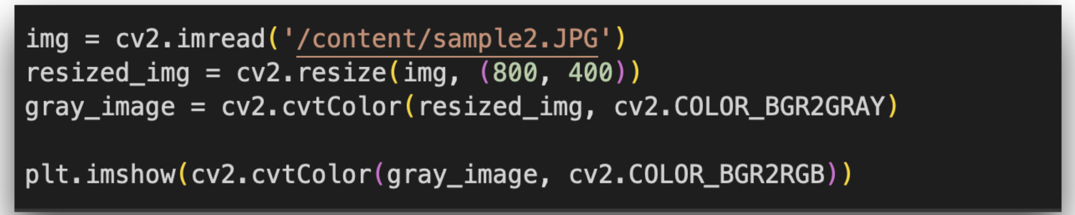
A.1. Contours: Contours in image processing refer to continuous curves that outline the boundaries of objects within an image. Algorithms, like those provided by OpenCV, are used to detect contours. These algorithms identify changes in intensity or color, revealing the boundaries of the objects.

A.2. Thresholding: Thresholding is a fundamental technique in image processing that simplifies visual data by dividing an image into two categories: *foreground* and *background*. It involves setting a specific intensity value (the threshold) to classify pixels. Pixels with intensities above the threshold are considered part of the foreground, while those below become part of the background.

A.3. Optical Character Recognition (OCR) : It is a technology that converts different types of documents, such as scanned paper documents, PDFs, or images captured by a digital camera, into editable and searchable data. It recognizes text characters within these images and translates them into machine-readable text.

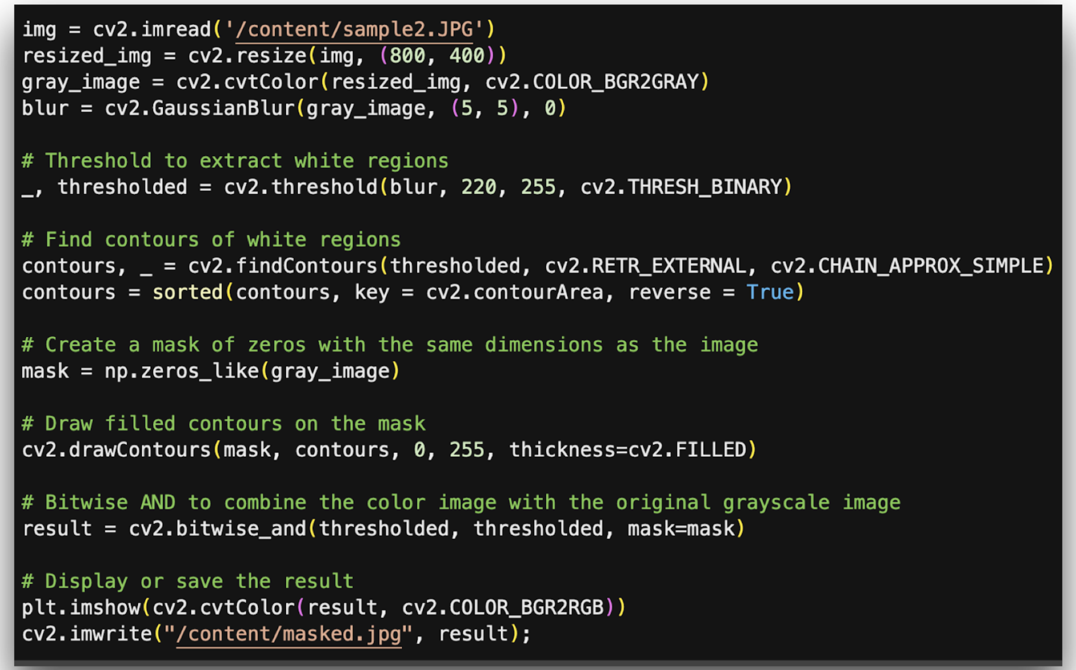
Appendix B: Code Snippets

B.1. Image Preprocessing



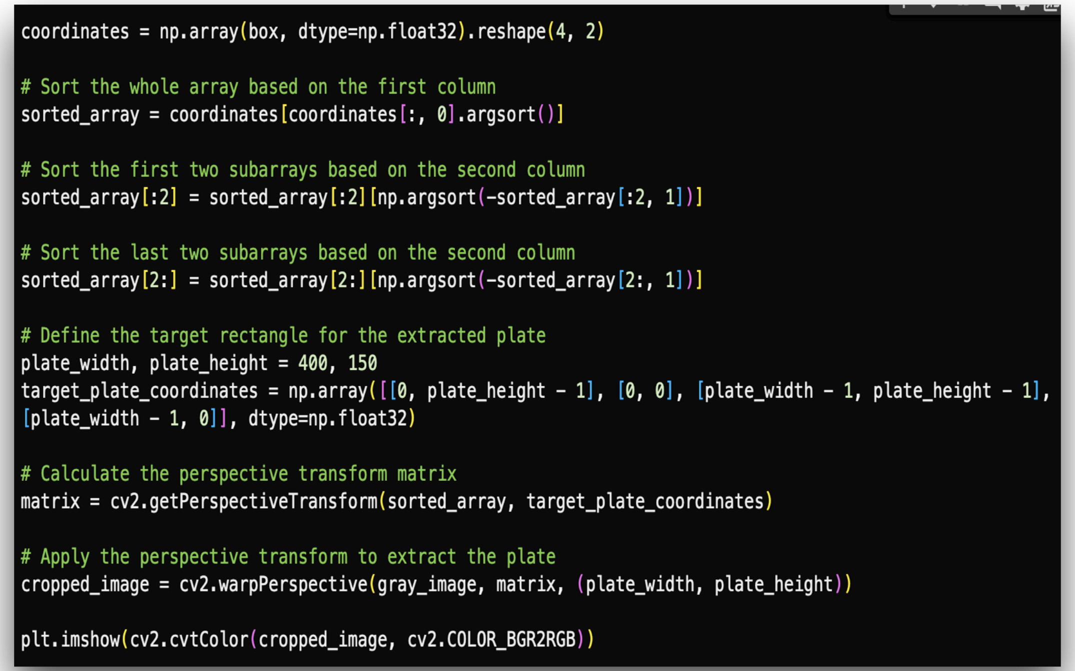
```
img = cv2.imread('/content/sample2.JPG')
resized_img = cv2.resize(img, (800, 400))
gray_image = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)

plt.imshow(cv2.cvtColor(gray_image, cv2.COLOR_BGR2RGB))
```

B.2. Thresholding



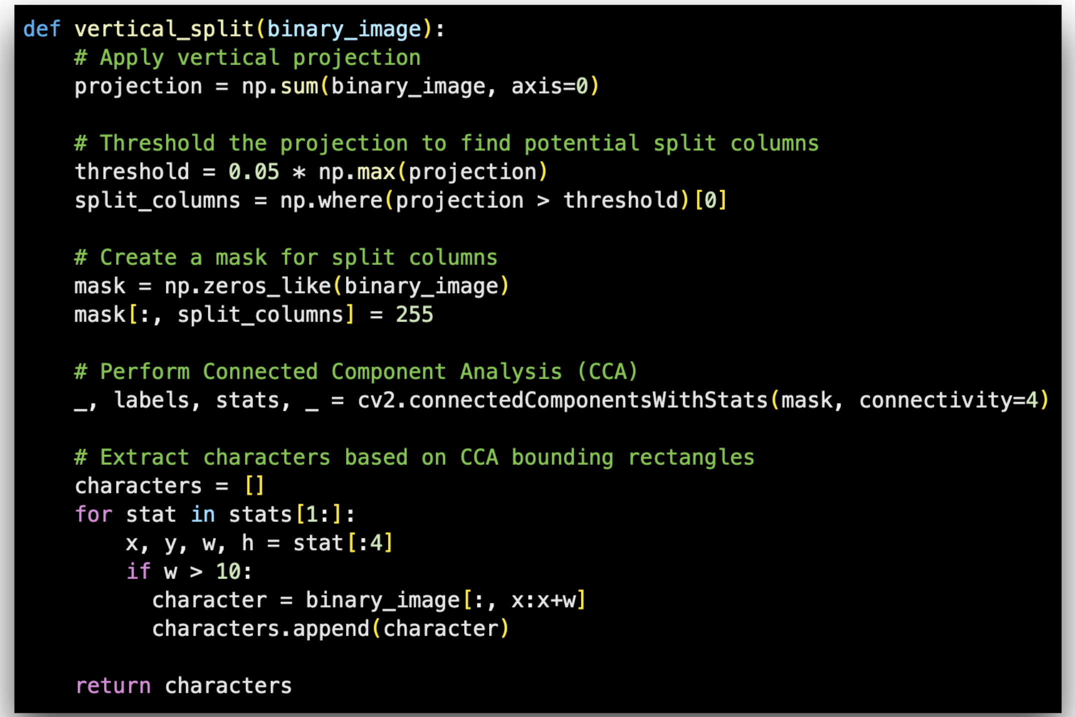
```python
img = cv2.imread('/content/sample2.JPG')
resized_img = cv2.resize(img, (800, 400))
gray_image = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray_image, (5, 5), 0)

# Threshold to extract white regions
_, thresholded = cv2.threshold(blur, 220, 255, cv2.THRESH_BINARY)

# Find contours of white regions
contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key = cv2.contourArea, reverse = True)

# Create a mask of zeros with the same dimensions as the image
mask = np.zeros_like(gray_image)

# Draw filled contours on the mask
cv2.drawContours(mask, contours, 0, 255, thickness=cv2.FILLED)

# Bitwise AND to combine the color image with the original grayscale image
result = cv2.bitwise_and(thresholded, thresholded, mask=mask)

# Display or save the result
plt.imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
cv2.imwrite("/content/masked.jpg", result);
```

B.3. Resizing Extracted Plate



```python
coordinates = np.array(box, dtype=np.float32).reshape(4, 2)

# Sort the whole array based on the first column
sorted_array = coordinates[coordinates[:, 0].argsort()]

# Sort the first two subarrays based on the second column
sorted_array[:2] = sorted_array[:2][np.argsort(-sorted_array[:2, 1])]

# Sort the last two subarrays based on the second column
sorted_array[2:] = sorted_array[2:][np.argsort(-sorted_array[2:, 1])]

# Define the target rectangle for the extracted plate
plate_width, plate_height = 400, 150
target_plate_coordinates = np.array([[0, plate_height - 1], [0, 0], [plate_width - 1, plate_height - 1],
[plate_width - 1, 0]], dtype=np.float32)

# Calculate the perspective transform matrix
matrix = cv2.getPerspectiveTransform(sorted_array, target_plate_coordinates)

# Apply the perspective transform to extract the plate
cropped_image = cv2.warpPerspective(gray_image, matrix, (plate_width, plate_height))

plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
```

B.4. Vertical Split for character segmentation



```
def vertical_split(binary_image):
    # Apply vertical projection
    projection = np.sum(binary_image, axis=0)

    # Threshold the projection to find potential split columns
    threshold = 0.05 * np.max(projection)
    split_columns = np.where(projection > threshold)[0]

    # Create a mask for split columns
    mask = np.zeros_like(binary_image)
    mask[:, split_columns] = 255

    # Perform Connected Component Analysis (CCA)
    _, labels, stats, _ = cv2.connectedComponentsWithStats(mask, connectivity=4)

    # Extract characters based on CCA bounding rectangles
    characters = []
    for stat in stats[1:]:
        x, y, w, h = stat[:4]
        if w > 10:
          character = binary_image[:, x:x+w]
          characters.append(character)

    return characters
```