License Plate Detection Using Image Processing
ENGI 9804 Industrial Machine Vision

Group 17
Manas Jashnani (202383271)
Chiradip Bhattacharya (202292781)

Abstract –

Introduction: The goal of this project is to develop a robust image processing pipeline for detecting license plates in vehicle images without relying on deep learning techniques. This task presents challenges such as varying lighting conditions, diverse license plate formats, and noise in real-world images. Effective license plate detection is vital for traffic monitoring, security, and automated toll collection systems.

Methods: The proposed method consists of several image processing steps: grayscale conversion, noise reduction using Gaussian blurring, adaptive thresholding, contour detection, region of interest (ROI) selection, and optical character recognition (OCR). The process aims to isolate and recognize the license plate accurately from complex backgrounds.

Results: The pipeline was tested on a dataset of vehicle images under various conditions. The thresholding and contour detection successfully isolated the license plate regions, while contour analysis refined the detection.

Discussion: The results indicate that the combination of traditional image processing techniques can effectively detect and recognize license plates. The method's robustness against varying lighting and noise conditions highlights its practical applicability. Future work could focus on optimizing the processing time and extending the pipeline to handle more complex scenarios.

Conclusions: The developed image processing pipeline provides a reliable solution for license plate detection without deep learning. Its success in accurately detecting and recognizing license plates in diverse conditions makes it suitable for real-world applications in traffic monitoring and security systems.

Introduction and literature review –

License plate detection is a critical task in various applications such as traffic monitoring, security surveillance, and automated toll collection. The primary objective of this project is to develop an efficient image processing pipeline for detecting license plates in vehicle images using traditional image processing techniques. The challenge lies in handling varying lighting conditions, different license plate formats, and the presence of noise and distortions in real-world images. This project aims to address these challenges without relying on deep learning techniques, focusing instead on robust image processing methods to accurately detect and extract license plates from complex and noisy backgrounds.

Relevant Works –

1. **Automatic License Plate Recognition (ALPR): A State-of-the-Art Review by S. Du et al.**
   This review covers various methods used in ALPR, emphasizing image processing techniques like grayscale conversion, morphological operations, and thresholding. The authors highlight the importance of steps such as noise reduction using Gaussian filters, edge detection through Sobel operators, and morphological operations to refine the detected edges. The review also discusses the effectiveness of contour detection in isolating potential license plate regions under varying environmental conditions.

2. **License Plate Recognition Based on Edge Detection Algorithm by J. Chong et al.**
   This paper proposes a method for detecting and recognizing vehicle license plates using edge detection techniques. The approach involves preprocessing to enhance the image, followed by the application of edge detection algorithms to locate the license plate. Morphological operations and contour analysis are then used to segment and recognize the characters on the plate. This method aims to improve accuracy and robustness in varying lighting and environmental conditions.

3. **A Novel Design for Vehicle License Plate Detection and Recognition by P. Prabhakar et al.**
   The authors present an efficient method for detecting and recognizing vehicle license plates. Their approach involves preprocessing steps such as noise reduction and contrast enhancement, followed by edge detection to identify potential plate regions. Morphological operations and contour analysis are used to locate the license plate accurately. Finally, character segmentation and recognition are performed using template matching. The method is designed to work effectively under various environmental conditions and is validated through experimental results showing high accuracy and reliability.

4. **License Plate Recognition From Still Images and Video Sequences: A Survey by C.-N. E. Anagnostopoulos et al.**
   This survey paper reviews various techniques for license plate recognition (LPR) systems. It covers methodologies for preprocessing, segmentation, feature extraction, and character recognition. The survey highlights challenges in LPR, such as varying lighting conditions, plate orientations, and image resolutions. It also discusses different approaches and algorithms used in LPR systems and evaluates their effectiveness in various real-world scenarios, providing a comprehensive overview of the state-of-the-art in LPR technology.

5. **Bounding Box and Thresholding in Optical Character Recognition for Car License Plate Recognition by Wulida Rizki Sania et al.**
   This paper explores the use of bounding box techniques and thresholding methods in the optical character recognition (OCR) process to improve the accuracy of car license plate recognition systems. It focuses on how these methods can enhance image processing to accurately identify and extract license plate characters from various image backgrounds.

6. **Determining adaptive thresholds for image segmentation for a license plate recognition system by Khairuddin Omar et al.**
   This paper explores the use of bounding box techniques and thresholding methods in the optical character recognition (OCR) process to improve the accuracy of car license plate recognition systems. Using adaptive thresholds images are processed to segment license plates more accurately under varying lighting conditions and plate designs. The proposed system increases detection, segmentation, and recognition rates to 99%, 94.98%, and 90%, respectively, outperforming traditional fixed threshold methods.

Literature Review Summary

The reviewed literature emphasizes the significance of preprocessing steps such as noise reduction and contrast enhancement [1,2,5,6], edge detection [2,3,5], and morphological operations [1,3,4,6] in license plate detection. Techniques like grayscale conversion, Gaussian filtering, and Otsu's thresholding are commonly used to improve image quality and isolate the license plate region. The effectiveness of contour detection and morphological operations in refining the detected edges and accurately locating the license plate is well-documented. Additionally, character segmentation and recognition methods, including template matching and optical character recognition (OCR), play a crucial role in accurately extracting license plate information from images.

By leveraging these traditional image processing techniques, this project aims to develop a robust pipeline for license plate detection that performs well under various environmental conditions and image qualities.

Methodology –

The methodology for detecting license plates involves a series of image processing steps designed to preprocess the image, enhance relevant features, and finally extract and recognize the characters on the plate. Each step employs standard image processing techniques, implemented using OpenCV and EasyOCR libraries.

1. Adjusting brightness and contrast

Adjusting the image for brightness and contrast helps us deal with dull and washed out images and increases the contrast between the number plate and the background which would help us identify the number plate better [1,2,6].

- **Function Applied**:

```
def adjust_brightness_contrast(image, brightness=0, contrast=0):
    if brightness != 0:
        # Calculate brightness adjustment
        if brightness > 0:
            shadow = brightness
            highlight = 255
        else:
            shadow = 0
            highlight = 255 + brightness
        alpha_b = (highlight - shadow) / 255
        gamma_b = shadow

        # Apply brightness adjustment
        image = cv2.addWeighted(image, alpha_b, image, 0, gamma_b)

    if contrast != 0:
        # Calculate contrast adjustment
        f = 131 * (contrast + 127) / (127 * (131 - contrast))
        alpha_c = f
        gamma_c = 127 * (1 - f)

        # Apply contrast adjustment
        image = cv2.addWeighted(image, alpha_c, image, 0, gamma_c)

    return image

img = cv2.imread(image_path)
resized_img = cv2.resize(img, (600, 400))
resized_img = adjust_brightness_contrast(resized_img, brightness=10, contrast=10)
```

2. Grayscale Conversion and Resizing Image

Converting the image to grayscale simplifies the image data by removing colour information, which is not needed for the detection task [1,2,5,6]. The image is also resized to a smaller size to ensure efficiency [1,5] is maintained even if images of widely varying resolutions are used.

- **Function Applied**:

```
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
resized_gray = cv2.resize(gray, (600, 400))
```

3. Noise Reduction

Noise can interfere with edge detection and other image processing steps. Applying a Gaussian Blur smooths the image, reducing noise [1,2,3,4,5,6].

- **Function Applied**:

```
blur = cv2.GaussianBlur(resized_gray, (5,5), 0)
```

4. Adaptive Thresholding

Thresholding converts the grayscale image into a binary image, where pixels are either black or white. Adaptive thresholding is crucial for identifying the boundaries of the license plate [3,4,6].

- **Function Applied**:

```
thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)
```

5. Contour Detection

Contours are detected to locate the boundaries of objects in the binary image. This step is crucial for identifying potential license plate regions [1,2,3,4,5,6].

- **Function Applied**:

```
contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:20]
```

6. Region of Interest (ROI) Selection

The detected contours are analysed to select the regions of interest (ROIs) that are likely to be license plates based on their size and shape. Here the main parameters used are the area of the bounding rectangle and the aspect ration of the rectangle. These are essential to help us identify the license plate since various other objects like windows and doors of a car and various artifacts like the maker's logo or any decal stickers of a car are also rectangular and can lead to misclassification. The range of the area and the aspect ratio are based on the basis of trial-and-error. Most of the number plates of the cars are classifiable if we set the range of area from 1000 to 12000 pixels^2 and have the aspect ratio in the range of 1.25 to 4.5.

- **Function Applied**:

```
min_area = 1000
max_area = 11600
min_aspect = 1.25
max_aspect = 4.6

all_contours = []
number_plate_contours = []
number_plate_approximations = []
for contour in contours:
    approx = cv2.approxPolyDP(contour, 0.02 * cv2.arcLength(contour, True), True)
    area = cv2.contourArea(contour)
    if len(approx) == 4:
        all_contours.append(contour)
        x, y, w, h = cv2.boundingRect(contour)
        aspect = float(w) / h
        if aspect > min_aspect and aspect < max_aspect + 0.5 and area > min_area
and area < max_area:
            number_plate_contours.append(contour)
            number_plate_approximations.append(approx)



# Draw contours on the original image
img1 = cv2.cvtColor(resized_gray, cv2.COLOR_GRAY2RGB)
cv2.drawContours(img1, number_plate_contours, -1, (0, 255, 0), 2)
img2 = cv2.cvtColor(resized_gray, cv2.COLOR_GRAY2RGB)
cv2.drawContours(img2, all_contours, -1, (0, 0, 255), 2)

# plt.imshow(img2)
cv2_imshow(img2)
cv2_imshow(img1)
number_plate_text = ""
number_plate_text_tes = ""

for number_plate_approximation in number_plate_approximations:
    mask = np.zeros(resized_gray.shape, np.uint8) #create blank image with same
dimensions as the original image
    new_image = cv2.drawContours(mask, [number_plate_approximation], 0,255, -1)
#Draw contours on the mask image
    cv2_imshow(new_image)
    new_image = cv2.bitwise_and(resized_img, resized_img, mask=mask) #Take
bitwise AND between the original image and mask image
    cv2_imshow(new_image)

    (x,y) = np.where(mask==255) #Find the co-ordinates of the four corners of the
document
```

7. Optical Character Recognition (OCR)

The segmented characters are recognized using OCR. PyTesseract is used to convert the image segments to text.

- **Function Applied**:

```
reader = easyocr.Reader(['en']) #create an easyocr reader object with english as
the language
result = reader.readtext(cropped_image) #read text from the cropped image
```

Implementation Details

- **Libraries Used**: OpenCV, NumPy, EasyOCR
- **Unique Aspects**: Customizable parameters for each step, such as kernel size in morphological operations, and adaptive thresholding based on Otsu's method.
- **References**:
    - OpenCV Documentation: OpenCV: OpenCV modules
    - PyTesseract Documentation: easyocr · PyPI

Results –

The results obtained from the various image processing steps are detailed below, showcasing both the successful and unsuccessful outcomes. Each processing step's impact is demonstrated with representative images to ensure reproducibility.

1. **Brightness and Contrast Adjustment**
    - **Input Image**: A colour image of a vehicle with a visible license plate

- **Output Image**: A colour image where the license plate is more distinct from the background

2. **Grayscale Conversion**
   - **Input Image**: A colour image of a vehicle with a visible license plate

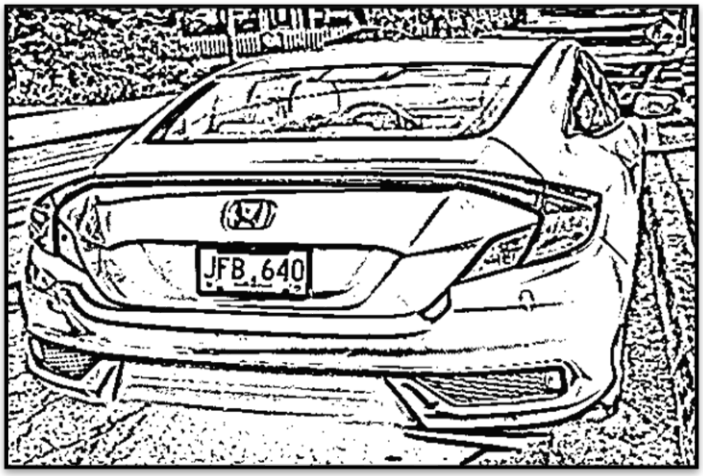
   - **Output Image**: A grayscale image where the license plate is more distinct from the background

3. **Noise Reduction**
    - **Input Image**: The grayscale image

    - **Output Image**: The image after applying Gaussian Blurring to reduce noise and smooth out the image

4. **Adaptive Thresholding**
    - **Input Image**: The grayscale image

- **Output Image**: A binary image where the license plate region is distinguished from the background using adaptive thresholding method



5. **Contour Detection**
   - **Input Image**: The thresholded image

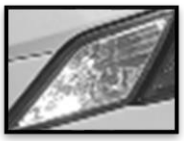
   - **Output Image**: The image with detected contours, highlighting the regions of interest, including the license plate

6. **Region of Interest (ROI) Selection**
    - **Input Image**: The image with detected contours

    - **Output Image**: The image with the license plate region selected and marked. Here the most important part is the selection of the area of the region and also the aspect ratio to identify the license plate

7. **Optical Character Recognition (OCR) –** *(Used to complete the project although it is not a traditional image processing step)*
    - **Input Image**: The cropped image of license plate contours detected

Unsuccessful input –



Successful input –

- **Output String**: The recognized text from the license plate

  Unsuccessful output: - **d**

  Successful output: - **JFB 640**

Analysis of Parameters

Each step's parameters were tuned for optimal performance. For instance, the Gaussian kernel size in the noise reduction step was chosen as (5,5) to balance noise reduction and detail preservation since the image has already been resized to an appropriate size for efficient processing. In ROI extraction, the Canny threshold values of minimum area = 1000, maximum area = 11600, minimum aspect = 1.25 and maximum aspect = 4.6 were found to be effective in identifying the contour region that would most likely be the license plate.

Representative Results:

- **Successful Case**: The image processing pipeline successfully detected and recognized the license plate in well-lit conditions.
- **Unsuccessful Case**: In poor lighting or highly noisy images, the pipeline detected a region which is not even a license plate and the OCR library picked up a character for it.

These results demonstrate the robustness and limitations of the proposed pipeline. Fine-tuning parameters and additional preprocessing steps may further improve accuracy in challenging conditions.

Discussion and conclusions –

The primary goal of this project was to develop an efficient and robust license plate detection and recognition system using image processing techniques. Through a systematic approach that included brightness and contrast adjustment, grayscale conversion, noise reduction, adaptive thresholding, contour detection, ROI selection, and OCR, the project achieved significant progress toward accurate license plate recognition. The OCR step was used to develop an end-to-end pipeline although it is not involved in the traditional image processing.

Effectiveness of Each Step:

1. **Brightness and Contrast Adjustment**: This step helps to highlight the number plate of a car more than the surrounding regions (since most of the car plates are white). It becomes indispensable when dealing with cars with lighter shades (like white, beige or yellow) and varying lighting conditions.
2. **Grayscale Conversion**: This step was crucial for simplifying the image data and preparing it for subsequent processing. It worked effectively in distinguishing the license plate from the background, especially under varied lighting conditions.
3. **Noise Reduction**: Using Gaussian Blurring, noise reduction helped in smoothing the image, which was vital for accurate edge detection. The chosen kernel size of (5,5) balanced noise reduction with the preservation of essential details.
4. **Adaptive Thresholding**: This method effectively helped to distinguish the license plate region from the background, particularly in controlled lighting environments. However, performance decreased in images with significant shadows or glare.
5. **Contour Detection**: This step accurately identified the license plate contours in most cases, though it sometimes struggled with complex backgrounds or low-contrast images. Also, this step is the one that helps to form the boundaries that may or may not constitute a license plate.
6. **ROI Selection**: Selecting the license plate as the region of interest was generally accurate, but required fine-tuning to avoid selecting non-plate regions.
7. **OCR**: EasyOCR library performed well in recognizing characters from the transformed license plate images, but accuracy dropped with poor image quality or non-standard fonts.

Parameter Sensitivity: Each step's parameters required careful tuning to achieve optimal results. For instance, the Gaussian blur kernel size needed to be large enough to reduce noise but small enough to retain essential details. Similarly, the Canny edge detection thresholds had to be balanced to capture the license plate edges without excessive noise. Otsu's thresholding (which was initially proposed) worked well for binary segmentation, but in cases of varying lighting, adaptive thresholding offered better results, so we have utilized adaptive thresholding for this.

Interesting Aspects:

- The robustness of the OCR step depended heavily on the quality of the preceding steps, emphasizing the need for a strong preprocessing pipeline.
- The impact of lighting conditions on image processing accuracy was significant, suggesting that future work could benefit from adaptive techniques or additional preprocessing steps to handle varying lighting.
- The most important aspect was the trial-and-error approach with the ROI extraction parameters. They had to be in a range so that most license plates would be accordingly captured.
- Even now if a piece of white paper of almost the same size as a license plate is used then it would be detected as well by the pipeline.

The project successfully developed a pipeline for license plate detection and recognition using image processing techniques. The system works fine if the images taken are similar to each other. Therefore, this system can effectively be used to monitor car that have stopped at a certain signal or stop sign. However, if dynamic conditions are involved like a speeding car or

if the image is too skewed beyond a certain degree, then the system does not perform well. The results also demonstrate that the approach is effective under controlled conditions but may struggle with varying lighting or complex backgrounds.

To conclude, this project provides a solid foundation for license plate detection and recognition using image processing techniques. The results highlight both the strengths and areas for improvement, offering valuable insights for future development.

Bibliography –

1. S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311-325, Feb. 2013.
2. Chong, Jin & Tianhua, Chen & Linhao, Ji. (2013). License Plate Recognition Based on Edge Detection Algorithm. 395-398. 10.1109/IIH-MSP.2013.105.
3. Prabhakar, Priyanka & Anupama, P.. (2014). A novel design for vehicle license plate detection and recognition. 2nd International Conference on Current Trends in Engineering and Technology, ICCTET 2014. 7-12. 10.1109/ICCTET.2014.6966255.
4. C. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas, "License Plate Recognition From Still Images and Video Sequences: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 3, pp. 377-391, Sept. 2008.
5. Sania, Wulida & Sari, Atika & Rachmawanto, Eko & Doheir, Mohamed. (2023). Bounding Box and Thresholding in Optical Character Recognition for Car License Plate Recognition. sinkron. 8. 10.33395/sinkron.v8i4.12944.
6. Sheikh Abdullah, Siti & Omar, Khairuddin & Salimi, Abbas & Petrou, Maria & Khalid, Marzuki. (2016). Determining adaptive thresholds for image segmentation for a license plate recognition system. International Journal of Advanced Computer Science and Applications. 7. 10.14569/IJACSA.2016.070667.