License Plate Detection and Text Extraction

Report-2

RACHEL PAUL	B221138CS
ADHITHYA E D	B220113CS
AJMAL	B220138CS

1. Proposed Methodology

The proposed methodology for detecting and extracting text from vehicle license plates is divided into five phases:

1. **Preprocessing**:
   - Convert the input image to grayscale to simplify the detection of edges.
   - Apply bilateral filtering to reduce noise while preserving the edges. This helps maintain the sharpness of key features, such as the borders of the license plate.
2. **License Plate Detection**:
   - Use the Canny edge detection algorithm to identify prominent edges within the image. License plates typically have a distinct rectangular shape with sharp edges, making them easier to detect.
   - Extract contours from the image using OpenCV's `cv2.findContours()` function. These contours represent possible objects in the image, which are then filtered to find the one that most closely approximates a rectangular shape.
3. **Bounding Box and Cropping**:
   - Once a likely license plate contour is found, a bounding box is generated. The bounding box is the smallest rectangle that can fully enclose the detected contour.
   - Crop the grayscale image to isolate the region containing the license plate, which improves the performance of the OCR step by reducing irrelevant parts of the image.
4. **Image Enhancement**:
   - Resize and sharpen the cropped license plate image to enhance the readability of the text.
   - Use bilateral filtering again to smooth the image and reduce noise, while sharpening techniques are applied to enhance edges.
   - Adjust the brightness and contrast of the image to further improve text visibility for the OCR engine.
5. **Text Extraction**:
   - Use EasyOCR to extract the text from the enhanced image. EasyOCR is particularly effective for recognizing text in real-world scenarios, such as license plates, as it is designed to handle a variety of fonts, sizes, and distortions.
   - Clean the extracted text by removing unwanted characters (e.g., spaces, newlines) to provide a clean and readable output.
6. **Displaying Results**:
   - Overlay the detected text onto the original image using OpenCV's `putText()` function.
   - Draw a bounding box around the detected license plate to visually confirm the region from which the text was extracted.

2. Tools/Techniques

The project relies on the following tools and techniques:

- **OpenCV**: Used for image preprocessing (grayscale conversion, filtering, edge detection) and contour detection. OpenCV also provides functions to draw bounding boxes and overlay text onto images.
- **EasyOCR**: A lightweight OCR tool that allows for the extraction of text from images. It works well with diverse text formats, making it ideal for real-world tasks like license plate recognition.
- **NumPy**: Used for array operations, including handling pixel data during image processing.
- **Matplotlib**: Used to visualize and display the processed images, including the annotated results.
- **imutils**: A library that provides convenience functions for basic image manipulations, such as resizing and contour detection.

3. Existing Approaches

License plate detection and recognition are common problems in computer vision. Existing approaches can be categorized as follows:

- **Traditional Methods**: These approaches typically involve image preprocessing (grayscale conversion, filtering), edge detection, and contour-based plate localization. OCR is then applied to extract the plate's characters. This method works well for simple scenarios but can struggle with complex images, such as those with low contrast or poor lighting conditions.
- **Machine Learning-Based Methods**: More advanced techniques involve deep learning models such as Convolutional Neural Networks (CNNs) for detecting license plates. Models like YOLO (You Only Look Once) or SSD (Single Shot Multibox Detector) can be trained to detect plates in real time. These approaches provide higher accuracy but require large datasets and significant computational resources.
- **Hybrid Approaches**: Some systems combine traditional and deep learning methods. For instance, traditional edge detection might be used for initial localization, while a deep learning model is applied for fine-grained detection and character recognition.

4. Design and Coding

The design of this system is centered around the sequential steps required for successful license plate detection and text extraction. The following sections break down the design components:

Image Preprocessing:

```
def preprocess_image(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    bfilter = cv2.bilateralFilter(gray, 11, 17, 17)  # Noise reduction
    edged = cv2.Canny(bfilter, 30, 200)  # Edge detection
    return edged, gray
```

License Plate Detection:

```
def detect_number_plate(edged, gray, img):
    contours = cv2.findContours(edged.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    contours = sorted(contours, key=cv2.contourArea,
reverse=True)[:10]  # Sort by area size
    bbox = find_bbox(contours)  # Locate the bounding box of the plate
    if bbox is not None:
        mask = np.zeros(gray.shape, np.uint8)
        new_image = cv2.drawContours(mask, [bbox], 0, 255, -1)
        new_image = cv2.bitwise_and(img, img, mask=mask)
        (x, y) = np.where(mask == 255)
        (x1, y1) = (np.min(x), np.min(y))
        (x2, y2) = (np.max(x), np.max(y))
        return x1, x2, y1, y2, bbox
    else:
        return None
```

Image Enhancement:

```
def enhance_image(cropped):
    cropped_resized = cv2.resize(cropped, None, fx=2, fy=2,
interpolation=cv2.INTER_CUBIC)
    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])  #
Sharpening kernel
    sharpened = cv2.filter2D(cropped_resized, -1, kernel)
    alpha = 1.5
    beta = 30
    adjusted = cv2.convertScaleAbs(sharpened, alpha=alpha, beta=beta)
    denoised = cv2.bilateralFilter(adjusted, 11, 17, 17)
    return denoised
```

Text Extraction with EasyOCR:

```
reader = easyocr.Reader(['en'])
def extract_text_from_image(cropped):
    enhanced_img = enhance_image(cropped)
    text = reader.readtext(enhanced_img)[0][-2]
    return text
```

Displaying Results:

```
def display_results(img, text, bbox):
    font = cv2.FONT_HERSHEY_SIMPLEX
    x1, y1 = bbox[0][0]
    x2, y2 = bbox[2][0]
    res = cv2.putText(img, text=text, org=(x1, y1 + 80),
fontFace=font, fontScale=1, color=(0, 255, 0), thickness=2)
    res = cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 3)
    plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
    plt.show()
```

Conclusion

This system successfully combines traditional image processing techniques with modern OCR to detect and extract license plate text. The use of EasyOCR enhances the accuracy of text recognition, while OpenCV provides efficient image preprocessing and plate detection capabilities. In future iterations, this system could be expanded to include deep learning models for more robust detection in difficult scenarios, such as nighttime images or blurry plates.