VEHICLE NUMBER PLATE DETECTION

MH 40BP 4231

OBJECTIVE

- This project focuses on detecting vehicle license plates and extracting text from them.
- The system combines traditional image processing with modern OCR techniques for accurate license plate recognition.

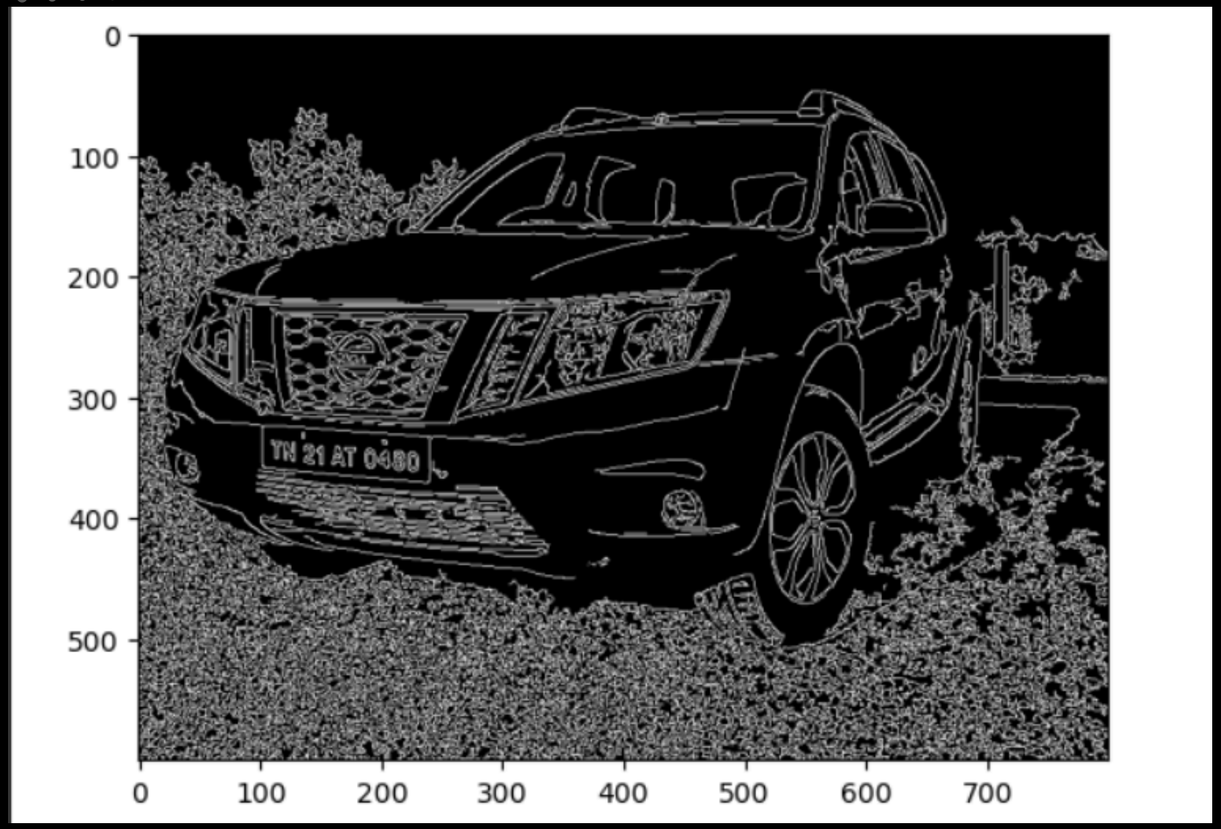
RELEVANCE

- Automatic traffic monitoring and enforcement
- Parking management
- Enhanced security

- Preprocessing
- License Plate Detection
- Bounding Box and Cropping
- Image Enhancement
- Text Extraction

METHODOLOGY

PREPROCESSING



Convert the image to grayscale, apply bilateral filtering to reduce noise while preserving edges, and then use the Canny algorithm for edge detection to highlight the significant boundaries in the image.

```python
# Function to preprocess the image: Convert to grayscale, apply filter, and detect edges
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

    # Detect the bounding box of the plate
    x1, x2, y1, y2, bbox = detect_number_plate(edged, gray, img)
    # Crop the region containing the plate
    cropped_image = gray[x1:x2+1, y1:y2+1]
    return cropped_image, bbox
```

CANNY EDGE DETECTION

1. Noise Reduction(Gaussian Filtering)

The noise in the image can be removed with the help of Guassian filters.it is a linear filter that uses Guassian distribution function to smooth images and reduce noise.

2. Gradient calculation (Edge Intensity and Direction)

To detect regions with rapid intensity changes (potential edges).
Use Sobel operators to compute gradients in the horizontal (Gx) and vertical (Gy) directions.

CANNY EDGE DETECTION

3. Non maximum suppression (Thinning the Edges)

The gradient magnitude image will often have multiple pixel values that might be considered as edges. Non-maximum suppression helps to thin out these edges by keeping only the local maxima along the edge direction.

4. Double Thresholding (Edge Classification)

Pixels are classified as strong edges, weak edges, or non-edges based on their gradient magnitudes. Strong edges exceed a high threshold and are retained, while non-edges fall below a low threshold and are discarded. Weak edges, which lie between these thresholds, are kept only if connected to strong edges, ensuring continuous, meaningful edges while filtering out noise.

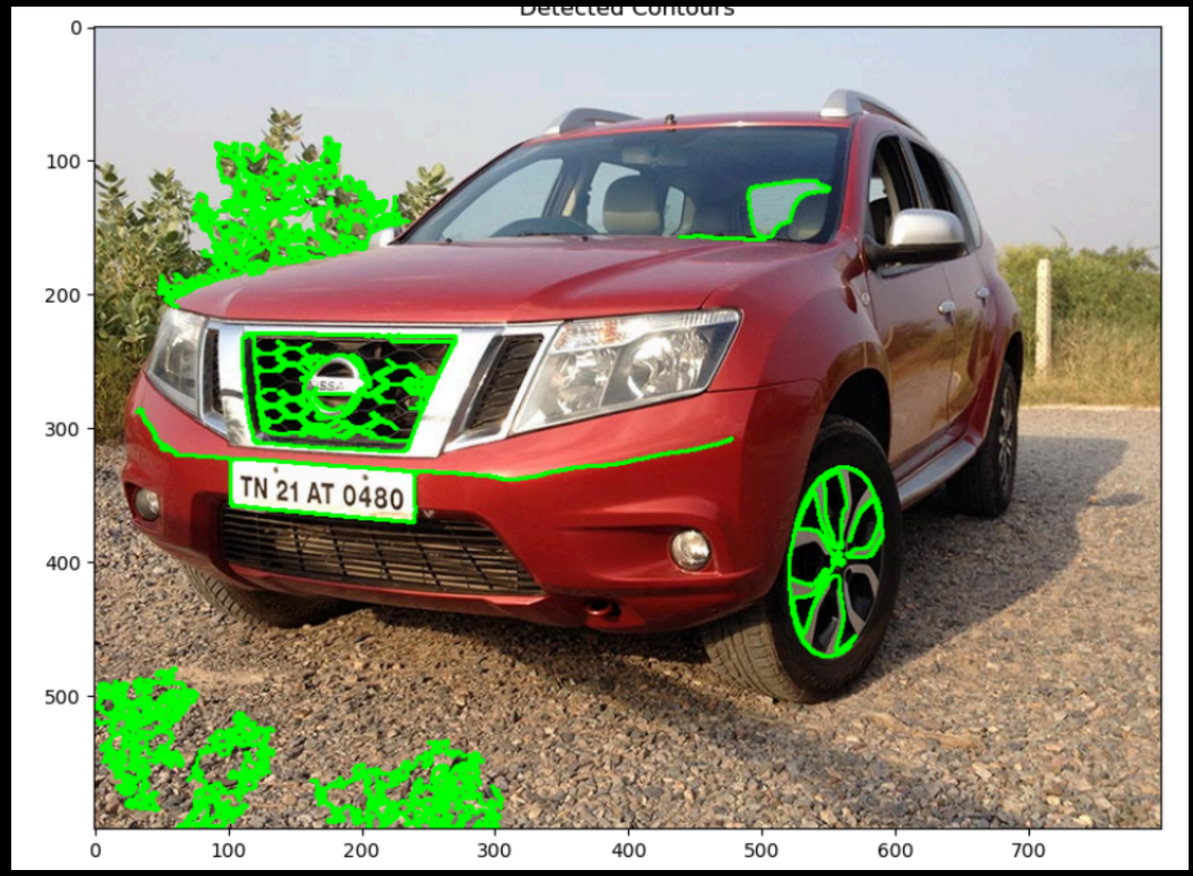
CANNY EDGE DETECTION

5. Edge Tracking by Hysteresis

Edge Tracking by Hysteresis is the final step in Canny Edge Detection, ensuring that only meaningful edges remain. Starting from strong edge pixels, the algorithm checks adjacent weak pixels and includes them in the edge map if they connect to strong edges. This preserves weak edges that form continuous edge structures while discarding isolated weak edges, likely caused by noise. Pixels not meeting these criteria are set to zero, cleaning up the edge map. The result is a refined edge-detected image with only significant, continuous edges retained for clearer analysis and further processing.

LICENSE PLATE DETECTION

Involves detecting contours to identify potential plate boundaries, which are filtered based on criteria like aspect ratio and area to accurately locate the rectangular shape of the license plate.



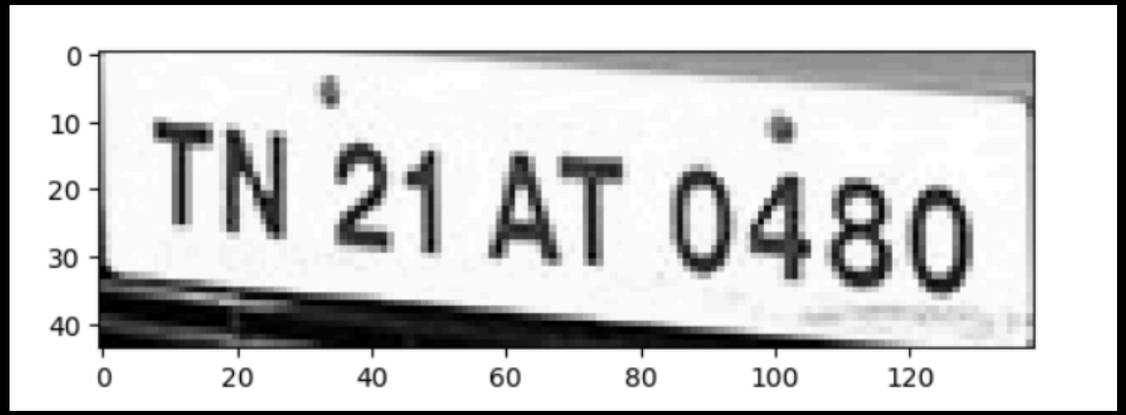
```python
# Function to detect the bounding box of the number plate in the image
def detect_number_plate(edged, gray, img):
    contours = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    contours = sorted(contours, key=cv2.contourArea, 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)  # Create a mask for the detected plate region
        new_image = cv2.drawContours(mask, [bbox], 0, 255, -1)
        new_image = cv2.bitwise_and(img, img, mask=mask)  # Apply the mask to the original image

        (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:
        print("No bounding box found")
        return 0, 0, 0, 0, None
```

BOUNDING BOX & CROPPING

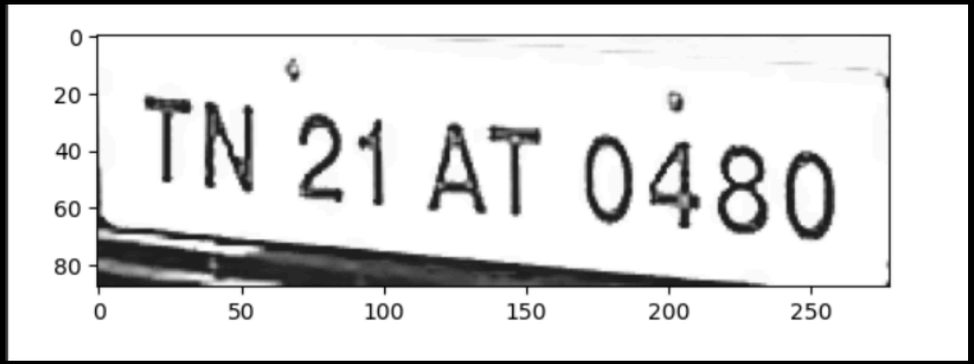
Generate a bounding box around the detected plate, crop it, and isolate it for clearer text extraction



```python
# Function to find the bounding box around the number plate from the contours
def find_bbox(contours):
    for contour in contours:
        # Approximate the contour to a polygon
        approx = cv2.approxPolyDP(contour, 10, True)
        # Check if the contour has four points (rectangle-like)
        if len(approx) == 4:
            return approx
    return None
```

IMAGE ENHANCEMENT

Resize, sharpen, and adjust brightness and contrast of the cropped plate image for better OCR performance



```python
# Function to enhance the image for better OCR results
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]])
    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

Use EasyOCR to read and clean the text, ensuring high readability

```python
# OCR function using 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
```

RESULTS & OBSERVATION

THANK YOU

ADHITHYA E D	B220113CS
AJMAL	B220138CS
RACHEL PAUL	B221138CS