Project Assignment-3

Reference Papers Implemented:

1) Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach

Authors : K Yogheedha, ASA Nasir, H Jaafar, SM Mamduh

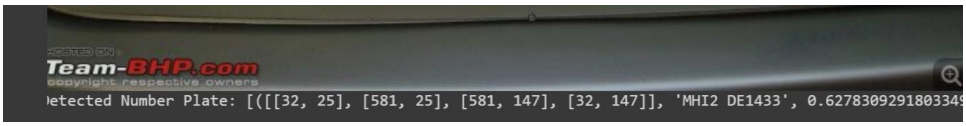
Year : 2018

	K Yogheedha etc. 2018	Image processing+OCR based on template matching	• Effective even under minimal lighting conditions.	• Difficulties in recognizing nonstandard license plate characters	[3] K. Yogheedha, A. S. A. Nasir, H. Jaafar and S. M. Mamduh, "Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach," 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia, 2018, pp. 1-8, doi: 10.1109/ICASSDA.2018.8477639.

```
import cv2
from google.colab.patches import cv2_imshow
import easyocr

def detect_number_plate(image_path, template_path):
    # Load the input image and the template
    img = cv2.imread(image_path)
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)

    # Check if image and template are loaded successfully
    if img is None or template is None:
        print("Error: Could not load image or template.")
        return
```

```
# Convert the input image to grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Match the template with the input image
result = cv2.matchTemplate(img_gray, template, cv2.TM_CCOEFF_NORMED)
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)

# Get the coordinates of the matched region
top_left = max_loc
h, w = template.shape

# Draw a rectangle around the matched region
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img, top_left, bottom_right, (0, 255, 0), 2)

# Extract the region of interest (ROI) containing the number plate
roi = img[top_left[1]:bottom_right[1], top_left[0]:bottom_right[0]]

# Convert the ROI to grayscale
roi_gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

# Use EasyOCR to extract text from the number plate region
reader = easyocr.Reader(['en'])
plate_text = reader.readtext(roi_gray)

# Display the result using cv2_imshow
cv2_imshow(img)
print("Detected Number Plate:", plate_text)
```

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Example usage
image_path = 'img_path.jpg'
template_path = 'template_path.jpg'
detect_number_plate(image_path, template_path)
```



Observations:

1. *Template Matching:* The script uses template matching to find a template (presumably a number plate pattern) within the input image.

2. *Rectangle Drawing:* It draws a rectangle around the detected region, providing a visual indication of where the number plate is located.

3. *EasyOCR Integration:* The Optical Character Recognition (OCR) library, EasyOCR, is utilized to extract text from the detected number plate region.

4. *Displaying Results:* The script displays the original image with the drawn rectangle and prints the detected number plate text.

Future Scope:

1. *Improving Accuracy:* Evaluate and enhance the accuracy of number plate detection. This could involve experimenting with different computer vision techniques or fine-tuning parameters.

2. *Adaptability:* Make the script more adaptable to different scenarios by allowing the user to input various templates or dynamically adapting to different plate patterns.

3. *Error Handling:* Implement robust error handling to manage cases where the image or template fails to load or when OCR doesn't yield satisfactory results.

4. *Performance Optimization:* Optimize the code for better performance, especially for large images or when processing multiple images.

5. *User Interaction:* Consider adding a user interface or integrating the script into a larger application, making it more user-friendly.

6. *Localization:* Extend the script to handle number plates in different languages or formats, accommodating variations in plate design.

7. *Integration with Database:* If applicable, integrate the script with a database to store and manage recognized number plate information.

8. *Security Applications:* Explore applications in security, surveillance, or access control by integrating the script into a broader system.