PROJECT TITLE: License Plate Recognition with OpenCV and Tesseract OCR

License Plate Recognition is widely used for automated identification of vehicle registration plates for security purpose and law enforcement. By combining computer vision techniques with Optical Character Recognition (OCR) we can extract license plate numbers from images enabling applications in areas like security, traffic management and toll systems.

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

Double-click (or enter) to edit

```
!sudo apt-get install tesseract-ocr
!pip install opencv-python pytesseract
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tesseract-ocr is already the newest version (4.1.1-2.1build1).
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: pytesseract in /usr/local/lib/python3.11/dist-packages (0.3.13)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packages (from opencv-python) (2.0.2)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (24.2)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from pytesseract) (11.1.0)
```

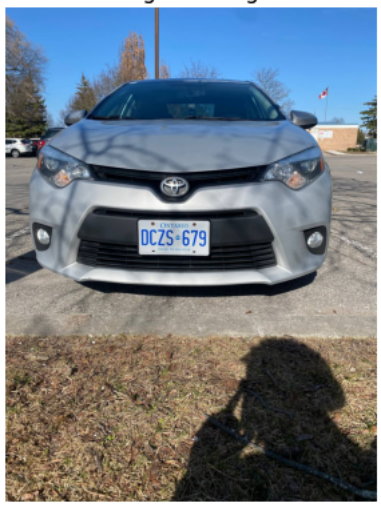
```
import cv2
import pytesseract
import matplotlib.pyplot as plt
import numpy as np

# Set the path to the Tesseract executable
pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
```

```
def detect_license_plate(image_path):
    image = cv2.imread(image_path)
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.title('Original Image')
    plt.show()

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(blurred, 100, 200)
    contours, _ = cv2.findContours(edges.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=cv2.contourArea, reverse=True)

    plate_contour = None
    for contour in contours:
```

```
        epsilon = 0.02 * cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, epsilon, True)
        if len(approx) == 4:
            plate_contour = approx
            break

    plate_number = None
    if plate_contour is not None:
        x, y, w, h = cv2.boundingRect(plate_contour)
        plate_image = gray[y:y+h, x:x+w]
        _, thresh = cv2.threshold(plate_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

        kernel = np.ones((3, 3), np.uint8)
        morphed = cv2.dilate(thresh, kernel, iterations=1)

        # Optional resizing to improve OCR accuracy
        resized = cv2.resize(morphed, None, fx=2, fy=2, interpolation=cv2.INTER_LINEAR)

        plate_number = pytesseract.image_to_string(resized, config='--psm 7')
        plate_number = plate_number.strip()
        print("Detected License Plate Number:", plate_number)

        cv2.imwrite("detected_plate.jpg", resized)
        plt.imshow(resized, cmap='gray')
        plt.title('Processed Plate Region')
        plt.axis('off')
        plt.show()

    else:
        print("License plate not detected.")
    return plate_number
```

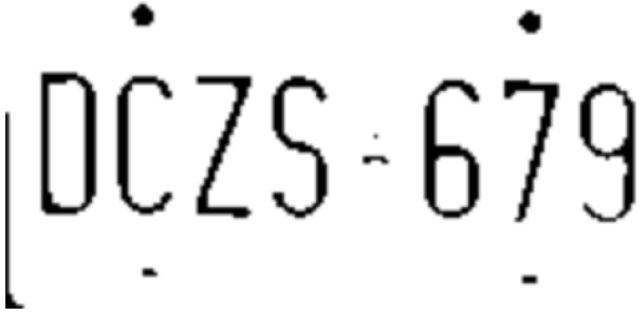
```
image_path = "corolla_2.jpg"
plate_number = detect_license_plate(image_path)
print("Final Output:", plate_number)
```

Original Image




```
Detected License Plate Number: OCZS-679
```

Processed Plate Region




```
Final Output: OCZS-679
```