Lane Violation Detection

S/18/580 | K.A. D. Lakruwan

INTRODUCTION

- The objective of this project is to develop a Lane Violation Detection System that can accurately detect and identify vehicles that violate traffic lanes. The system will analyze input video footage captured from traffic surveillance cameras to perform the above functionality.
- In addition to this, the system will try to crop the license plate of the detected vehicle.
- This system aims to enhance the road safety, traffic management and law enforcement by effectively identifying and alerting authorities about vehicles that violate road lanes

Objectives

- Identify the vehicles that commit lane violations.
  - Identify the moving objects.
  - Check whether a vehicle is inside the region of interest.
- Identify and read the license plates of those vehicles.

openCV usage

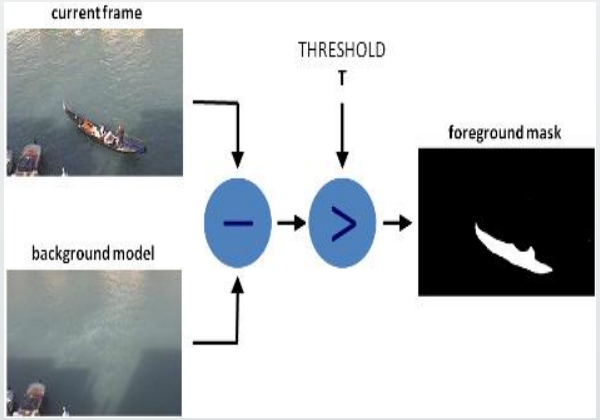
In order to achieve aforementioned objectives I've use several openCV libraries .

- Detecting the moving vehicles
  - cv2. createBackgroundSubtractorMOG2()
  - cv2. createBackgroundSubtractorKNN()
  - cv2.morphologyEx()
  - cv2.getStructuringElement()
  - cv2.findContours
  - cv2.VideoCapture
- Find whether the vehicle is in the ROI
  - cv2.pointPolygonTest
  - To Identify the license plate : cv2.gaussianBlur / cv2.threshold / cv2.canny / cv2.fincontours / cv2.approxPolyDP

Implementation

- I have implemented the object detection using background subtraction method.
  - Background Subtraction method

Each pixel in the current frame is compared to its corresponding pixel in the background model. The difference between the current pixel and the background pixel is calculated. If the difference exceeds a predefined threshold, the pixel is considered part of the foreground.



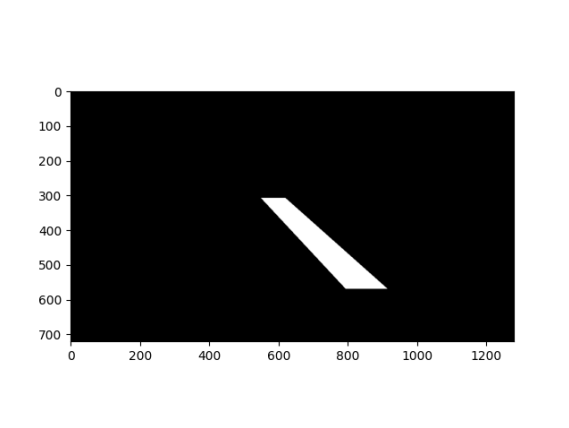
but for the most of the time this foreground would get disturbed by the noise therefore morphological operations such as erosion and dilation is usually applied to this mask.

- openCV offers many background Subtraction algorithms such as cv2.backgroundSubtractorMOG2(), cv2.backgroundSubtractorKNN() . In these algorithms instead of just subtracting a single frame from another they will collect and choose a certain number of frames as background and then they will compare it with the other frames to obtain the foreground.
  - **dist2Threshold**: It determines the threshold for deciding whether a pixel is part of the background or foreground.
  - **detectShadows**: A boolean parameter that specifies whether the algorithm should detect and label shadow
- In this project I have use cv2.backgroundSubtractorKNN() to create the backgroundsubtractor object.
- This object helps in segmenting the moving foreground objects from the static background.
- The background subtractor is applied to the frame to obtain a segmented mask using backgroundObject.apply(frame). This mask helps in isolating the moving objects in the frame.
- And then for the foreground mask I've applied erosion, dilation and closing to reduce noise. There after I have use a threshold to remove the shadows. Finally I've find the contours and drawn the bounding rectangle around it.

```python
def initialize_background_subtractor():
    return cv2.createBackgroundSubtractorKNN( dist2Threshold=400,detectShadows=True)
    # return cv2.createBackgroundSubtractorMOG2( varThreshold=50,detectShadows=True)


def apply_mask(frame, background_object, kernel):
    fgmask = background_object.apply(frame)
    _, fgmask = cv2.threshold(fgmask, 250, 255, cv2.THRESH_BINARY)
    # fgmask=cv2.morphologyEx(fgmask,cv2.MORPH_OPEN,kernel)
    fgmask=cv2.morphologyEx(fgmask,cv2.MORPH_CLOSE,kernel)
    # fgmask = cv2.erode(fgmask, kernel, iterations=1)
    # fgmask = cv2.dilate(fgmask, kernel, iterations=2)
    return fgmask

```

```python
def find_and_draw_cars(frame, fgmask, roi_polygon, output_folder):
    contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    frame_copy = frame.copy()
    unique_image_id = 0

    for cnt in contours:
        if cv2.contourArea(cnt) > 400:
            x, y, width, height = cv2.boundingRect(cnt)

            if cv2.pointPolygonTest(roi_polygon, (x + width // 2, y + height // 2), False) >= 0:
                cv2.rectangle(frame_copy, (x, y), (x + width, y + height), (0, 0, 255), 2)
                cv2.putText(frame_copy, 'Violation detected', (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 255, 0), 1, cv2.LINE_AA)
                car_region = frame[y:y + height, x:x + width]
                unique_image_id += 1
                image_name = os.path.join(output_folder, f"car_{unique_image_id}.jpg")
                cv2.imwrite(image_name, car_region)
            else:
                cv2.rectangle(frame_copy, (x, y), (x + width, y + height), (0, 255, 0), 2)

    return frame_copy
```

- Then for each video I have define a ROI that includes the lane that vehicles are tend to violate then by calculating the center point of the bounding box of a contour we check whether it lies in ROI using **cv2.pointPolygonTest.** Then finally if a contour is in ROI it's marked with a red bounding box.

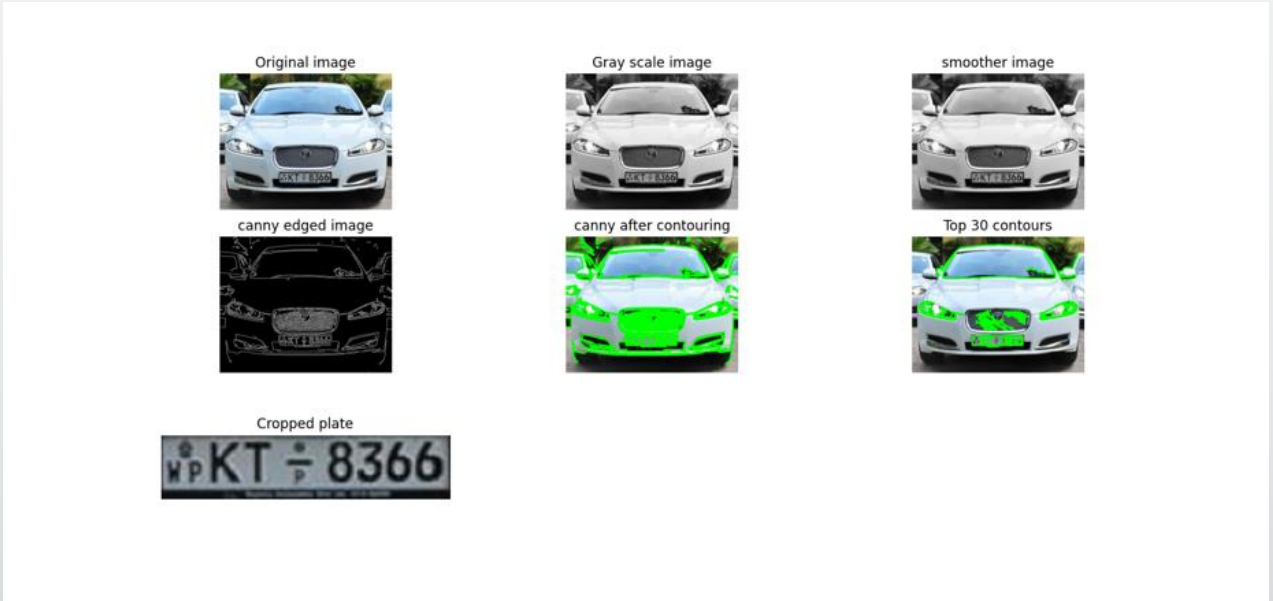


- Then I would store those frames where lane violations occurred in a different folder.

- But in most of these images license plate is either blured or not clear.
- My initial intention was to feed these images to the license plate detection system but as license plates are blured in these images I'm going to demonstrate it using a different set of images.

License plate Detection System

- Dataset
  - https://www.kaggle.com/datasets/andrewmvd/car-plate-detection?resource=download

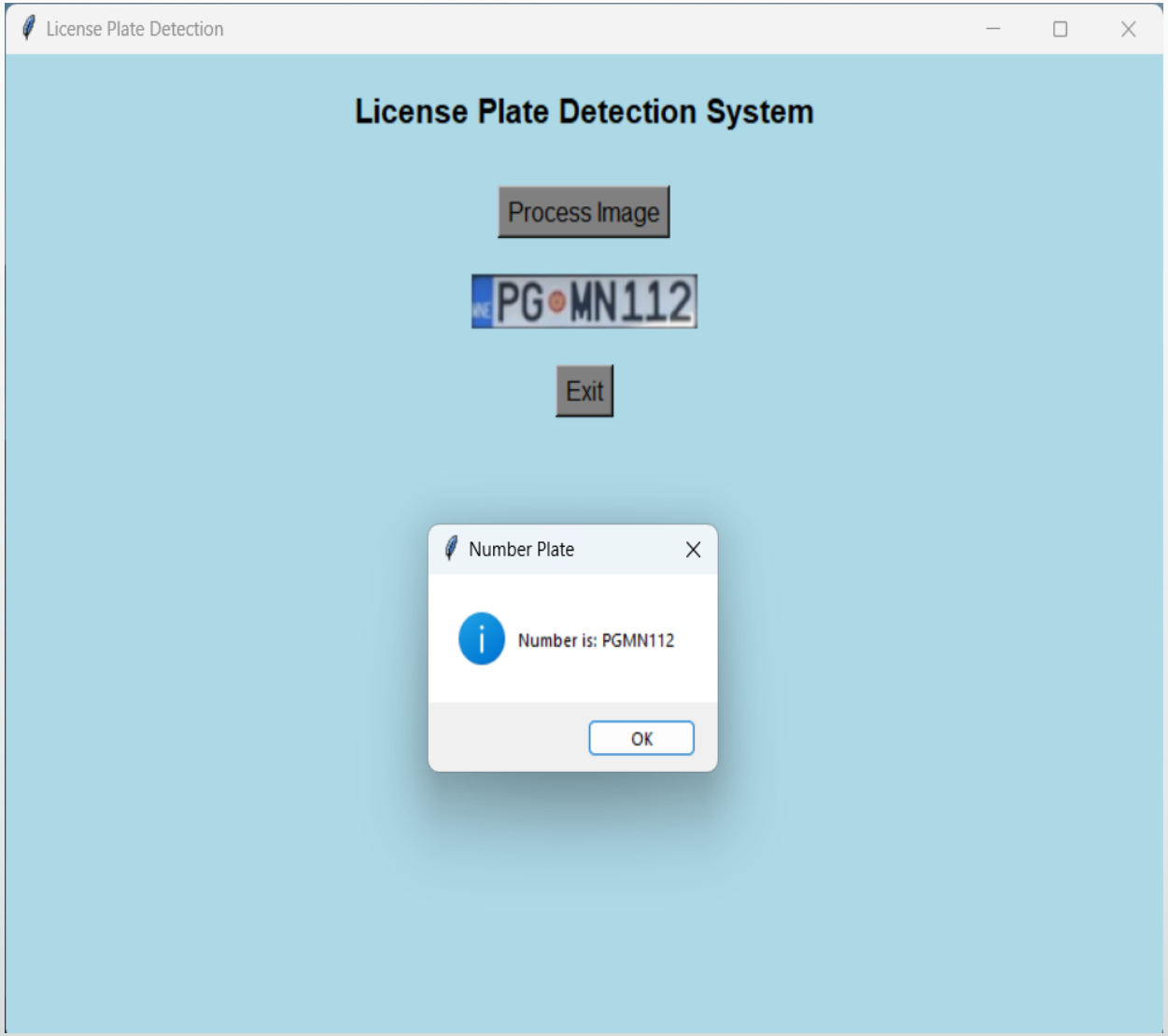


- Apply preprocessing mechanisms such as GaussianBlur(),binary thresholding to smooth the gray scaled image
- Then perform the canny edge detection
- Then find the contours and filter out the most prominent contours
- Then find contours with four vertices using cv2.approxPolyDP() function
- Then crop the rectangular contour using it's bounding box.
- Finally read the text of the cropped plate using tesseract OCR.

```python
image = imutils.resize(image, width=500)
        #convert the image to grayscale
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        #binary thresholding
        ret1,bina = cv2.threshold(gray,127,255,cv2.THRESH_BINARY)
        #smoothing using bilateral filter
        smoothed = cv2.bilateralFilter(bina, 11, 17, 17)
        #bluring using gaussian blur
        blur =cv2.GaussianBlur(smoothed,(5,5),0)


        # Edge detection
        #? if the pixel is less than threshold_1 it is discarded and if the pixel is higher than threshold_2 it is considerd an edge.
        edged = cv2.Canny(blur, 170, 200)

```

```python
#?Loop through the sorted cnts array.
      for i in cnts:
          #?calcultes the perimater of the contour.
          perimeter = cv2.arcLength(i, True)
          #?It approximate the contour as a simple polygon and the accuracy is given by 0.002*perimeter.
          approx = cv2.approxPolyDP(i, 0.02 * perimeter, True)
          #?If the simplified polygon has four points it is assumed to be the liscense palte.
          if len(approx) == 4:
              x, y, w, h = cv2.boundingRect(i)
              cropped_image = image[y:y+h, x:x+w]

              # Extract text from the number plate
              #?This configuration detects the capital letters and the numbers.
              text = pytesseract.image_to_string(Image.fromarray(cropped_image), config='--psm 10 -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')

```

- Thereafter I've created a simple GUI to display the results.



Challenges

- Issues related to background subtraction method when used for object detection.
  - Background subtraction is greatly affected by the noise hence it produces many false positive results.
  - In background subtraction method there is no any method to detect overlapping vehicles.
  - Have to manually change parameters according to the each video as the background is changing.
- Lack of quality in the videos.
  - Most of the videos are either blurred or they have been captured in angle which is not suitable for object detection.
- Unavailability of a robust Object Tracking Algorithm in openCV.

Project Schedule and Milestones

THANK YOU!