Automatic Number Plate Recognition System

Automatic License/Number Plate Recognition (ANPR) is the process of detecting position of number plate and then using Optical Character Recognition technique to identify the text on the plate from a image or video. It is widely used in video surveillance and for security purposes. In this article, we'll see various steps to implement ANPR using Python.

Step 1: Importing Required Libraries

We are using OpenCV, EasyOCR, matplotlib, numpy, random and imutils.

import cv2

from matplotlib import pyplot as plt

import numpy as np

import easyocr

import imutils

import random

Step 2: Reading Image and Applying Filters

First we read the image and then clean the image by image preprocessing techniques like converting it to grayscale and then apply a bilateral filter to reduce noise.

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)): Converts the image from BGR (OpenCV default) to RGB format and displays it using matplotlib.

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY): Converts the original image from BGR to grayscale to simplify further processing.

bfilter = cv2.bilateralFilter(gray, 11, 17, 17): Applies a bilateral filter to the grayscale image to reduce noise while preserving edges.

from google.colab import files

uploaded = files.upload()

img = cv2.imread('Test1.jpg')

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.title('Original Image')

plt.show()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

bfilter = cv2.bilateralFilter(gray, 11, 17, 17)

plt.imshow(cv2.cvtColor(bfilter, cv2.COLOR\_BGR2RGB))

plt.title('Processed Image')

plt.show()

Step 3: Edge Detection

Canny edge detection algorithm identifies edges in a image by first smoothing it to reduce noise then detecting areas with sharp brightness changes. It uses double threshold to separate strong and weak edges. This process highlights the important edges which makes it easier to separate number plate.

edged = cv2.Canny(bfilter, 30, 200): Applies Canny edge detection algorithm to the filtered image helps in detecting edges based on the specified threshold values.

plt.imshow(cv2.cvtColor(edged, cv2.COLOR\_BGR2RGB)): Converts edge-detected image from BGR to RGB format and displays it using matplotlib.

edged = cv2.Canny(bfilter, 30, 200)

plt.imshow(cv2.cvtColor(edged, cv2.COLOR\_BGR2RGB))

plt.title('Edge Detection')

plt.show()

Step 4: Finding Contours

Contours are boundaries of shapes with the same intensity in an image. They helps in identifying objects or separate them from the background. The cv2.findContours() function is used to detect these contours in binary images which makes it easier to locate and analyze specific regions within the image.

keypoints = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE): Finds contours in the edge-detected image using the RETR\_TREE retrieval mode and the CHAIN\_APPROX\_SIMPLE contour approximation method.

contours = imutils.grab\_contours(keypoints): Extracts contours from the keypoints returned by cv2.findContours() usin imutils.grab\_contours().

contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]: Sorts the contours by area in descending order and selects the top 10 largest contours for further processing.

approx = cv2.approxPolyDP(contour, 10, True): Loops through sorted contours and approximates each contour to a polygon with a precision of 10 pixels.

if len(approx) == 4: Checks if the approximated contour has four sides which indicating a quadrilateral and assigns it to location (license plate).

keypoints = cv2.findContours(edged.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

contours = imutils.grab\_contours(keypoints)

contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]

location = None

for contour in contours:

approx = cv2.approxPolyDP(contour, 10, True)

if len(approx) == 4:

location = approx

break

print("Location: ", location)

Step 5: Marking Number Plate

After this our next step will be masking out only area which include number plate so that later when we are going to extract text from it using OCR we can do it easily.

mask = np.zeros(gray.shape, np.uint8): Creates a blank mask with same dimensions as grayscale image and it initialized to zeros (black).

new\_image = cv2.drawContours(mask, [location], 0, 255, -1): Draws contour (location of the license plate) on the mask filling the contour with white (255).

new\_image = cv2.bitwise\_and(img, img, mask=mask): Applies the mask to the original image which retains only area within the contour (license plate region).

plt.imshow(cv2.cvtColor(new\_image, cv2.COLOR\_BGR2RGB)): Converts masked image from BGR to RGB format and displays it using matplotlib.

mask = np.zeros(gray.shape, np.uint8)

new\_image = cv2.drawContours(mask, [location], 0, 255, -1)

new\_image = cv2.bitwise\_and(img, img, mask=mask)

plt.imshow(cv2.cvtColor(new\_image, cv2.COLOR\_BGR2RGB))

plt.title('Masked Image')

plt.show()

Step 6: Extracting Text from Using OCR

(x, y) = np.where(mask == 255): Finds coordinates of the white pixels in mask which represent the license plate area.

(x1, y1) = (np.min(x), np.min(y)): Identifies top-left corner of the bounding box by finding the minimum x and y coordinates.

(x2, y2) = (np.max(x), np.max(y)): Identifies bottom-right corner of the bounding box by finding the maximum x and y coordinates.

cropped\_image = gray[x1:x2+1, y1:y2+1]: Crops image using the identified coordinates to isolate the license plate region.

(x, y) = np.where(mask == 255)

(x1, y1) = (np.min(x), np.min(y))

(x2, y2) = (np.max(x), np.max(y))

cropped\_image = gray[x1:x2+1, y1:y2+1]

plt.imshow(cv2.cvtColor(cropped\_image, cv2.COLOR\_BGR2RGB))

plt.title('Cropped Image')

plt.show()

Now it is an important step in ANPR to convert image into text. This step will help us to use the number plate data. We can store the data on number plate on database and use it later for number of applications like automatic toll or automatic parking charges etc.

reader = easyocr.Reader(['en']): Initializes an EasyOCR reader for the English language helps in allowing text extraction from images.

result = reader.readtext(cropped\_image): Uses EasyOCR to extract text from the cropped license plate image and stores the result.

reader = easyocr.Reader(['en'])

result = reader.readtext(cropped\_image)

Step 7: Displaying Output

Once we've extracted the text from number plate we add it to the original image. We draw a rectangle around the license plate and place the text next to it. Finally the image with the text and rectangle is displayed.

text = result[0][-2]: Extracts recognized text from the OCR result (the number plate text).

font = cv2.FONT\_HERSHEY\_SIMPLEX: Specifies font type (Hershey Simplex) for drawing text on the image.

res = cv2.putText(img, text=text, org=(location[0][0][0] ... lineType=cv2.LINE\_AA): Adds detected text on original image near the number plate using specified font, color and position.

res = cv2.rectangle(img, tuple(location[0][0]), tuple(location[2][0]), (0, 255, 0), 3): Draws a green rectangle around detected license plate area on original image.

text = result[0][-2]

print("Detected Text: ", text)

font = cv2.FONT\_HERSHEY\_SIMPLEX

res = cv2.putText(img, text=text, org=(location[0][0][0], location[1][0][1] + 60), fontFace=font, fontScale=1, color=(0, 255, 0), thickness=2, lineType=cv2.LINE\_AA)

res = cv2.rectangle(img, tuple(location[0][0]), tuple(location[2][0]), (0, 255, 0), 3)

plt.imshow(cv2.cvtColor(res, cv2.COLOR\_BGR2RGB))

plt.title('Final Image with Text')

plt.show()

With this approach we can perform Automatic License/Number Plate Recognition (ANPR) using Python by filtering, masking and extracting the number plate text with EasyOCR.