from skimage.io import imread

from skimage.filters import threshold\_otsu

import matplotlib.pyplot as plt

import imutils

import cv2

from skimage import measure

from skimage.measure import regionprops

import matplotlib.pyplot as plt

import matplotlib.patches as patches

import os

import shutil

!sudo apt install tesseract-ocr

from google.colab.patches import cv2\_imshow

!pip3 install pytesseract

import pytesseract

# car image -> grayscale image -> binary image

car\_image = imread("/content/test.png", as\_gray=True)

#car\_image = imutils.rotate(car\_image, 270)

# car\_image = imread("car.png", as\_gray=True)

# it should be a 2 dimensional array

print(car\_image.shape)

# the next line is not compulsory however, a grey scale pixel

# in skimage ranges between 0 & 1. multiplying it with 255

# will make it range between 0 & 255 (something we can relate better with

gray\_car\_image = car\_image \* 255

fig, (ax1, ax2) = plt.subplots(1, 2)

ax1.imshow(gray\_car\_image, cmap="gray")

threshold\_value = threshold\_otsu(gray\_car\_image)

binary\_car\_image = gray\_car\_image > threshold\_value

# print(binary\_car\_image)

ax2.imshow(binary\_car\_image, cmap="gray")

# ax2.imshow(gray\_car\_image, cmap="gray")

plt.show()

Erode unwanted pixels

    img\_binary\_lp = cv2.erode(img\_binary\_lp, (3,3))

    # Dilate unwanted pixels

    img\_binary\_lp = cv2.dilate(img\_binary\_lp, (3,3))

    LP\_WIDTH = img\_binary\_lp.shape[0]

    LP\_HEIGHT = img\_binary\_lp.shape[1]

    # Make borders white

    img\_binary\_lp[0:3,:] = 255

    img\_binary\_lp[:,0:3] = 255

    img\_binary\_lp[72:75,:] = 255

    img\_binary\_lp[:,330:333] = 255

# connected component analysis(CCA) (finding connected regions) of binary image

# this gets all the connected regions and groups them together

label\_image = measure.label(binary\_car\_image)

def getVProjection(image):

    #  The image image is converted to a black and white two-value map, and the RET receives the current threshold, the two-value chart of the output of the output

    ret, thresh1 = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

    (h,w)=thresh1.shape #Return high and wide

    a = [0 for z in range(0, w)] #A = [0, 0, 0, ..., 0, 0] initializes an array of length W for recording the number of black points of each column

    #Record the peak of each column

    for j in range(0,w): #Traverse one column

        for i in range(0,h):  #Traverse

            if  thresh1[i,j]==0:  #If this point is black

                a[j]+=1          #The column's counter adds a count

                thresh1[i,j]=255  #After the record is finished, it will be white.

    for j  in range(0,w):  #Traverse each column

        for i in range((h-a[j]),h):  #From the top of this column, the top of the black is started to be blacked to the bottom

            thresh1[i,j]=0   #Black

    cv2\_imshow(thresh1)

def getHProjection(image):

    # The image image is converted to a black and white two-value map, and the RET receives the current threshold, the two-value chart of the output of the output

    ret, thresh1 = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY)

    (h,w)=thresh1.shape #Return high and wide

    a = [0 for z in range(0, h)]  #A = [0, 0, 0, ..., 0, 0] initializes an array of length h, used to record the number of black spots of each line

    #Record the peak of each line

    for j in range(0,h): #Traverse

        for i in range(0,w):  #Traverse one column

            if  thresh1[j,i]==0:  #If this point is black

                a[j]+=1          #The column's counter adds a count

                thresh1[j,i]=255  #After the record is finished, it will be white.

    for j  in range(0,h):  #Traverse every line

        for i in range(a[j]):  #From the top of this line, the number of black dots were burned to the right.

            thresh1[j,i]=0   #Black

    cv2\_imshow(thresh1)

if \_\_name\_\_ == '\_\_main\_\_':

    img = cv2.imread('test.jpg')

    cv2\_imshow(img)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

 

