5/28/2024

Jaya Chaithanya

License Plate detection

[Document subtitle]

Number Plate Detection

What:

Number plate detection is a type of detection in which it detects only the license plate of the car. It can be used in the highways to capture the vehicle which is moving faster than the limit. This project uses Python. It also requires Time, CV2, Imutils, NumPy, Pandas and pytesseract.

Code:

import time  
import cv2  
import imutils  
import numpy as np  
import pandas as pd  
import pytesseract

These lines are used to import Time, CV2, Imutils, NumPy, Pandas and pytesseract.

pytesseract.pytesseract.tesseract\_cmd = **r"C:\Users\jayac\AppData\Local\Programs\Python\Python39\Scripts\pytesseract.exe"**

This line is used to import the pytesseract.

image = cv2.imread(**'image.jpg'**)  
image = imutils.resize(image, width=500)  
cv2.imshow(**"Original Image"**, image)

These lines are used to import the image and used to show the image for verification. It supports .jpg and .png format images only

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
cv2.imshow(**"Grayscale Conversion"**, gray)

These lines are used to convert the colored image into a black and white image (gray-scale image) for enhanced operation.

gray1 = cv2.bilateralFilter(gray, 11, 17, 17)  
cv2.imshow(**"Bilateral Filter"**, gray1)

Then the image is then blurred to reduce the noise.

edged = cv2.Canny(gray, 170, 200)  
cv2.imshow(**"Canny Edges"**, edged)

These lines are used to perform edge detection to the image for extracting the data.

cv2.waitKey(0)  
cv2.destroyAllWindows()

These lines are used to close the opened windows.

(cnts, \_) = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)  
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:30]  
  
NumberPlateCnt = None  
count = 0

These lines are used to find contours in the edged images.

for c in cnts:  
 # approximate the contour  
 peri = cv2.arcLength(c, True)  
 approx = cv2.approxPolyDP(c, 0.02 \* peri, True)  
 # if the approximated contour has four points, then assume that the screen is found  
 if len(approx) == 4:  
 NumberPlateCnt = approx  
 break

This snippet is used for looping the contours. If the approximated contour has four points, then assume that the screen is found.

mask = np.zeros(gray.shape, np.uint8)  
new\_image = cv2.drawContours(mask, [NumberPlateCnt], 0, 255, -1)  
new\_image = cv2.bitwise\_and(image, image, mask=mask)  
cv2.namedWindow(**"Final Image"**, cv2.WINDOW\_NORMAL)  
cv2.imshow(**"Final Image"**, new\_image)

This snippet of code is for masking the other part of the license plate. Here, it will show the exact accurately cropped part of the license plate

config = **'-l eng --oem 1 --psm 3'**

This line is used for configuration of the tesseract.

text = pytesseract.image\_to\_string(new\_image, config=config)

This small line of snippet is used for run tesseract OCR on image

print(text)

This line is used to print the number in the license plate of the vehicle.

cv2.waitKey(0)  
cv2.destroyAllWindows()

These lines are used to destroy the opened the final window.

Full Code:

# works in IDLE 3.9 only...  
# try using in idle  
# it converts the image into biterial filter then grayscale , resizing and canny edges and gives the final image of the number plate  
import time  
import cv2  
import imutils  
import numpy as np  
import pandas as pd  
import pytesseract  
  
pytesseract.pytesseract.tesseract\_cmd = **r"C:\Users\jayac\AppData\Local\Programs\Python\Python39\Scripts\pytesseract.exe"**# read and resize image to the required size  
image = cv2.imread(**'image.jpg'**)  
image = imutils.resize(image, width=500)  
cv2.imshow(**"Original Image"**, image)  
  
# convert to gray scale  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
cv2.imshow(**"Grayscale Conversion"**, gray)  
  
# blur to reduce noise  
gray1 = cv2.bilateralFilter(gray, 11, 17, 17)  
cv2.imshow(**"Bilateral Filter"**, gray1)  
  
# perform edge detection  
edged = cv2.Canny(gray, 170, 200)  
cv2.imshow(**"Canny Edges"**, edged)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
  
# find contours in the edged image  
(cnts, \_) = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)  
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:30]  
  
NumberPlateCnt = None  
count = 0  
# loop over contours  
for c in cnts:  
 # approximate the contour  
 peri = cv2.arcLength(c, True)  
 approx = cv2.approxPolyDP(c, 0.02 \* peri, True)  
 # if the approximated contour has four points, then assume that screen is found  
 if len(approx) == 4:  
 NumberPlateCnt = approx  
 break  
  
# mask the part other than the number plate  
mask = np.zeros(gray.shape, np.uint8)  
new\_image = cv2.drawContours(mask, [NumberPlateCnt], 0, 255, -1)  
new\_image = cv2.bitwise\_and(image, image, mask=mask)  
cv2.namedWindow(**"Final Image"**, cv2.WINDOW\_NORMAL)  
cv2.imshow(**"Final Image"**, new\_image)  
  
# configuration for tesseract  
config = **'-l eng --oem 1 --psm 3'**# run tesseract OCR on image  
text = pytesseract.image\_to\_string(new\_image, config=config)  
  
# data is stored in CSV file  
raw\_data = {**'date'**: [time.asctime(time.localtime(time.time()))], **''**: [text]}  
df = pd.DataFrame(raw\_data)  
df.to\_csv(**'data.csv'**, mode=**'a'**)  
  
# print recognized text  
print(text)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

Input:



Output:

Grayscale conversion:



Biterial filter:



Canny Edges:



Final image:



Text Output:

