Algoritmo de reconocimiento de placas vehiculares

Realizamos la importación de las librerías opencv, easyocr.

Opencv

OpenCV (Open Source Computer Vision) es una librería software open-source de visión artificial y machine learning.

OpenCV provee una infraestructura para aplicaciones de visión artificial.

Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios

Easyocr

EasyOCR, como su nombre indica, es un paquete de Python que permite a los desarrolladores de visión artificial realizar sin esfuerzo el reconocimiento óptico de caracteres.

EasyOCR se implementa utilizando Python y la biblioteca PyTorch. Si tiene una GPU compatible con CUDA, la biblioteca de aprendizaje profundo PyTorch subyacente puede acelerar enormemente la detección de texto y la velocidad de OCR.

```
1   import cv2
2   import easyocr
```

En la línea número 5 instanciamos una variable, con la clase easyocr, dentro de los parámetros”([“es”,”en”])” definimos los lenguajes a utilizar, en este caso español e inglés. Además, definimos a gpu como True, debido a que el equipo de ejecución del algoritmo cuenta con una gráfica dedicada.

```
3   #
4   reader = easyocr.Reader(["es","en"], gpu = True)
```

En la línea 7 generamos un array con el nombre placa, y en la línea 8 realizamos la lectura del archivo de imagen objetivo, para la extracción de información de la placa .

```
6
7   placa = []
8   image = cv2.imread("placa1.jpg",1)
```

En la línea número 10 realizamos que la imagen cambié de color a una escala de grises, con la línea número 11 hacemos que el resultado obtenido por la línea número 10 se difemine, para obtener una imagen de menor calidad, esto con el objetivo de que el procesamiento de imagen tenga una menor duracion de tiempo, y el proceso sea mas acelerado.

```
9
10  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
11  gray = cv2.blur(gray, (3,3))
12  cv2.imshow('placablur', gray)
```

Entrada - Resultado

Para la niña número 14 realizamos que la imagen generada en escala de grises, se convierte en imagen donde resalten los bordes, Con la línea número 15, Buscamos un área donde rodea la placa, para determinarlo como contorno.

```
14  canny = cv2.Canny(gray,150,200)
15  canny = cv2.dilate(canny, None,iterations=1)
16  cv2.imshow('placablur', canny)
```

Entrada - Resultado

Ahora procedemos a buscar los contornos de canny que ya habíamos trabajado con la anterior imagen, esto se consigue con cv2.RETR_LIST.

```
17
18    cnts,_=cv2.findContours(canny,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
```

Ahora requerimos desechar el resto de los contornos exceptuando el contorno que envuelve a la placa, para lo cual generamos un for, esto con el objetivo de solo trabajar con contornos de un tamaño específico, tamaño de placa, debido que pueden existir contornos que superen el tamaño normal de una, además en la variable área almacenamos todos los contornos se encuentre en la imagen. Con cv2.boundingRect, generamos un rectángulo que envuelve la placa Es que con las variables x, y, alto y ancho. Posteriormente utilizáramos cv2.aproxPolyDP para determinar los vértices del contorno, con un porcentaje de 9%.

```
for c in cnts:
    area = cv2.contourArea(c)
    x,y,w,h = cv2.boundingRect(c)
    epsilon = 0.09*cv2.arcLength(c, True)
    approx = cv2.approxPolyDP (c, epsilon, True)
```

Ahora ingresamos a un if, con el objetivo de que encontremos un contorno con cuatro vértices, por eso la comparación de la variable aproximada a 4, además de esto el área debe ser mayor a 9000.

```
26      if len(approx)==4 and area>9000:
27          print('area=', area)
28          #cv2.drawContours (image, [c],e, (e,255,0), 2)
29          aspect_ratio = float(w)/h
```

Un aspect_radio definimos un margen de error de 2.0. Además de esto en la línea 40 almacenamos en la variable placa el área donde está presente la matrícula en la imagen en escala de grises, para esto utilizamos la información de x,y,w y w.

```
38          if aspect_ratio>2.0:
39              # Recorte de la placa
40              placa = gray[y:y+h,x:x+w]
```

Con la línea 43, realizamos la lectura con la librería esayocr, de los caracteres que se encuentran en la imagen de placa recortada.

```
43              reader = reader.readtext(placa)
```

En la línea 53 generamos un rectángulo que cubre la placa, además de un texto que identifique los valores encontrados del área seleccionada para el reconocimiento.

```
53              cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
```

Las líneas siguientes simplemente mostramos los resultados obtenidos en diferentes ventanas.

```
39
40  cv2.imshow('Image', image)
41  cv2.imshow('Canny', canny)
42  cv2.moveWindow('Image',45,10)
43  cv2.waitKey(0)
```

Resultados:

Una vez finalizada la ejecución del código, obtenemos una imagen etiquetada con lo encontrado.


Image
CVL 657 18?
CVL 657 18
CVL 657 18