RECONOCIMIENTO DE PLACAS

JULIAN GIRALDO C

JUAN DAVID ALVAREZ

IMPORTAMOS LIBRERÍAS - OPENCV

- La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar.
- 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.
- Se usa en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos.

PYTESSREACT

- El reconocimiento óptico de caracteres (OCR por sus siglas en inglés) es un sistema que proporciona un reconocimiento de caracteres alfanuméricos completo en una imagen. El sistema permite extraer texto de una imagen para luego convertirlo en un archivo editable.

LOCALIZAMOS EL ARCHIVO INSTALADO PARA EJECUTAR TESSREACT

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'
```

- Esta ruta varía según dónde se haya instalado el tesseract en la máquina

LECTURA DE IMAGEN

```
image = cv2.imread('auto001.jpg')
```

- Con el método ***cv2.imread***, el archivo jpg

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

- Con el método ***cv2.cvtColor***, se convierte la imagen previa a escalas de grises

DETECCIÓN DE BORDES

```
canny = cv2.Canny(gray,150,200)
```

- **Este método de detección de bordes lo que hace es:**

1. Toma la nueva imagen en escala de grises y de esta manera puede detectar los bordes blancos ya que al transformar dicha imagen, la placa no se verá afectada porque es de color blanco desde un principio

IMAGEN DEL AUTO ANTES Y DESPUÉS ESCALA DE GRISES

RESALTAR BORDES DETECTADOS

```
canny = cv2.dilate(canny,None,iterations=1)
```

- Con esta instrucción, resaltamos los bordes.

Aquí debemos ingresar la variable ***canny***, es la que detectó los bordes previamente

UNIRDE MANERA CONTINUA LOS CONTORNOS

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

- Con esta función, podemos unir todos los contornos de manera continua. Enviamos los bordes previamente identificados

CICLO PARA INGRESAR A CADA UNO DE LOS CONTORNOS

- Primero debemos de encontrar el área del contorno .

```
for c in cnts:
  # Encuentra el área del contorno
  area = cv2.contourArea(c)
```

Guardamos cada valor de ***contourArea*** enviando como valor de entrada "c".

```
x,y,w,h = cv2.boundingRect(c)
epsilon = 0.09*cv2.arcLength(c,True)
approx = cv2.approxPolyDP(c,epsilon,True)
```

Calculo del rectángulo del contorno.
Definimos nuevas variables, x,y,w,h que son las que reciben la salida de ***boundingRect***

IDENTIFICACIÓN DE LA PLACA

- Si la longitud de approx es igual a 4, se entiende que una placa tiene 4 bordes, y el área es mayor a > 9000, entonces se puede confirmar que es una placa.

```
if len(approx)==4 and area>9000:
```

Mostramos el área de la placa en pantalla

```
area= 9007.0
```

VERIFICACIÓN DE PLACA

- Para verificar efectivamente que es una placa, lo que debemos de tener en cuenta es la relación de proporciones. En este caso, la proporción ideal de la placa es 2.4

```
aspect_ratio = float(w)/h

if aspect_ratio>2.4:
```

MUESTRA DE DATOS DE LA PLACA

```
placa = gray[y:y+h,x:x+w]

# Se extrae el texto asociado a la placa utilizando Tesseract
text = pytesseract.image_to_string(placa,config='--psm 11')

# Se imprime la texto de la placa encontrada
print('PLACA: ',text)
```

- Se crea una variable de nombre placa, esta guardar la información de la ubicación dónde están localizados los bordes.
- Despues, con el uso de tesseract, identificamos los caracteres que se encuentran allí.

MOSTRAMOS LA PLACA EN MODO VENTANA

```
# Se muestra una imagen de la placa
cv2.imshow('PLACA',placa)
cv2.moveWindow('PLACA',780,10)
```

- Llamamos a la función ***cv2.imshow*** para mostrar la imagen que obtuvimos de la placa, luego movemos la ventana.

INSERTAR TEXTO A LA IMAGEN

```
cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)
cv2.putText(image,text,(x-20,y-10),1,2.2,(0,255,0),3)
```

- Creamos una nueva ventana con cv2, indicando la imagen, la posición que lleva esta y sumando las variables para así llegar a la placa de manera satisfactoria, se indica también el color que en este caso es verde y el grosor.

RESULTADO FINAL

```
cv2.imshow('Image',image)
cv2.moveWindow('Image',45,10)
cv2.waitKey(0)
```

- Se muestra la imagen ya con las letras y los bordes insertados, del mismo modo, se mueve la ventana