SISTEMA DE DETECCION DE PLACAS PARA PARQUEADERO

CRISTIAN ANDRES FIGUEROA CASTRO

PROYECTO FINAL

OBJETIVOS:

- Implementar código en OpenCV para la detección de placas de vehículos.
- Generar una base de datos tipo JSON para guardar datos del vehículo.
- Validar la información del vehículo a la entrada o salida.

Con este proyecto queremos realizar un sistema de backend para la detección de placas de los vehículos que ingresen a un establecimiento de parqueo, haciendo como primer paso el registro del vehículo y por un sistema que captura la placa del vehículo, analiza sus características y obteniendo sus números para el registro en la base de datos, guardando datos como placa y hora de entrada, además genera espacios vacíos para la hora de salida y cantidad de dinero pagado.

Toma y almacén de datos:

Para tomar los datos de la placa del vehículo como primer paso se debe llamar a la imagen a analizar, y definirle unos tamaños para realizar un correcto análisis.

```
def crearId(diccionario):

    id = list(diccionario.keys())[len(diccionario)-1]
    return (int(id) + 1)
def agregarPlaca(dic,placa,hora):
    if len(dic)==0:
        dic[1]= {
            "placa": placa,
            "HoraE":hora,
            "HoraS":"sin salida      ",
            "pago":"sin pagar"
        }
    else:
        id = crearId(dic)
        dic[id] = {
        "placa": placa,
        "HoraE":hora,
        "HoraS":"sin salida      ",
        "pago":"sin pagar"
        }
    return dic
```

```
def verCarro(carros,hora):
    listaCarros = "Codigo\t\tPlacas\t\tHora entrada\t\tHora
salida\t\t\tTiempo\t\tPago\n"
    x=datetime.strptime(hora, '%Y-%m-%d %H:%M')
    for carro in carros:
        y=datetime.strptime(carros[carro]["HoraE"], '%Y-%m-%d %H:%M')
        listaCarros += "\t" + str(carro) + "\t\t" +
carros[carro]["placa"]+ "\t\t" + carros[carro]["HoraE"]+\
        "\t" + carros[carro]["HoraS"]+"\t"+ str(x-y) + "\t\t"+
carros[carro]["pago"]+"\n"
    return listaCarros

frame = cv2.imread("carro2A.jpg")
hora_E= datetime.now().strftime('%Y-%m-%d %H:%M')
frame = cv2.resize(frame,(844,480),interpolation=cv2.INTER_CUBIC)
with open("parqueadero.json","r") as archivo:
    dicparqueadero=json.load(archivo)
archivo.close()
Ctexto=''
al,an,c=frame.shape
    #print(al,an,c)
x1=int(an/3)
x2=int(x1*2)

y1=int(al/3)
y2=int(y1*2)
cv2.rectangle(frame,(220,350),(650,450),(0,0,0),cv2.FILLED)
cv2.rectangle(frame,(x1,y1),(x2,y2),(0,255,0),2)
recorte=frame[y1:y2,x1:x2]
```

Imagen 1: Zona de análisis.

Con esta parte del código se limita el área de análisis en donde nosotros consideramos que estarán las placas, aquí se empieza a analizar por lo que se considera que el área que se quiere analizar es un área amarilla, lo que hacemos es analizar solo en los colores RGB y

tomamos nuestro contorno como también leemos nuestro archivo Json para recibir los datos.

```
nB=np.matrix(recorte[:,:,0])
nG=np.matrix(recorte[:,:,1])
nR=np.matrix(recorte[:,:,2])
Color=cv2.absdiff(nG,nB)
img_gris = cv2.cvtColor(recorte, cv2.COLOR_RGB2GRAY)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (13,5))
# crea imagen white hat
   _,umbral=cv2.threshold(Color,40,255,cv2.THRESH_BINARY)

contornos,_=cv2.findContours(umbral,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE
)
contornos = sorted(contornos,key=lambda x:
cv2.contourArea(x),reverse=True)
```

Imagen 2. Toma de colores amarillos.

Como se puede observar se toma únicamente el color amarillo siendo la mezcla de el verde y el rojo. Al momento de tener la zona de interés, se le aplican contornos y se analiza la imagen.

```
for contorno in contornos:
    area = cv2.contourArea(contorno)
    if area>500 and area<5000:
        x,y,ancho,alto=cv2.boundingRect(contorno)
        xpi=x+x1
        ypi=y+y1
        xpf=x+ancho+x1
        ypf=y+alto+y1

        cv2.rectangle(frame,(xpi,ypi),(xpf,ypf),(255,255,0),2)
```

```
        placa = frame[ypi:ypf,xpi:xpf]

        alp,anp,cp = placa.shape

        Mva = np.zeros((alp,anp))

        mBp = np.matrix(placa[:,:,0])
        mGp = np.matrix(placa[:,:,1])
        mRp = np.matrix(placa[:,:,2])

        for col in range(0,alp):
            for fil in range(0,anp):
                Max= max(mRp[col,fil],mGp[col,fil],mBp[col,fil])
                Mva[col,fil]=255-Max

        _,bin=cv2.threshold(Mva,150,255,cv2.THRESH_BINARY)

        bin = bin.reshape(alp,anp)
        bin = Image.fromarray(bin)
        bin = bin.convert("L")

        if alp >+ 36 and anp >=82:
            pytesseract.pytesseract.tesseract_cmd = r'C:\Program
Files\Tesseract-OCR\tesseract.exe'

            config = "--psm 1"
            texto = pytesseract.image_to_string(bin, config=config)

            if len(texto)>=7:
                Ctexto = texto
```

En esta parte del código como dije anterior mente se realiza el procesado de la imagen para detectar números y letras de las placas además de realizar un cuadro de marcación para la detección de la placa.

Imagen 3. Marco de placa.

Y para la deteccion de letras y numeros se utilizo la herramienta de codigo abierto Pytesseract que es es una herramiento de OCR (Reconocimiento Óptico de Caracteres), esta nos permite hacer el reconicimiento de la zona que ya reconocimos de manera sencilla y entregarnos una variable tipo string en Ctexto.

```
print(Ctexto)….> respuesta: EK0 731
```

Con los datos de la placa en tipo string ya queda solo enviar a la base de datos con la hora de ingreso y además visualizarla en la imagen, para eso hacemos lo siguiente.

```
`       placafinal = Ctexto.replace('-','')
        placaf = placafinal.replace(' ','')
        for i in dicparqueadero:
            if str(placaf[0:6])==(dicparqueadero[i]["placa"]):
                cont+=1
                if dicparqueadero[i]["pago"]!="sin pagar":
                    cont+=1
        if cont==1:
        cv2.putText(frame,'Vehiculo ya
        registrado',(250,430),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)

        cv2.putText(frame,Ctexto[0:7],(360,390),cv2.FONT_HERSHEY_SIMPLEX,1
        ,(0,255,0),2)
        elif cont==2:
        cv2.putText(frame,'Puede
        salir',(350,430),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2)

        cv2.putText(frame,Ctexto[0:7],(360,390),cv2.FONT_HERSHEY_SIMPLEX,1
        ,(0,255,0),2)
        else:
        cv2.putText(frame,'Ingrese',(360,430),cv2.FONT_HERSHEY_SIMPLEX,1,(
        0,255,0),2)

        cv2.putText(frame,Ctexto[0:7],(360,390),cv2.FONT_HERSHEY_SIMPLEX,1
        ,(0,255,0),2)

        dicparqueadero=agregarPlaca(dicparqueadero,placaf[0:6],hora_E)
        with open("parqueadero.json","w") as archivo:
        json.dump(dicparqueadero,archivo)
        archivo.close()
```

En muchos casos, al momento de obtener los datos de la placa del vehiculo, el sistema puede obtener otro tipo de datos, como puntos, espacios y líneas que pueden alterar la toma de datos y el envio a la base de datos por lo que se deciden remplazar y eliminar con la función .replace

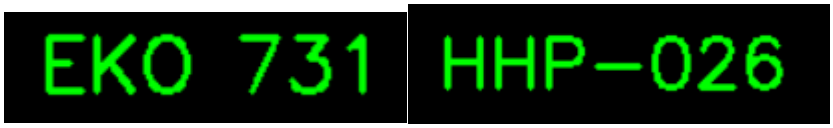

Imagen 4. Toma de caracteres de Pytesseract.

Luego de remplazar los caracteres y borrarlos, se tiene que analizar en la base de datos la condición del vehículo, esto se refiere a que si el vehículo ya se encuentra registrado en la base de datos no se hará un nuevo registro y se informara que ya se encuentra registrado, si este no es el caso el vehículo será registrado en la base de datos y se le dirá al cliente de que puede ingresar y el ultimo caso es cuando ya ha pagado, saldrá en color rojo el mensaje de que ya puede salir.

Imagen 5. Mensajes en patalla.

Ademas de imprimir la imagen, en la consola se imprime la cantidad de vehiculos que llamando a la funcion creada vercarro() y enviando los datos de el diccionario con la hora de ingreso, esto crea la siguiente tabla.

```
print(verCarro(dicparqueadero,hora_E))
cv2.imshow("vehiculos",frame)
cv2.imshow("muestra",Color)
    #cv2.imshow("prueba",contorno)
    #print(contornos)
cv2.waitKey(0)
# cierra todas las ventanas
cv2.destroyAllWindows()
```

Codigo	Placas	Hora entrada	Hora salida	Tiempo	Pago
1	UDU322	2023-05-12 17:08	sin salida	4:48:00	sin pagar
2	HHP026	2023-05-12 21:55	sin salida	0:01:00	sin pagar
3	EKO731	2023-05-12 21:56	sin salida	0:00:00	sin pagar

Imagen 6. Tabla cantidad de vehiculos.

Metodo de pago:

Para poder generar la salida en el registro de el vehiculo, es necesario realizar un pago y quedara registrado la hora de salida y cantidad de pago que se realizo, esto sera balidado por el sistema y verificara que registros tiene.

```
import os
import time
import json
from datetime import datetime

def vermenu():

    print("***************PAGO DE PARQUEADERO***************")
    print('''Seleccione alguna de las siguientes opciones:
          1. Pagar                2. Visualizar Horas
          0. Salir''')

def verCarro(carros,hora,placa):
    listaCarros = "Codigo\t\tPlacas\t\tHora entrada\t\tHora
salida\t\t\tTiempo\t\tPago\n"
    x=datetime.strptime(hora, '%Y-%m-%d %H:%M')
    for carro in carros:
        if carros[carro]["placa"]==placa:
            y=datetime.strptime(carros[carro]["HoraE"], '%Y-%m-%d %H:%M')
            listaCarros += "\t" + str(carro) + "\t\t" +
carros[carro]["placa"]+ "\t\t" + carros[carro]["HoraE"]+\
            "\t" + carros[carro]["HoraS"]+"\t"+ str(x-y) + "\t\t"+
carros[carro]["pago"]+"\n"
    return listaCarros
with open("parqueadero.json","r") as archivo:
    dicparqueadero=json.load(archivo)

vermenu()
opcmenu=input()
hora_E= datetime.now().strftime('%Y-%m-%d %H:%M')
x=datetime.strptime(hora_E, '%Y-%m-%d %H:%M')

while opcmenu != "0":
    if opcmenu =="1":
        placa=input("Digite la placa del vehiculo: ")
        for i in dicparqueadero:
            if dicparqueadero[i]["placa"]==placa and
dicparqueadero[i]["pago"]=="sin pagar":
                y=datetime.strptime(dicparqueadero[i]["HoraE"], '%Y-%m-%d
%H:%M')
                H=x-y
                Hpago=(H.seconds/60/60)
                print(H)
                if H.days==0:
                    if Hpago%1>0.10:
```

```
                        pago=(int(Hpago)+1)*2000
                        print(int(Hpago),"1")
                    else:
                        pago=int(Hpago)*2000
                        print(int(Hpago),"2")
                else:
                    if Hpago%1>0.10:
                        pago=(int(Hpago)+1)*2000
                        print(int(Hpago),"1")
                    else:
                        pago=int(Hpago)*2000
                        print(int(Hpago),"2")
                print("Vehiculo de placa:",placa)
                print("Debe pagar $",pago)
                print('''Seleccione alguna de las siguientes opciones:
                     1. Pagar                0.salir''')
                opc=input()
                if opc=="1":
                    print("********** Cancelando **********")
                    dicparqueadero[i]["pago"]=str(pago)

dicparqueadero[i]["HoraS"]=datetime.now().strftime('%Y-%m-%d %H:%M')
                    time.sleep(3)
                    print("********** GRACIAS POR SU VISITA **********")
                    with open("parqueadero.json","w") as archivo:
                        json.dump(dicparqueadero,archivo)
                    archivo.close()
                    time.sleep(3)
                elif opc=="0":
                    print("********** GRACIAS POR SU CONSULTA**********")
                    time.sleep(3)
                    break
            elif dicparqueadero[i]["placa"]==placa and
dicparqueadero[i]["pago"]!="sin pagar":
                print("\n")
                print("Vehiculo de placa:",placa)
                print("***Ya cancelo puede salir**")
                print("\n")
    elif opcmenu=="2":
        placa=input("Digite la placa del vehiculo: ")
        print(verCarro(dicparqueadero,hora_E,placa))
        time.sleep(5)
    vermenu()
    opcmenu=input()
```

Generamos 2 opciones 1 para el pago y otra para visualiazar, y en el mometo de pago se ve de la siquiente manera:

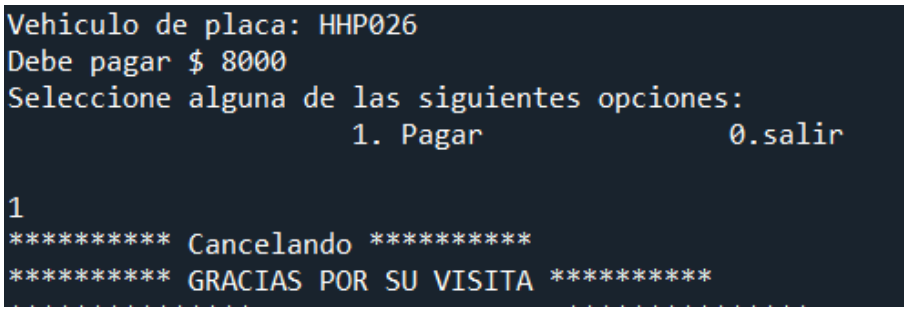

Imagen 7. Pago

Para realizar el pago, es necesario que el cliente introduzca la placa del vehículo y así el sistema le entregara la información de cuanto debe pagar para poder salir.

Conclusiones:

- La detección de placas de vehículos utilizando un programa para registrar en un archivo JSON y un sistema de pago es una solución efectiva para automatizar el proceso de registro y pago en parqueaderos. Con este sistema, los usuarios pueden acceder a los estacionamientos sin la necesidad de tener que interactuar con un operador.
- Al automatizar el proceso de registro y pago, se minimiza el riesgo de fraude y de errores humanos en el manejo de la información.