VISION COMPUTACIONAL - TERCER PARCIAL

DANIEL ALEJANDRO CANO MARTINEZ - 277832

PLANTEAMIENTO DEL PROBLEMA

Implementar un detector de objetos utilizando el módulo DNN de OpenCV.

a. Investigar sobre el dataset de Imágenes de objetos MS COCO. Decir quién y cuándo lo creó. Qué contiene, cuantas clases y número de imágenes. Agregar para que tipo de problemas se ha utilizado y el enlace de donde se puede bajar públicamente.

b. Utilizar la biblioteca OpenCV para implementar un programa que detecte objetos que pertenezcan a alguna(s) de clase(s) del dataset MS COCO. Para el programa que usted realice, puede utilizar cualquiera de las DNN y frameworks soportados que vienen en la documentación de OpenCV. Pero para obtener un buen desempeño, debería asegurarse que la DNN que elija está ya pre-entrenada con el dataset MS COCO. En cualquier caso, deberá también entregar una breve semblanza de la DNN que eligió para su trabajo (nombre de la DNN, quien la propuso, descripción breve de la arquitectura, y para qué problema se propuso).

c. Para probar su implementación, debe tomar una foto (o video, si su capacidad de cómputo lo permite) usted mismo donde aparezcan objetos de las clases de MS COCO. Si es muy difícil hacer una foto o video con las clases que usted eligió para probar, puede utilizar alguna imagen o video del dominio público y poner la referencia de donde lo obtuvo. El caso es que NO utilice fotos o videos de este dataset MS COCO.

DESCRIPCIÓN DE LA SOLUCIÓN

El dataset MS COCO (Common Objects in Context) es un dataset de captura, clasificación, reconocimiento, detección y segmentación de imágenes, este contiene alrededor de 330 mil imágenes y mas de 2 millones de instancias en 80 categorías de objetos, con 5 capturas por imagen y 250 mil personas con puntos clave.

Un modulo DNN permite cargar modelos previamente entrenados de los marcos de aprendizaje mas populares. MobileNet es una red de detección de caja múltiple (SSD) de disparo único destinada a realizar la detección de objetos.

El ejercicio básicamente se toma directamente del enlace proporcionado, los cambios son prácticamente mínimos, Para comenzar se realiza la lectura del archivo de clases y se guarda como un arreglo usando el método split() que nos separara los elementos dado un separador especificado, en este caso por cada salto de línea.

```
with open('./MSCOCO/object_detection_classes_coco.txt', 'r') as f:
    class_names = f.read().split('\n')
```

Se obtienen colores aleatorios dadas las clases de forma aleatoria para el dibujado de los bordes que señalaran los objetos detectados.

```
COLORS = np.random.uniform(0,255,size=(len(class_names), 3))
```

Se procede a cargar la DNN de MSCOCO, para ello brindamos el archivo que contiene el modelo de la dnn y su configuración como parámetros del método “dnn.readNet”.

```
model = cv.dnn.readNet(model='./MSCOCO/frozen_inference_graph.pb', config='./MSCOCO/ssd_mobilenet_v2_coco_2018_03_29.pbtxt.txt', framework='TensorFlow')
```

Se prepara la salida de video, dando ruta y nombre, así como formato, frames y tamaño de pantalla.

```
cap = cv.VideoCapture('./Videos de Entrada/1.mp4')

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

out = cv.VideoWriter('./Videos de Salida/1.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))
```

Se añadió un contador para poder hacer el cambio de video de entrada, esto es una función particular de mi programa, por lo que no se encuentra en el ejercicio de OpenCV.

```
contador = 0
```

Abrimos un ciclo para poder observar la detección de objetos en el video de entrada, primeramente, se leerá el frame de la captura, si este no retorna información, entonces se aumentará el contador previamente establecido.

```
while cap.isOpened():
    ret, frame = cap.read()

    if not(ret):
        contador = contador + 1
```

Si el retorno es exitoso se toma el frame y se obtienen los tamaños.

```
if ret:
    image = frame
    image_height, image_width, _ = image.shape
```

Se crea un blob dado el frame capturado. Y se le proporciona el resultado al modelo pre-entrenado. Posteriormente se genera un arreglo con las coincidencias encontradas.

```
blob = cv.dnn.blobFromImage(image = image, size = (300,300), mean = (104,117,123), swapRB = True)

model.setInput(blob)
output = model.forward()
```

Se itera para cada detección en el arreglo resultante y se toma el porcentaje de certeza analizado.

```
for detection in output [0,0,:,:]:
    confidence = detection[2]
```

Posteriormente se compara y si se supera el 40% de certeza se dibuja el rectángulo rodeando el objeto detectado con el color correspondiente que se generó al inicio y se escribe el frame en el archivo de salida.

```
if confidence > .4:
    class_id = detection[1]

    class_name = class_names[int(class_id)-1]
    color = COLORS[int(class_id)]

    box_x = detection[3] * image_width
    box_y = detection[4] * image_height

    box_width = detection[5] * image_width
    box_height = detection[6] * image_height

    cv.rectangle(image, (int(box_x), int(box_y)),(int(box_width), int(box_height)), color, thickness = 2)
    out.write(image)
    cv.putText(image, class_name, (int(box_x), int(box_y - 5)), cv.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
```

Finalmente se muestra en pantalla la imagen resultante

```
cv.imshow('image', image)
```

Adicionalmente se añadió la funcionalidad para detectar teclas presionadas, siendo capaz de cambiar de entrada presionando 1,2 o 3 y de terminar el programa presionando “Q”.

```
if cv.waitKey(1) & 0xFF == ord('q'):
    break

elif cv.waitKey(1) & 0xFF == ord('1'):
    contador = 0
    cap = cv.VideoCapture('./Videos de Entrada/1.mp4')
    out = cv.VideoWriter('./Videos de Salida/1.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))

elif cv.waitKey(1) & 0xFF == ord('2'):
    contador = 1
    cap = cv.VideoCapture('./Videos de Entrada/2.mp4')
    out = cv.VideoWriter('./Videos de Salida/2.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))

elif cv.waitKey(1) & 0xFF == ord('3'):
    contador = 2
    cap = cv.VideoCapture('./Videos de Entrada/3.mp4')
    out = cv.VideoWriter('./Videos de Salida/3.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))
```

El contador previamente mencionado sirve para cambiar automáticamente de escena, dadas las siguientes condiciones.

```
elif not(ret) and contador == 1:
    cap = cv.VideoCapture('./Videos de Entrada/2.mp4')
    out = cv.VideoWriter('./Videos de Salida/2.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))

elif not(ret) and contador == 2:
    cap = cv.VideoCapture('./Videos de Entrada/3.mp4')
    out = cv.VideoWriter('./Videos de Salida/3.mp4', cv.VideoWriter_fourcc(*'mp4v'), 30, (frame_width, frame_height))

else :
    break
```

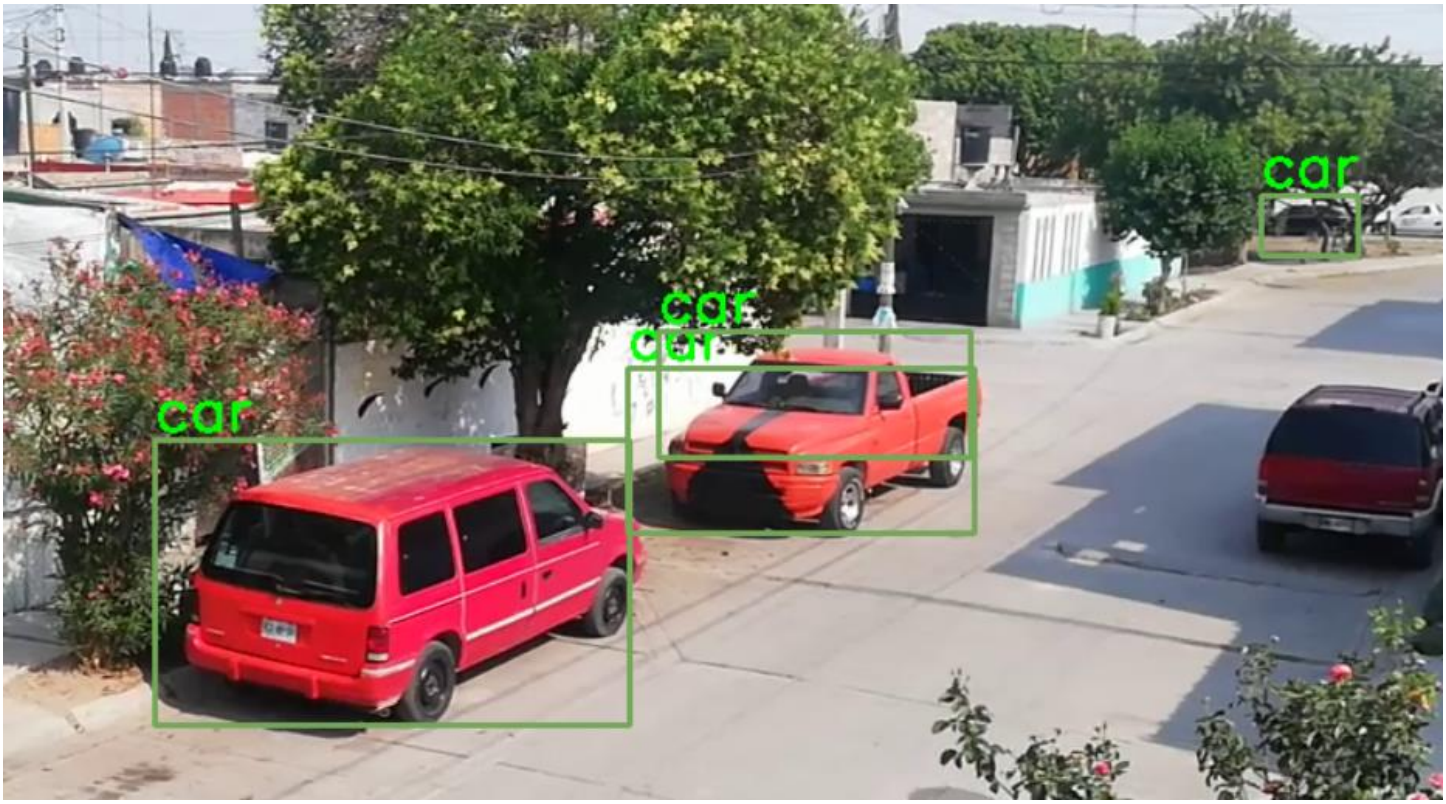
Por ultimo las instrucciones para terminar la captura de video y cerrar las ventanas.

```
cap.release()
cv.destroyAllWindows()
```

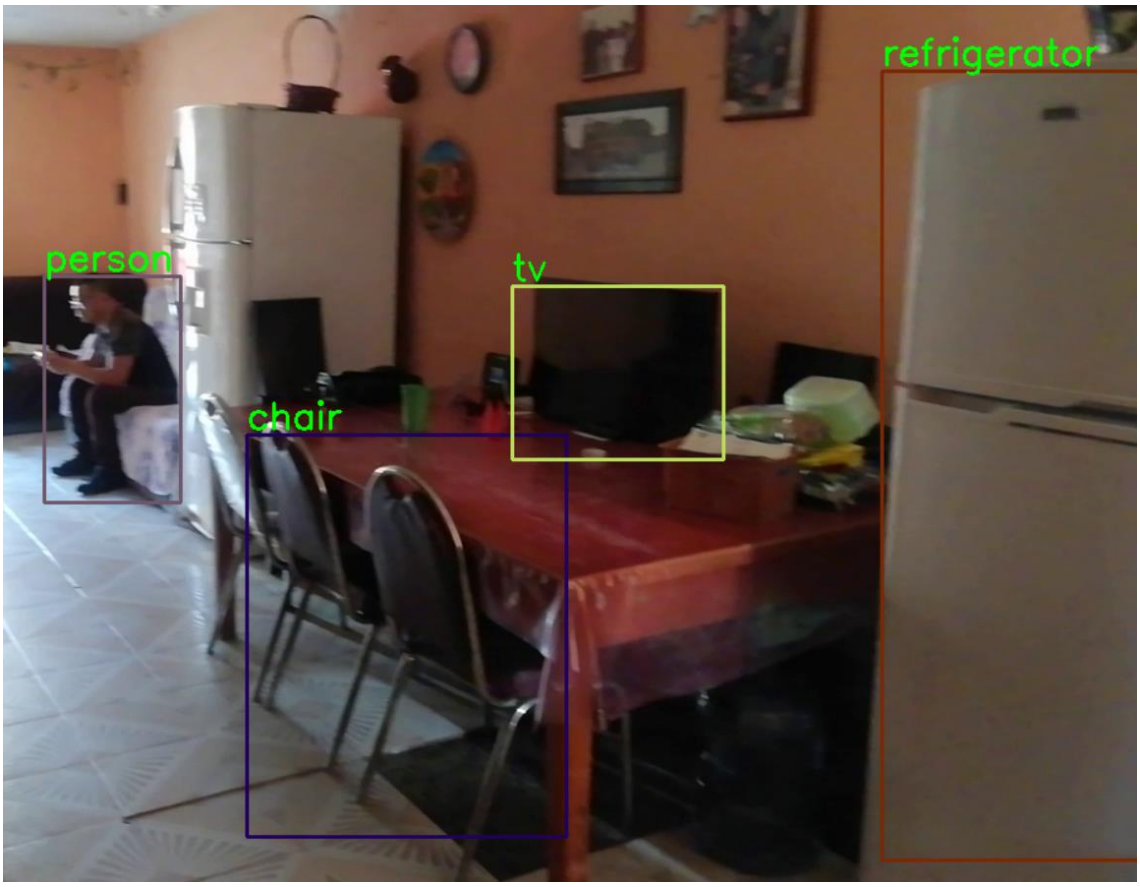
DESCRIPCIÓN DE RESULTADOS

Durante la escritura del código se testeo con la cámara del dispositivo para determinar si todo funcionaba correctamente, al terminar decidí grabar diversos videos de entrada para detectar diferentes objetos basado en la premisa de que muchas de las clases en el archivo corresponden a objetos cotidianos.

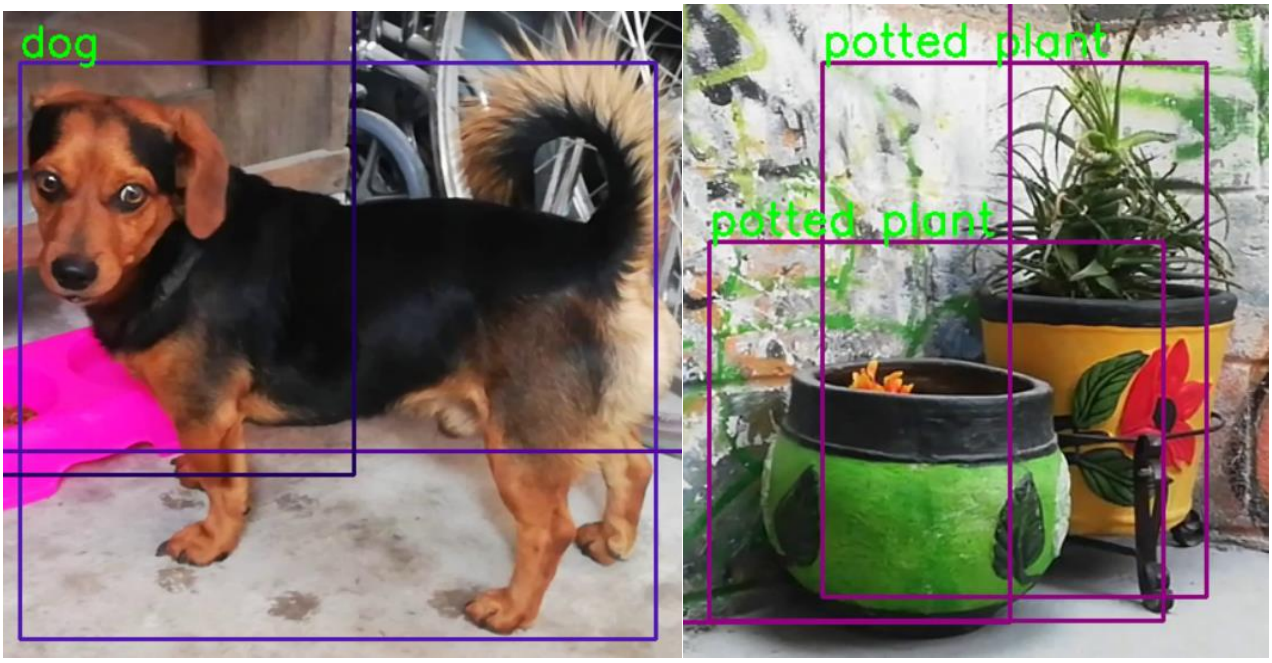
El video 1 muestra una toma de la calle, la detección se limita a automóviles.



En el video 2 tenemos una toma del interior de una casa, con diversos objetos a detectar como lo son, sillas, refrigeradores, televisores y hasta personas.



En el tercer y ultimo video se muestra la toma de un patio, donde detectamos un perro y plantas en macetas.



VISION COMPUTACIONAL - TERCER PARCIAL

UASLP Universidad Autónoma de San Luis Potosí

FACULTAD DE INGENIERÍA Ingeniería en Computación

DISCUCIÓN

El proyecto tiene un rendimiento descente, en ocaciones trabaja perfectamente con una iluminacion y angulo apropiado, sin embargo en muchas ocaciones clasifica objetos de forma erronea e inclusive clasifica secciones vacias como paredes o conjuntos de objetos como un objeto unico. Esto se puede entender debido a las limitaciones tanto de hardware que se usan o inclusive la DNN como tal.

CONCLUSIÓN

Es un proyecto interesante a implementar, resulta divertido experimentar con diversas entradas y ver los resultados generados, inclusive durante los errores resulta cómico. El proyecto a nivel código esta bien documentado y es bastante légible por lo que se puede entender muy facilmente la compocición del mismo. En general creo que es una buena solución para realizar la detección de objetos en proyectos mas grandes, pues con un equipo adecuado se pueden obtener resultados bastante confiables. Además que es el proceso más acertado a utilizar tomando en cuenta los métodos de detección vistos a lo largo del curso.