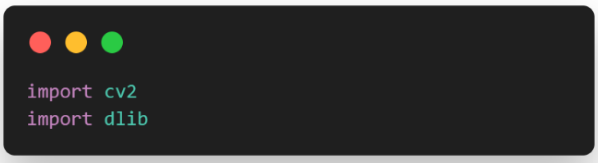
SEGUIMIENTO DE OBJETOS EN MOVIMIENTO CON VISIÓN ARTIFICIAL

Tal vez te has encontrado en ocasiones con videos en los cuales se muestra un rectángulo siguiendo un carro, una persona o cualquier objeto en movimiento, y si te has preguntado ¿cómo se logra esto? Pues, en este artículo te mostraré una forma de seguir objetos en movimiento con visión artificial. Una vez entiendas cómo funciona esto podrás seguir cualquier cosa que se mueva de forma fácil.

Para este código vamos a usar dos librerías, las cuales son Open CV y DLIB; importamos estas librerías primero que todo, y una vez hecho esto podemos comenzar.



```
import cv2
import dlib
```

Ahora vamos a inicializar nuestra captura de video con la función .videoCapture() y como parámetro el nombre del video que usaremos y tendremos guardado en una carpeta con el código, en caso de hacerlo con una cámara en tiempo real usaremos la variable de la cámara.

```
Cap = cv2.VideoCapture("video.mp4")
```

Antes de continuar, declararemos una variable llamada, en este caso, Trackers, la cual será una lista vacía, que nos servirá más adelante para obtener algunos valores importantes. Una vez teniendo declarada esta variable, inicializaremos el ciclo while, y leeremos nuestros frames con la función .read().

```
Trackers=[]
while True:
    Rec, Frames= Cap.read()
```

Aplicaremos un redimensionamiento a nuestros frames, ya que el video es demasiado grande, por lo cual usaremos la función .resize(), como parámetros la variable que queremos redimensionar y los valores de ancho y alto en píxeles, a la cual queremos hacer el redimensionamiento.

Una vez teniendo este redimensionamiento, aplicaremos un cambio en la escala de colores, por el hecho de que OpenCV lee los frames como BGR por defecto, y nosotros los

necesitaremos en RGB para cuando queramos obtener las posiciones más adelante, por eso usaremos la función .cvtColor() y para los parámetros la variable a cambiar y el tipo de cambio.

```
Frames = cv2.resize(Frames,(800,480))
rgb_Frame = cv2.cvtColor(Frames,cv2.COLOR_BGR2RGB)
```

Para poner en uso la lista vacía que habíamos declarado al inicio, crearemos un ciclo for con el cual obtendremos las posiciones.

```
for Tracker in Trackers:
    Tracker.update(rgb_Frame)
    Position=Tracker.get_position()
    x1=int(Position.left())
    y1=int(Position.top())
    x2=int(Position.right())
    y2=int(Position.bottom())
```

¿Qué es lo que hace esta ciclo for? Pues bien, primero tenemos la variable Tracker la cual estará iterando en Trackers, variable que declaramos como lista vacía.

A esta variable la actualizaremos mediante la función .update() y como parámetro la variable con el cambio de escala de BGR a RGB, ahora ¿por qué hacemos esta actualización? Pues esto es porque tenemos un video con movimiento y al cambiar constantemente de posiciones los objetos, necesitamos tener la posición actualizada de esos cambios para una mejor estimación de las posiciones actuales.

Después de esto, la función .get_position() nos dejará obtener la posición actualizada que después será colocada en las coordenadas con las variables x1, y1, x2 y y2; estas variables nos darán las coordenadas gracias a las funciones .left(), .top(), .right() y .bottom() respectivamente.

Teniendo estas posiciones, crearemos un recuadro que será el que se mostrará una vez estemos siguiendo el objeto en movimiento, este se creará con la función .rectangle() y como parámetros lo siguiente:

- Imagen en la que mostraremos el rectángulo.
- Las coordenadas en *x* y *y* de inicio.
- Las coordenadas en *x* y *y* del final.

- Color del cual se mostrará el rectángulo.
- Grosor del rectángulo.

Para los valores de las coordenadas usaremos las obtenidas en el ciclo for.

```
cv2.rectangle(Frames,(x1,y1),(x2,y2),(0,255,0,2),3)
```

Ahora, mostraremos un texto con las coordenadas en donde está el objeto seguido, para esto crearemos una variable con la posición en donde queremos mostrar el texto, en este caso tomamos como referencia las variables x1 y y1 restándole a esta última un valor de 30.

```
Locate_Text = (int(x1)), (int(y1 - 30))
```

Teniendo esto, vamos a crear otra variable con el texto que queremos mostrar, verás que usamos un método .format(), este método nos ayuda a insertar una variable como texto gracias al uso de una lista y las llaves "{}", dentro de estas llaves se insertan esas variables entregadas como parámetros a el método.

```
Text = "OBJECT TRACKED IN [{}, {}]".format(x1, y2)
```

Ya solo nos faltaría mostrará ese texto, y esto se hará con la función .putText() y para los parámetros usaremos los siguientes elementos:

- Lugar en el cual se mostrará el texto.
- El texto a mostrar.
- La localización de donde se mostrará el texto.
- El tipo de fuente del texto.
- El tamaño del texto.
- El color del texto.
- Grosor del texto.

```
cv2.putText(Frames, Text, Locate_Text, cv2.FONT_HERSHEY_SIMPLEX, .5, (0, 0, 255), 1)
```

Ahora crearemos un comando mediante el teclado, para esto usaremos la función .waitKey(), crearemos una variable en la cual se almacenará el tiempo de lectura, una vez tengamos esa variable crearemos un condicional if, en el cual indicaremos que, si se presiona la letra “s”, ya sea mayúscula o minúscula, se empiece el proceso de selección para el área de interés, esto mediante la función .selectROIs(), la cual tendrá como parámetros:

- Nombre de la ventana
- Lugar en el que se mostrara.
- El parámetro booleano showCrosshair.
- El parámetro booleano fromCenter.

Si el parámetro showCrosshair es True, muestra el rectángulo de seguimiento con una especie de rejillas, de ser False solo se mostrará el rectángulo de seguimiento.

Para el caso de fromCenter, si es True la selección se hará de forma central extendiéndose hacia ambos lados, en caso de ser False será conforme se vaya extendiendo la selección.

```
t=cv2.waitKey(30)
if t == ord('S') or t== ord('s'):
    Rois=cv2.selectROIs("VIDEO",Frames,showCrosshair=True,fromCenter=False)
```

Ahora, tendremos un ciclo for en donde la variable Roi estará iterando en la variable Rois (variable que contiene la selección), cada iteración de Roi obtiene las coordenadas de la selección por lo cual las variables x1, y1, width y heigth están igualadas a la variable Roi. Para obtener x2, que es el segundo punto del rectángulo utilizaremos la coordenada de x1 más width y para y2, lo contrario, utilizaremos y1 más heigth.

```
for Roi in Rois:
    x1,y1,width,height=Roi
    x2=x1+width
    y2=y1+height
```

Para terminar con el ciclo for, usaremos la librería dlib, esta librería contiene un algoritmo de seguimiento de objetos que es la que usaremos, primero inicializamos la función .correlation_tracker(), esta función se encarga de comparar la similitud de la selección del área de interés con áreas de los fotogramas sucesivos. Teniendo esto, crearemos un rectángulo a partir de la función .rectangle() y las variables x1, x2, y1 y y2, este rectángulo será el encargado de comparar las similitudes.

Continuando con el ciclo, inicializaremos el seguimiento con la función .start_track(), como parámetros usaremos los frames en RGB y el rectángulo que creamos con la función .rectangle(), y ya para finalizar solos nos queda agregar lo que hicimos en el ciclo for a nuestra variable Trackers, esto gracias a la función .append().

```
Tkr=dlib.correlation_tracker()
Rect=dlib.rectangle(x1,y1,x2,y2)
Tkr.start_track(rgb_Frame,Rect)
Trackers.append(Tkr)
```

Teniendo esto, podemos ahora si mostrar nuestro video y empezar el seguimiento, usaremos la función .imshow(), con el nombre que tendrá la ventana y la variable que mostraremos como parámetros.

```
cv2.imshow("VIDEO",Frames)
```

Crearemos nuestro comando de salida, en este caso será al presionar la tecla “q”, ya sea minúscula o mayúscula.

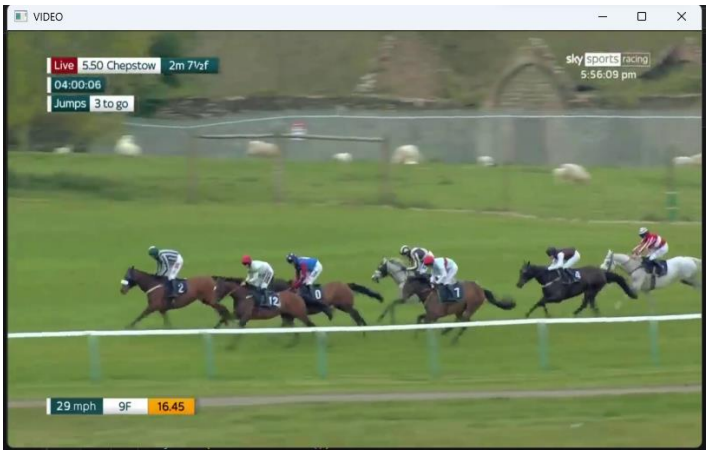
```
if t == ord('Q')or t== ord('q'):
    break
```

Por último, borraremos la video captura con la función .release(), y destruiremos las ventanas con .destroyAllWindows().

```
Cap.release()
cv2.destroyAllWindows()
```

RESULTADOS OBTENIDOS

La primer imagen es el video comenzando, en este video es donde vamos hacer los comandos que hemos programado en el código.



Si presionamos la tecla “s”, el video se pausará y podremos indicar el área a seguir, después de seleccionar el área presionamos la tecla “Enter” para continuar con el video y ver el texto con las coordenadas.

Este mismo proceso se puede repetir y así tener dos o más áreas seleccionadas y seguidas.

Si llegaste hasta esta parte te agradezco el tiempo que me regalaste y espero te haya ayudado con tus dudas o inquietudes de aprender más, esto solo es un pequeño fragmento de explicación si quieres seguir aprendiendo más puedes visitar las siguientes páginas que te ayudaran a seguir aprendiendo y mejorando en el tema de la visión artificial y Python.

1.- https://omes-va.com/

2.- https://www.pythonpool.com/

3.- https://hetpro-store.com/TUTORIALES/

4.- https://learnopencv.com/

Lo principal es la sabiduría; adquiere sabiduría, Y con todo lo que obtengas adquiere inteligencia. Proverbios 4:7