011_P11_Equals Rotation

- Inicie sesión en mi Github

-Creé un nuevo archivo para la práctica 11en Python

-Hice ya lo que indicaba de la práctica 11 de Python “Equals Rotation OpenCV Python”

-Realice el código el cual el principal objetivo Igualdades con rotación y reducción de fondo

Objetivo: De la imagen deseada encontrar las similitudes en otra imagen.

Objetivo 2: En video poder extraer el fondo de la imagen mediante la detección de movimiento.

-Ejecuté primero el código de las similitudes de la otra imagen y tomé de ejemplo unas imágenes parecidas que agregué la cual hizo un diagrama con las similitudes de ambas imágenes.

-Ya tomé su respectiva evidencia.

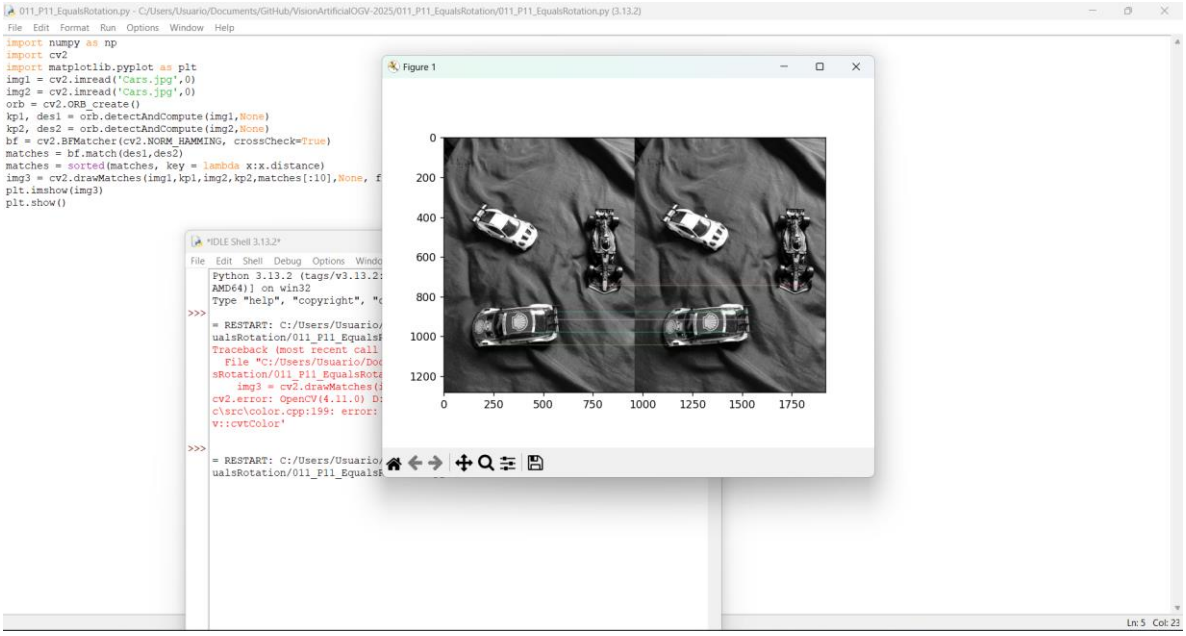
-Ahora hice el segundo código con ayuda de chat, ya con esta tenía que detectar el movimiento en video y generaba 2 ventanas con el movimiento y me daba el contorno de lo que detectaba.

-Ya tomé las evidencias y del código los dividí.

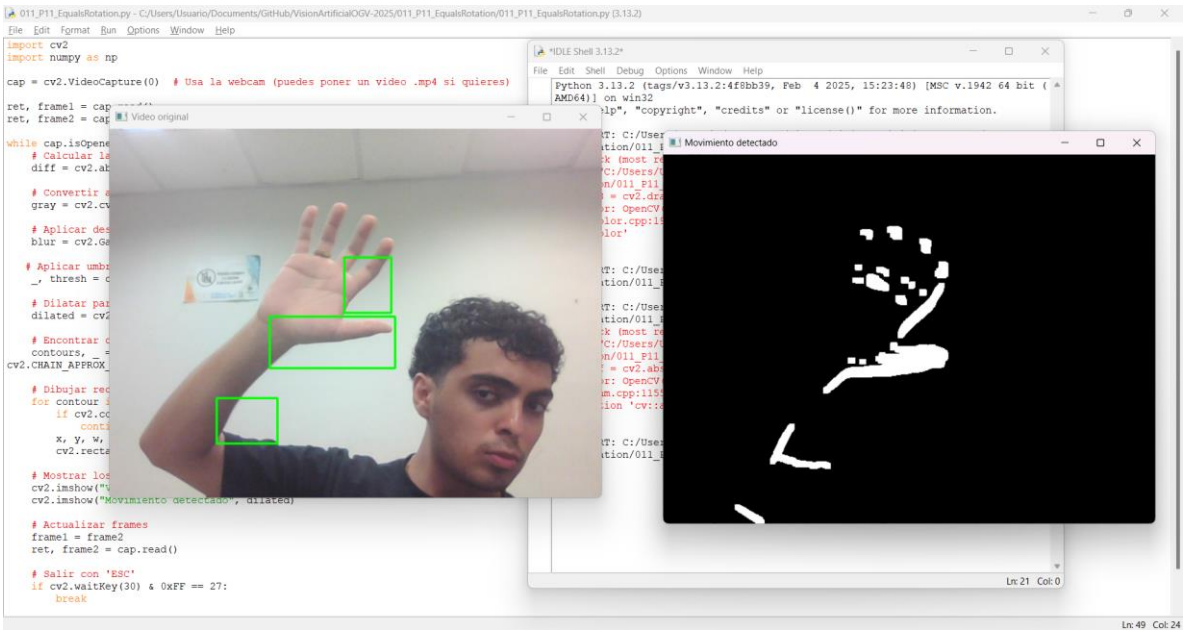
-Ya para finalizar, creé el reporte. -Subí todos los archivos a mi GithubDesktop sobre la práctica 11 de Python “Equals Rotation”.

IMAGEN DE EJEMPLO:

PRUEBA 1:



PRUEBA 2:



MI CÓDIGO 1:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
img1 = cv2.imread('Cars.jpg',0)
img2 = cv2.imread('Cars.jpg',0)
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1,des2)
matches = sorted(matches, key = lambda x:x.distance)
img3 = cv2.drawMatches(img1,kp1,img2,kp2,matches[:10],None, flags=2)
plt.imshow(img3)
plt.show()
```

MI CÓDIGO 2:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)  # Usa la webcam (puedes poner un video .mp4 si quieres)

ret, frame1 = cap.read()
ret, frame2 = cap.read()

while cap.isOpened():
    # Calcular la diferencia absoluta entre dos frames
    diff = cv2.absdiff(frame1, frame2)

    # Convertir a escala de grises
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)

    # Aplicar desenfoque para reducir ruido
    blur = cv2.GaussianBlur(gray, (5,5), 0)

   # Aplicar umbral para binarizar el movimiento
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)

    # Dilatar para rellenar huecos
    dilated = cv2.dilate(thresh, None, iterations=3)

    # Encontrar contornos (zonas en movimiento)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    # Dibujar rectángulos en zonas de movimiento
    for contour in contours:
        if cv2.contourArea(contour) < 700:  # Ignorar movimiento pequeño
            continue
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Mostrar los resultados
    cv2.imshow("Video original", frame1)
    cv2.imshow("Movimiento detectado", dilated)

    # Actualizar frames
    frame1 = frame2
    ret, frame2 = cap.read()

    # Salir con 'ESC'
    if cv2.waitKey(30) & 0xFF == 27:
        break
```

ENLACE DE MI REPOSITORIO DE GITHUB:

https://github.com/Ing-OscarValencia/VisionArtificialOGV-2025.git