**Centro de Enseñanza Técnica Industrial**

**Visión Artificial.**

**Ingeniería Mecatrónica.**

**Practica# 11: igualdades con rotación y reducción de fondo.**

**Platel: CETI Colomos.**

**Nombre: Ruiz Macías Luis Enrique - 21310196**

**Grado/Grupo: 6°G**

**Objetivo1:** De la imagen deseada encontrar las similitudes en otra imagen.

**Objetivo2:** En VIDEO poder extraer el fondo de la imagen mediante la detección de movimiento.

En el presente código tenemos dos funcionalidades que para su practicidad se harán por separado, un parte toma una imagen y de esta misma saca las coincidencias de la misma, esto por vecindad y en la otra parte se presentara un video original y otro video con la función de por medio de detectar el movimiento se mostrar únicamente el primer plano, lo que hará que solo se presente lo que se mueve en un primer plano, siendo el fondo estático y mostrándose en negro, al igual que las zonas que no se mueven en el plano.

A continuación, se presenta el Código:

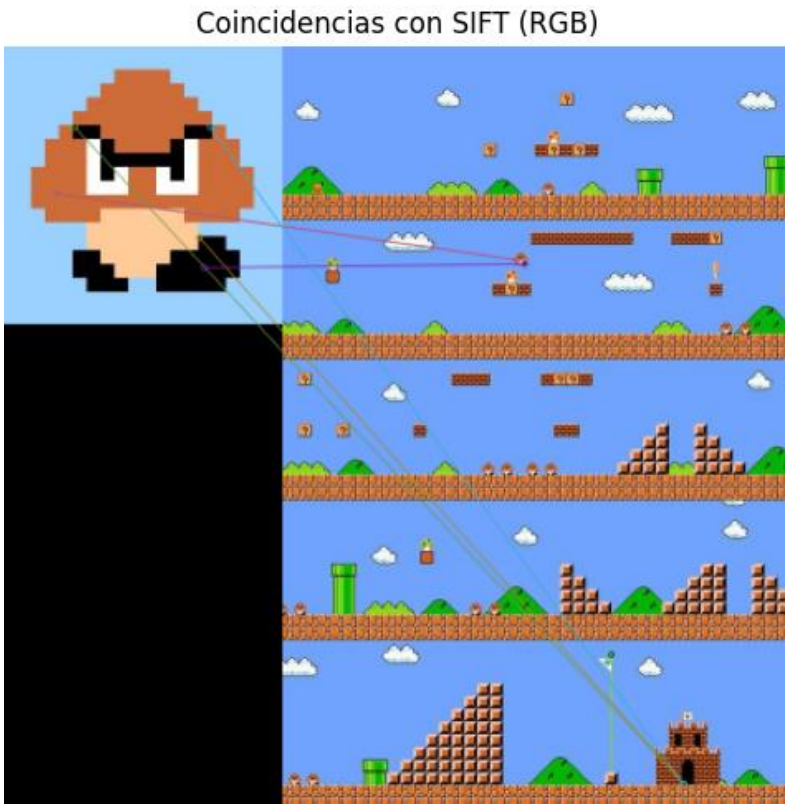
```
import cv2

import numpy as np

import matplotlib.pyplot as plt


# === FUNCIONALIDAD 1: Buscar similitudes con SIFT ===

def buscar_similitudes(template_path, escena_path):

    # Cargar imágenes en color (para mostrar en RGB)

    color_img1 = cv2.imread(template_path, cv2.IMREAD_COLOR)

    color_img2 = cv2.imread(escena_path, cv2.IMREAD_COLOR)


    if color_img1 is None:

        print(f"No se pudo cargar: {template_path}")

        return

    if color_img2 is None:

        print(f"No se pudo cargar: {escena_path}")

        return
```

```
# Convertir a escala de grises para SIFT
gray_img1 = cv2.cvtColor(color_img1, cv2.COLOR_BGR2GRAY)
gray_img2 = cv2.cvtColor(color_img2, cv2.COLOR_BGR2GRAY)

# Inicializar SIFT
sift = cv2.SIFT_create()

# Detectar keypoints y descriptores
kp1, des1 = sift.detectAndCompute(gray_img1, None)
kp2, des2 = sift.detectAndCompute(gray_img2, None)

# FLANN matcher
index_params = dict(algorithm=1, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)

# Test de Lowe
good = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good.append(m)

print(f"Coincidencias válidas encontradas: {len(good)}")

# Dibujar coincidencias sobre imágenes en COLOR
```

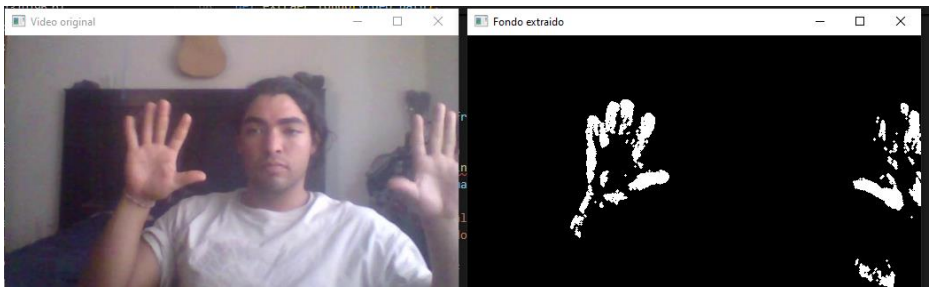
```
    result = cv2.drawMatches(color_img1, kp1, color_img2, kp2, good, None,
                             flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    # Redimensionar resultado
    scale_percent = 50
    width = int(result.shape[1] * scale_percent / 100)
    height = int(result.shape[0] * scale_percent / 100)
    resized_result = cv2.resize(result, (width, height), interpolation=cv2.INTER_AREA)

    # Convertir a RGB
    rgb_result = cv2.cvtColor(resized_result, cv2.COLOR_BGR2RGB)

    # Mostrar usando matplotlib
    plt.figure(figsize=(12, 6))
    plt.imshow(rgb_result)
    plt.title("Coincidencias con SIFT (RGB)")
    plt.axis('off')
    plt.show()

    # Guardar imagen resultante (opcional)
    cv2.imwrite('resultado_RGB.jpg', resized_result)
    print("Resultado guardado como 'resultado_RGB.jpg'.")

# === FUNCIONALIDAD 2: Extraer fondo en video ===
def extraer_fondo(video_path):
    cap = cv2.VideoCapture(video_path)
```

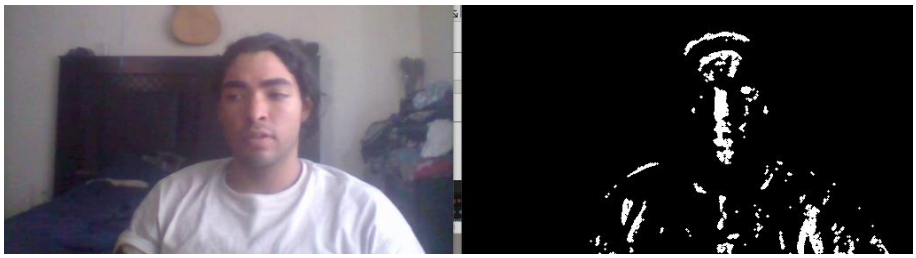
```
    subtractor = cv2.createBackgroundSubtractorMOG2(history=200,
varThreshold=25, detectShadows= False)

    if not cap.isOpened():
        print(f"No se pudo abrir el video: {video_path}")
        return

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        mask = subtractor.apply(frame)

        # Limpieza con morfología
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

        cv2.imshow('Video original', frame)
        cv2.imshow('Fondo extraido', mask)

        if cv2.waitKey(30) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

```
# === MENÚ PRINCIPAL ===
if __name__ == '__main__':
    print("1. Buscar similitudes entre imágenes (SIFT)")
    print("2. Extraer fondo de un video")
    opcion = input("Elige una opción (1 o 2): ")

    if opcion == '1':
        buscar_similitudes(
            r'C:\Users\USUARIO\Documents\GitHub\Vision artifical 2025\Igualdad rotacion y reduccion fondoP11\template.jpg',
            r'C:\Users\USUARIO\Documents\GitHub\Vision artifical 2025\Igualdad rotacion y reduccion fondoP11\escena.jpg'
        )
    elif opcion == '2':
        extraer_fondo(
            r'C:\Users\USUARIO\Documents\GitHub\Vision artifical 2025\Igualdad rotacion y reduccion fondoP11\video.mp4'
        )
    else:
        print("Opción no válida.")
```




Como vemos los diferentes puntos de pixel que tienen una vecindad similar a la que se presenta a la imagen muestran pintos de coincidencia, esto por tonalidad y posición, siendo la primera parte la cual en la que ambas imágenes se procesan para hacerlas a escala de grises, y al final se muestran las conciencias para después ser procesadas y al final ser mostradas en color con la ayuda de MATLIB





Y en esta ultima parte se muestra un video normal y corriente formato mp4 en el cual se presenta mi persona moviéndose, con el fondo estático.
El propósito era extraer el fondo, y esto se logró mediante la captura de movimiento, de hecho, en las capturas presentes se pueden mostrar las diferencias de que es lo que se esta moviendo en la imagen, siendo así que las zonas en primer plano solo son as que se mueven, si pasa un tiempo sin moverse se pierden.