Reporte de Práctica: Eliminación de Fondo en Tiempo Real con OpenCV

Objetivo de la práctica

Implementar un sistema de eliminación de fondo en tiempo real mediante la técnica de sustracción de fondo utilizando el modelo BackgroundSubtractorMOG2 de OpenCV. El objetivo es identificar y aislar los objetos en movimiento dentro de una escena capturada por la cámara.

Desarrollo del código

1. **Captura de video** con cv2.VideoCapture(0) desde la cámara.
2. **Creación del sustractor de fondo** con parámetros:
    - history=500: Cuántos frames se usan para el modelo de fondo.
    - varThreshold=50: Sensibilidad al cambio.
    - detectShadows=False: Desactiva la detección de sombras.
3. **Obtención de la máscara binaria** (blanco = movimiento).
4. **Limpieza del ruido** con operación morfológica **open**.
5. **Conversión a imagen de 3 canales** para poder aplicar bitwise_and.
6. **Visualización en tiempo real** del original y la imagen con fondo eliminado.
7. **Salida del programa** con la tecla q.

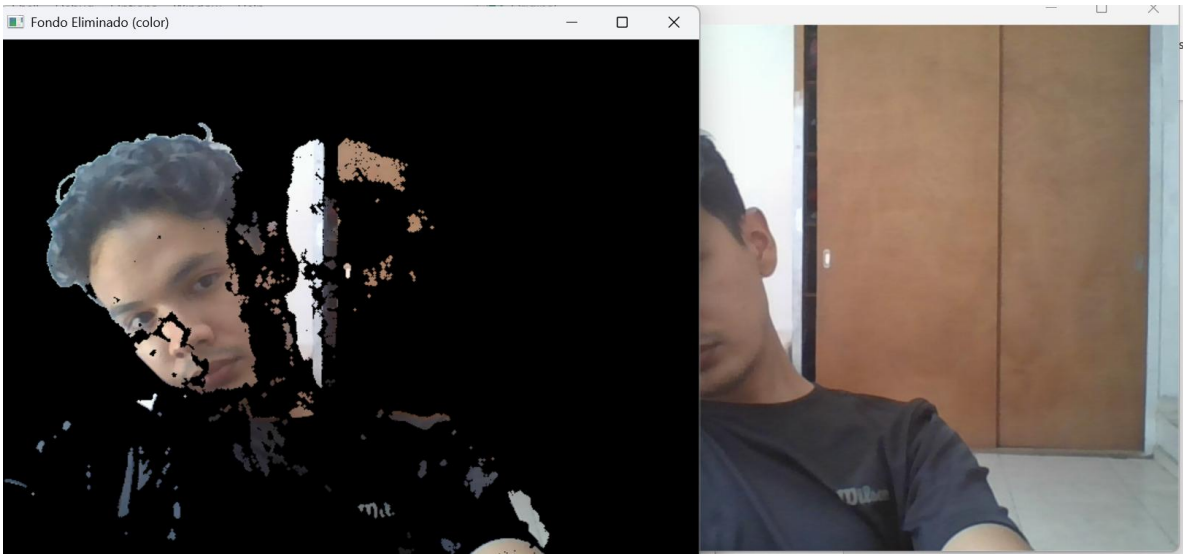
```
import cv2

import numpy as np


# Captura de video

cap = cv2.VideoCapture(0)


# Subtractor de fondo MOG2

fgbg = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=50, detectShadows=False)
```

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

    # Obtener la máscara del fondo (blanco = movimiento)
    fgmask = fgbg.apply(frame)

    # Aplicar filtro morfológico para limpiar ruido
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)

    # Convertir la máscara binaria en máscara de 3 canales para usarla con el frame en color
    mask_3ch = cv2.merge([fgmask, fgmask, fgmask])

    # Aplicar la máscara al frame original (bitwise AND)
    foreground = cv2.bitwise_and(frame, mask_3ch)

    # Mostrar resultados
    cv2.imshow("Original", frame)
    cv2.imshow("Fondo Eliminado (color)", foreground)

    # Salir con 'q'
    if cv2.waitKey(30) & 0xFF == ord('q'):
```

```
    break

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

Resultados obtenidos

- Se logra eliminar de forma efectiva el fondo, mostrando solo los objetos en movimiento.
- Se observan resultados limpios gracias a la operación morfológica.
- El sistema responde en tiempo real, siendo útil para aplicaciones como:
    - Seguimiento de personas
    - Realidad aumentada
    - Sistemas de seguridad



Conclusiones

- BackgroundSubtractorMOG2 es una herramienta poderosa y sencilla para segmentar movimiento.

- Es fundamental aplicar filtros morfológicos para eliminar ruido y mejorar los resultados.
- La eficiencia del sistema puede depender de la iluminación y el movimiento de cámara.

Reporte de Práctica: Detección y Coincidencia de Características con ORB

Objetivo de la práctica

Implementar un sistema de detección y emparejamiento de características entre dos imágenes utilizando el algoritmo **ORB (Oriented FAST and Rotated BRIEF)** en conjunto con el emparejador **BFMatcher**, con el fin de identificar similitudes visuales.

- Requisitos: Dos imágenes (Imagen1.jpg e Imagen2.jpg) con contenido similar para comparar

---

Desarrollo del código

1. **Lectura de imágenes** en escala de grises (cv2.imread(..., 0)).
2. **Creación del detector ORB** con cv2.ORB_create().
3. **Detección y descripción de características** para cada imagen con detectAndCompute.
4. **Creación del emparejador** con BFMatcher y configuración para usar la distancia Hamming.
5. **Ordenamiento de coincidencias** por cercanía (menor distancia = mayor similitud).
6. **Visualización de los primeros 10 emparejamientos** usando cv2.drawMatches() y matplotlib.

```
import numpy as np

import cv2
```

```
import matplotlib.pyplot as plt

img1 = cv2.imread('Imagen1.jpg',0)
img2 = cv2.imread('Imagen2.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()
```

Resultados obtenidos

- El algoritmo identifica puntos clave relevantes en ambas imágenes.
- Se emparejan exitosamente los 10 mejores matches (coincidencias visuales).
- Los resultados muestran líneas que conectan las zonas similares entre ambas imágenes.
- Este método es útil para:
    - Reconocimiento de objetos

- Seguimiento de movimiento
- Empalme de imágenes (stitching)

Conclusiones

- ORB es una alternativa eficiente y libre de patentes a algoritmos como SIFT y SURF.
- Su desempeño es adecuado para aplicaciones en tiempo real o con recursos limitados.
- El emparejamiento por distancia Hamming permite comparar descriptores binarios de forma rápida y precisa.

Link github:

https://github.com/Leo1Glez99/Visi-n-Artificial/tree/main/Practica_011