César Soltero Pérez

18310460

7E1

Sistemas de visión artificial y procesamiento de imágenes

Práctica 11

Igualdades con rotación y reducción de fondo

Objetivo 1: De la imagen deseada encontrar las similitudes en otra imagen.

Desarrollo:

Imágenes a comparar

Resultado:

Keypoints Imagen 1: 2553
Keypoints Imagen 2: 2319
Similar Matches: 1982
Porcentaje de 85.4678%

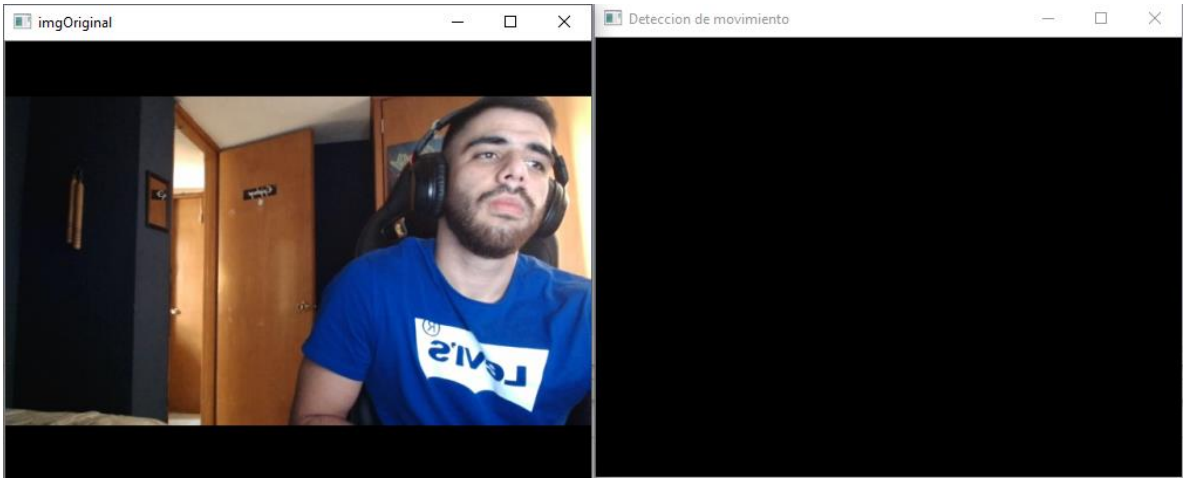
Código:

```
import cv2
import numpy as np
import imutils

original = cv2.imread("perro1.jpg")
original = imutils.resize(original, width=500)
image_to_compare = cv2.imread("perro2.jpg")
image_to_compare = imutils.resize(image_to_compare, width=500)

#

if original.shape == image_to_compare.shape:
    print('Las imagenes tiene el mismo tamaño y canal')
    difference = cv2.subtract(original, image_to_compare)
    b, g, r = cv2.split(difference)
    print(cv2.countNonZero(b))

    if (cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and
cv2.countNonZero(r) == 0):
        print('Las imagenes son completamente iguales')
    else:
        print('Las imagenes no son iguales')

#

shift = cv2.xfeatures2d.SIFT_create()
kp_1, desc_1 = shift.detectAndCompute(original, None)
kp_2, desc_2 = shift.detectAndCompute(image_to_compare, None)
print("Keypoints imagen 1", str(len(kp_1)))
print("Keypoints imagen 2", str(len(kp_2)))
index_params = dict(algorithm=0, trees=5)
search_params = dict()
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(desc_1, desc_2, k=2)
good_points = []

for m, n in matches:

    if m.distance < 0.6*n.distance:
        good_points.append(m)
```

```
number_keypoints = 0
if (len(kp_1) <= len(kp_2)):
    number_keypoints = len(kp_1)
else:
    number_keypoints = len(kp_2)

print("similar matches",len(good_points))
print("Porcentaje de ", len(good_points) / number_keypoints * 100,
"%")

result = cv2.drawMatches(original, kp_1, image_to_compare, kp_2,
good_points, None)

cv2.imshow("Resultado", cv2.resize(result, None, fx = 1, fy=1))
cv2.imwrite("Feature_matching.jpg", result)
cv2.imshow("Original", original)
cv2.imshow("Duplicate", image_to_compare)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

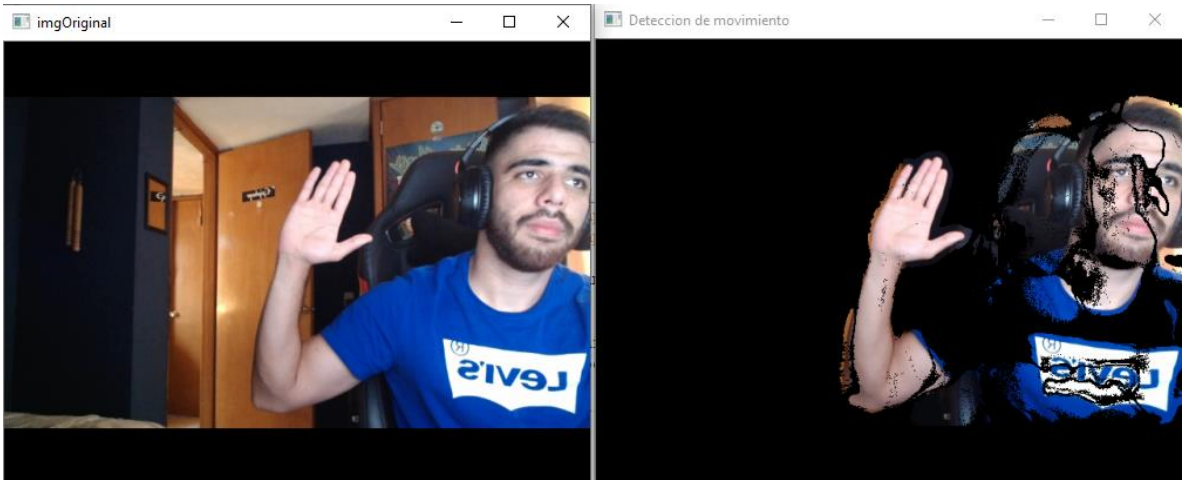
Objetivo 2: En VIDEO poder extraer el fondo de la imagen mediante la detección de movimiento.

Imagen estática (sin movimiento)



El resultado es, una imagen completamente negra

Imagen con ovimiento



Al detectar movimiento, se muestra una imagen sin fondo, siendo visible sólo la parte en movimiento

Código:

```
import cv2
import imutils

cam = cv2.VideoCapture(1)
##

knn_sub = cv2.createBackgroundSubtractorKNN()
mog2_sub = cv2.createBackgroundSubtractorMOG2()

while(cam.isOpened()):
    ret, frame = cam.read()

    framegauss = cv2.GaussianBlur(frame, (5, 5), 0)
    image_blur = cv2.GaussianBlur(framegauss, (51,
51),cv2.BORDER_DEFAULT)
    image_bw = cv2.cvtColor(image_blur, cv2.COLOR_BGR2GRAY)
    framegauss = cv2.GaussianBlur(image_bw, (5, 5), 0)

    if not ret:
        break

    mog_sub_mask = mog2_sub.apply(framegauss)
    knn_sub_mask = knn_sub.apply(framegauss)
```

```
    # Suma
    mask2 = cv2.add(mog_sub_mask, knn_sub_mask)
    res = cv2.bitwise_and(frame, frame, mask=mask2)
    frame = imutils.resize(frame, width=500)
    res = imutils.resize(res, width=500)
    cv2.imshow('imgOriginal', frame)
    cv2.imshow('Deteccion de movimiento', res)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('a'):
        break
```

Repositorio:

https://github.com/Cesarsp41/Practica-11