МІНІСТЕРСТВО ОСВІТИ І НАУКИ
УКРАЇНИ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ
<<ЛЬВІВСЬКА ПОЛІТЕХНІКА>>

Інститут ІКНІ

Кафедра систем штучного інтелекту

ЗВІТ

Лабораторна робота **№2**
З курсу “ Обробка зображень методами штучного інтелекту”

Виконав:

Гавриляк Тарас

гр. **КН-408**

Прийняв(ла):

Пелешко Д. Д.

Львів – 2022

Тема: Суміщення зображень на основі використання дескрипторів.

Мета: Навчитись вирішувати задачу суміщення зображень засобом видобування особливих точок і використання їх в процедурах матчінгу.

Завдання

Варіант №8 **(BRISK)**

Вибрати з інтернету набори зображень з різною контрастністю і різним флуктуаціями освітленості. Для кожного зображення побудувати варіант спотвореного (видозміненого зображення). Для кожної отриманої пари побудувати дескриптор і проаналізувати можливість суміщення цих зображень і з визначення параметрів геметричних перетворень (кут повороту, зміщень в напрямку x і напрямку y).

Для перевірки збігів необхідно написати власну функцію матчінгу, а результати її роботи перевірити засобами OpenCV. Якщо повної реалізації дескриптора не має в OpenCV, то такий необхідно створити власну функцію побудови цих дискрипторів. У цьому випадку матчінг можна здійснювати стандартними засобами (якщо це можливо).

Код програми (Python):

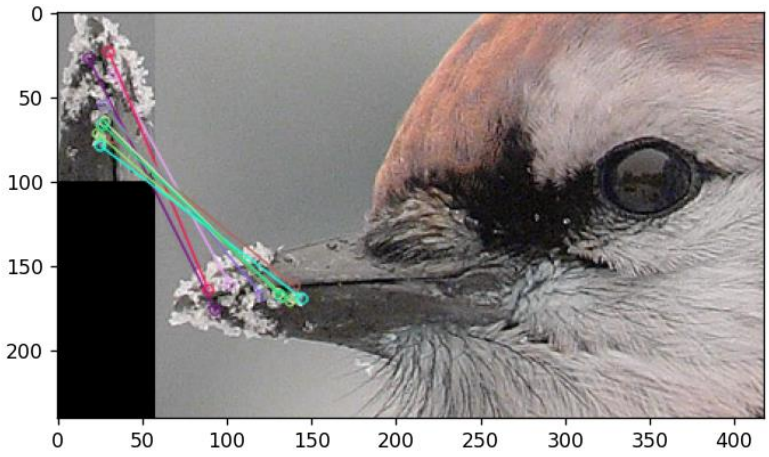
```
import cv2
import matplotlib.pyplot as plt

brisk = cv2.BRISK_create()


def hamming(x, y):
    assert x.shape == y.shape

    return sum(el1 != el2 for el1, el2 in zip(x, y))


def cvBF(image1, image2, descriptors1, descriptors2, keypoints1, keypoints2):
    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches = bf.match(descriptors1, descriptors2)
    matches = sorted(matches, key=lambda x: x.distance)
    match_img = cv2.drawMatches(image1, keypoints1, image2, keypoints2,
matches[:10], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.imshow(match_img), plt.show()


def ownMatch(image1, image2, descriptors1, descriptors2, keypoints1,
keypoints2):
    matches = []
    for i, el1 in enumerate(descriptors1):
        for j, el2 in enumerate(descriptors2):
            matches.append(cv2.DMatch(_distance=int(hamming(el1, el2)),
_imgIdx=0, _queryIdx=i, _trainIdx=j))

    matches = sorted(matches, key=lambda x: x.distance)

    match_img = cv2.drawMatches(image1, keypoints1, image2, keypoints2,
matches[:10], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
```

```
    plt.imshow(match_img), plt.show()


img1 = cv2.imread('bird_part.jpg')
img1 = cv2.rotate(img1, cv2.ROTATE_90_CLOCKWISE)
img2 = cv2.imread('bird.jpg')

kp1, des1 = brisk.detectAndCompute(img1, None)
kp2, des2 = brisk.detectAndCompute(img2, None)

ownMatch(img1, img2, des1, des2, kp1, kp2)
cvBF(img1, img2, des1, des2, kp1, kp2)
```

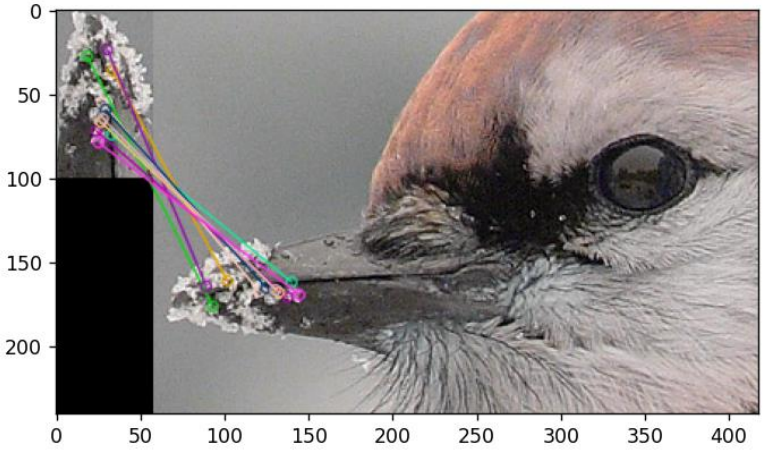
Результат роботи програми:

Для пошуку матчів, я вирізав частину фото, яке подається на вхід, за допомогою засобів OpenCV повернув вирізане фото та подав для порівняння. Програмно обмежив кількість точок матчінгу = 5, щоб прискорити обрахунки.

Результат роботи власного матчера:



Результат роботи BRISK:



Висновок: у ході роботи - навчився вирішувати задачу суміщення зображень засобом видобування особливих точок і використав їх в процедурах матчінгу.