Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій
Кафедра «Системи штучного інтелекту»

Звіт до лабораторної роботи №2
З дисципліни «Обробка зображень методами штучного інтелекту»

Виконав:
студент групи КН-410
Свистович С. А.
Прийняв:
Пелешко Д.Д.

Львів-2022

Варіант 11

Завдання:

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 геометричних перетворень (кут повороту, зміщень в напрямку x і напрямку y).

11. ORB

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 дескрипторів. У цьому випадку матчінг можна здійснювати стандартними засобами (якщо це можливо).

Код завдання:

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from scipy.spatial.distance import hamming

def detect_orb_descriptors(img1, img2):
    orb = cv.ORB_create()
    kp1, des1 = orb.detectAndCompute(img1, None)
    kp2, des2 = orb.detectAndCompute(img2, None)

    print(f"Keypoints: {len(kp1)}, descriptors: {des1.shape}")
    print(f"Keypoints: {len(kp2)}, descriptors: {des2.shape}")
    return kp1, kp2, des1, des2

def bf_match(img1, img2, kp1, kp2, des1, des2):
    bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key = lambda x: x.distance)

    fig = plt.figure(figsize=(25, 25))
    ax = fig.add_subplot()
    ax.axis('off')
    img3 = cv.drawMatches(img1, kp1, img2, kp2, matches[:10], None,
                          flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.imshow(img3)
```

```
    plt.show()

def my_match(img1, img2, kp1, kp2, des1, des2):
    matches = []
    for i, k1 in enumerate(des1):
        for j, k2 in enumerate(des2):
            matches.append(cv.DMatch(_distance = hamming(k1, k2) * len(k1),
                                     _imgIdx = 0, _queryIdx = i, _trainIdx =
j))
    matches = sorted(matches, key = lambda x: x.distance)

    fig = plt.figure(figsize=(25, 25))
    ax = fig.add_subplot()
    ax.axis('off')
    img3 = cv.drawMatches(img1, kp1, img2, kp2, matches[:10], None,
                          flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.imshow(img3)
    plt.show()

img_1 = cv.cvtColor(cv.imread("train.jpg"), cv.COLOR_BGR2RGB)
img_2 = cv.cvtColor(cv.imread("query.jpg"), cv.COLOR_BGR2RGB)

kp1, kp2, des1, des2 = detect_orb_descriptors(img_1, img_2)

bf_match(img_1, img_2, kp1, kp2, des1, des2)

my_match(img_1, img_2, kp1, kp2, des1, des2)

img_1 = cv.cvtColor(cv.imread("train.jpg"), cv.COLOR_RGB2GRAY)
img_2 = cv.cvtColor(cv.imread("query.jpg"), cv.COLOR_RGB2GRAY)

kp1, kp2, des1, des2 = detect_orb_descriptors(img_1, img_2)

bf_match(img_1, img_2, kp1, kp2, des1, des2)

my_match(img_1, img_2, kp1, kp2, des1, des2)
```

Результат:

Brute-Force Matching:

My Matching:

Висновок: під час виконання даної лабораторної роботи, я навчився вирішувати задачу суміщення зображень засобом видобування особливих точок і використання їх в процедурах матчінгу.