МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №2

З дисципліни «Обробка зображень методами штучного інтелекту»

Виконав:
студент групи КН-408
Черещук Любомир

Викладач:
Пелешко Д. Д.

Львів – 2022

Тема роботи: Суміщення зображень на основі використання дескрипторів.
Мета роботи: Навчитись вирішувати задачу суміщення зображень засобом видобування особливих точок і використання їх в процедурах матчінгу.

Теоретичні відомості.

У 2004 році Д.Лоу, Університет Британської Колумбії, придумав алгоритм - Scale Invariant Feature

Transform (SIFT), який видобуває ключові (особливі) точки і обчислює їх дескриптори.

Загалом алгоритмі SIFT складається з п’яти основних етапів:

1. Виявлення масштабно-просторових екстремумів (Scale-space Extrema Detection) – основним завданням етапу є виділення локальних екстремальних точок засобом побудова пірамід гаусіанів (Gaussian) і різниць гаусіанів (Difference of Gaussian, DoG).

2. Локалізація ключових точок (Keypoint Localization) - основним завданням етапу є подальше учточнення локальних екстремумів з метою фільтрації їх набору - тобто видалення з подальшого аналізу точок, які є краєвими, або мають низьку контрастність.

3. Визначення орієнтації (Orientation Assignment) - для досягнення інваріантності повороту растра на цьому етапі кожній ключовій точці присвоюється орієнтація.

4. Дескриптор ключових точок (Keypoint Descriptor) - завданням етпау є побудова дескрипторів, які містяь інформація про окіл особливої точки для задачі подальшого порівння на збіг.

5. Зіставлення по ключових точках (Keypoint Matching) - пошук збігів для вирішення завдання суміщення зображень.

1.2. Алгоритм RANSAC - Random sample consensus

Для досягнення високої точності визначення збігів об'єктів на зображеннях зазвичай відфільтрувати дескриптори тільки за відстані є

недостатньо. Якщо об'єкт рухається на сцені або зображений з іншого ракурсу, то при застосуванні трансформації «накладення» n точок одного зображення на відповідні по найближчому сусіду n точок іншого, можна виявити особливості, що не відносяться до загального об'єкту і тим самим зменшити кількість хибно виявлених зв'язків.

Схема роботи алгоритму RANSAC полягає в циклічному повторенні пошуку матриці трансформації HH між чотирма особливими точками si, які випадково обираються на одному зображенні, і відповідними їм точками на другому:

$$s_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}$$

Найкращою матрицею трансформації вважається та, в якій досягнуто мінімум суми відхилень будь-яких спеціальних точок зображень при перетворенні HH, за задану кількість циклів (≤ 2000):

$$\sum_i \left[\left(x_i - \frac{h_{11}x'_i + h_{12}y'_i + h_{13}}{h_{31}x'_i + h_{32}y'_i + h_{33}}\right)^2 + \left(y_i - \frac{h_{21}x'_i + h_{22}y'_i + h_{23}}{h_{31}x'_i + h_{32}y'_i + h_{33}}\right)^2 \right]$$

У підсумкову множину srcPoints’ додаються тільки ті точки srcPo srcPointsii, відхилення яких становить менше заданого порогу:

$$|dstPointsi - H * srcPointsi| < reprojThreshold$$

,де srcPointssrcPoin - множина усіх особливих точок першого зображення, а dstPoints - множина відповідних їм особливі точки другого.

Хід роботи

Варіант 10 (номер в списку групи – 32).

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).

10. BRIEF.

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:

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt


def match_self(des1, des2):
    match = []
    distances = {}

    for i, d_1 in enumerate(des1):
        if np.sum(d_1) != 0:
            d = []
            for _, d_2 in enumerate(des2):
                norm = cv.norm(d_1, d_2, cv.NORM_HAMMING)
                d.append(norm)
            orders = np.argsort(d).tolist()
            if d[orders[0]]/d[orders[1]] <= 0.98:
                match.append((i, orders[0]))
            distances[f'{i}-{orders[0]}'] = d[orders[0]]

    return [cv.DMatch(pair[0], pair[1], distances[f'{pair[0]}-{pair[1]}']) for pair
in match]
```

```
images = ['img1.jpeg', 'img2.jpeg']

for image in images:
    # Load the image
    training_image = cv.imread(image, cv.IMREAD_GRAYSCALE)

    # resize
    scale_percent = 30  # percent of original size
    width = int(training_image.shape[1] * scale_percent / 100)
    height = int(training_image.shape[0] * scale_percent / 100)
    dim = (width, height)
    training_image = cv.resize(training_image, dim, interpolation=cv.INTER_AREA)

    # Create test image by adding Scale Invariance and Rotational Invariance
    test_image = cv.pyrDown(training_image)
    test_image = cv.pyrDown(test_image)
    num_rows, num_cols = test_image.shape[:2]
    rotation_matrix = cv.getRotationMatrix2D((num_cols/2, num_rows/2), 30, 1)
    test_image = cv.warpAffine(test_image, rotation_matrix, (num_cols, num_rows))

    # initialization of detector and extractor
    fast = cv.FastFeatureDetector_create()
    brief = cv.xfeatures2d.BriefDescriptorExtractor_create()

    # key points
    kp1 = fast.detect(training_image, None)
    kp2 = fast.detect(test_image, None)

    # descriptors
    kp1, des1 = brief.compute(training_image, kp1)
    kp2, des2 = brief.compute(test_image, kp2)

    # self match
    matches = match_self(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)
    result = cv.drawMatches(training_image, kp1, test_image,
                            kp2, matches[:10], None, flags=2)

    plt.rcParams['figure.figsize'] = [14.0, 7.0]
    plt.title('Best Self Matching Points')
    plt.imshow(result)
    plt.show()

    # library match
    bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=True)
    matches_bf = bf.match(des1, des2)
    matches_bf = sorted(matches_bf, key=lambda x: x.distance)
    result_bf = cv.drawMatches(training_image, kp1, test_image,
                            kp2, matches_bf[:10], None, flags=2)

    plt.rcParams['figure.figsize'] = [14.0, 7.0]
```

```
plt.title('Best BF Matching Points')
plt.imshow(result_bf)
plt.show()
```

Результат роботи програми:

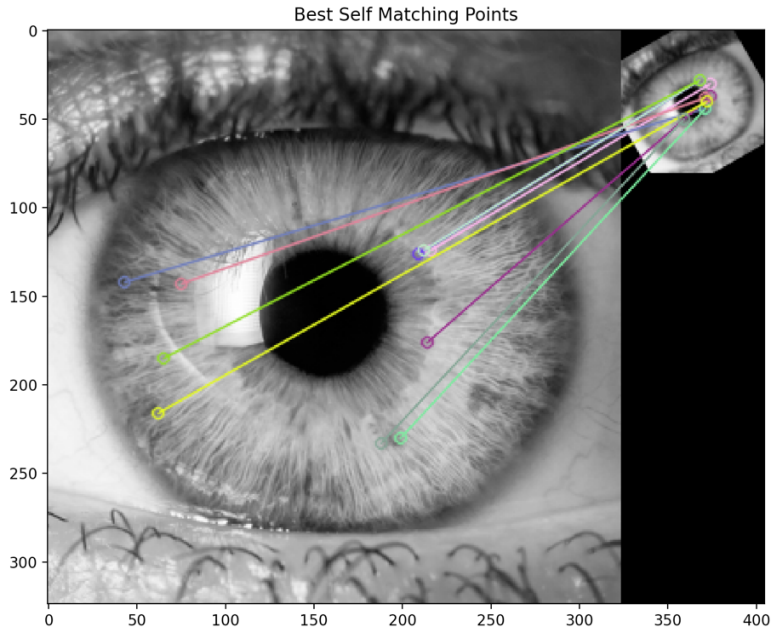

Рис. 1 Результат роботи власного метчінгу для 1 зображення.

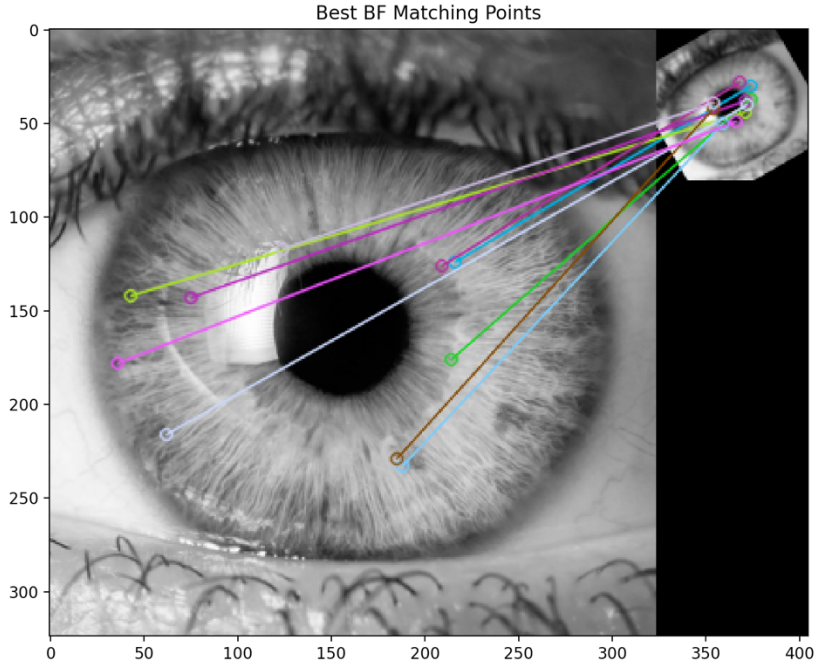

Рис. 2 Результат роботи брут форс метчінгу для 1 зображення.

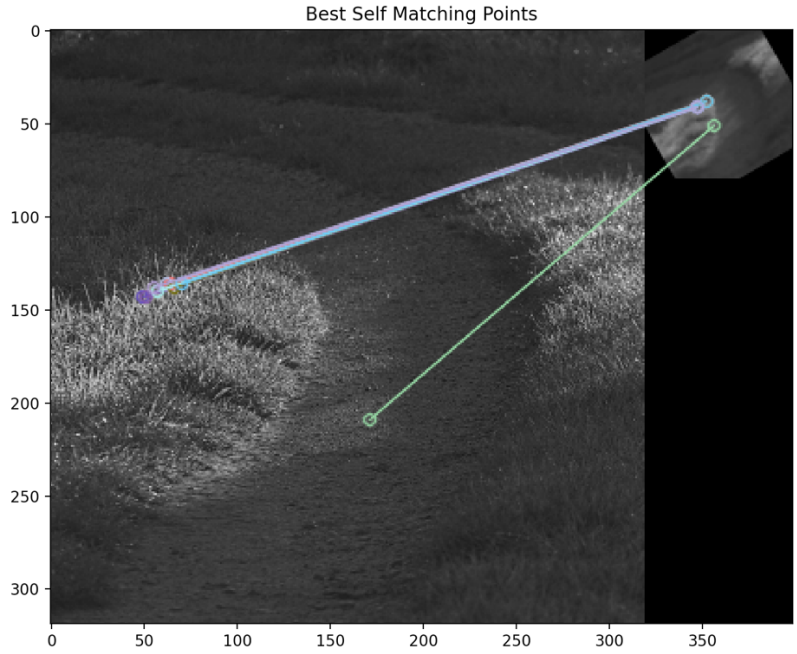

Рис. 1 Результат роботи власного метчінгу для 2 зображення.

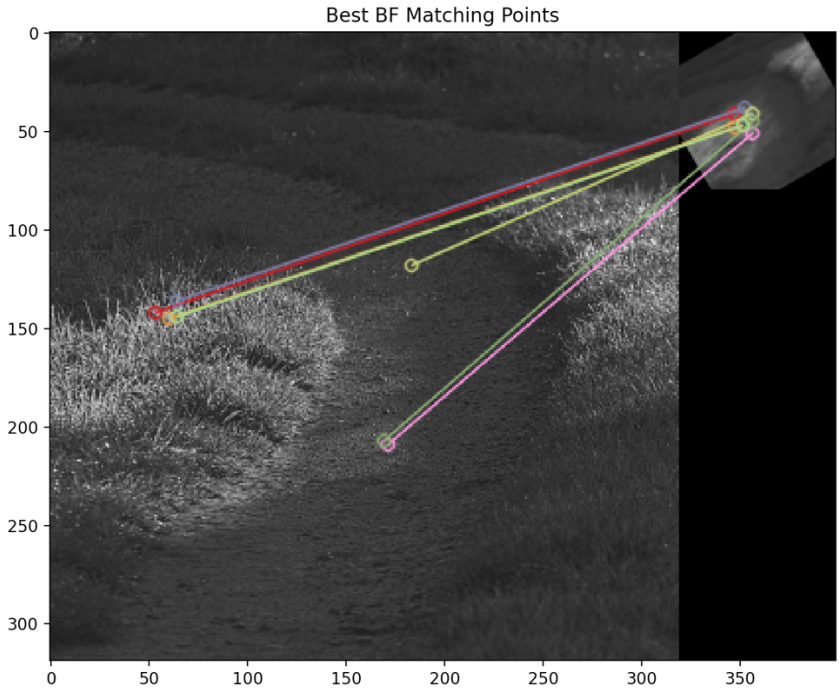

Рис. 2 Результат роботи брут форс метчінгу для 2 зображення.

Висновок: виконавши дану роботу я навчився вирішувати задачу суміщення зображень засобом видобування особливих точок і використання їх в процедурах матчінгу. Також було створений власний метчінг на основі норми Хемінга. Оскільки це норма Хемінга, то параметр crosscheck не є необхідним для власного метчінгу.