МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №2

З курсу “Обробка зображень методами штучного інтелекту”

Виконав:
студент групи КН-410
Романишин Микола

Викладач:
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.

Хід роботи

Варіант 10

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).

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

def fast_plus_brief(img1, img2):

  # Initiate FAST detector
  fast = cv.FastFeatureDetector_create()
  # Initiate BRIEF extractor
  brief =
cv.xfeatures2d.BriefDescriptorExtractor_create(use_orientation = True)
  # find the keypoints with FAST
  kp1 = fast.detect(img1,None)
  kp2 = fast.detect(img2,None)
  # compute the descriptors with BRIEF
  kp1, des1 = brief.compute(img1, kp1)
  kp2, des2 = brief.compute(img2, kp2)

  return kp1, kp2, des1, des2

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

    fig = plt.figure(figsize=(20, 20))
    ax = fig.add_subplot()
```

```
    ax.axis('off')
    result = cv.drawMatches(img1, kp1, img2, kp2, matches[:10], None,

flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    plt.imshow(result)
    plt.show()

def custom_matcher(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),
                                     _imgIdx = 0, _queryIdx = i,
_trainIdx = j))
    matches = sorted(matches, key = lambda x: x.distance)

    fig = plt.figure(figsize=(20, 20))
    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()

img1 = cv.imread("sample_data/train.jpg", cv.IMREAD_GRAYSCALE)
img2 = cv.imread("sample_data/test.jpg", cv.IMREAD_GRAYSCALE)

kp1, kp2, des1, des2 = fast_plus_brief(img1, img2)
bf_match(img1, img2, kp1, kp2, des1, des2)

custom_matcher(img1, img2, kp1, kp2, des1, des2)
```

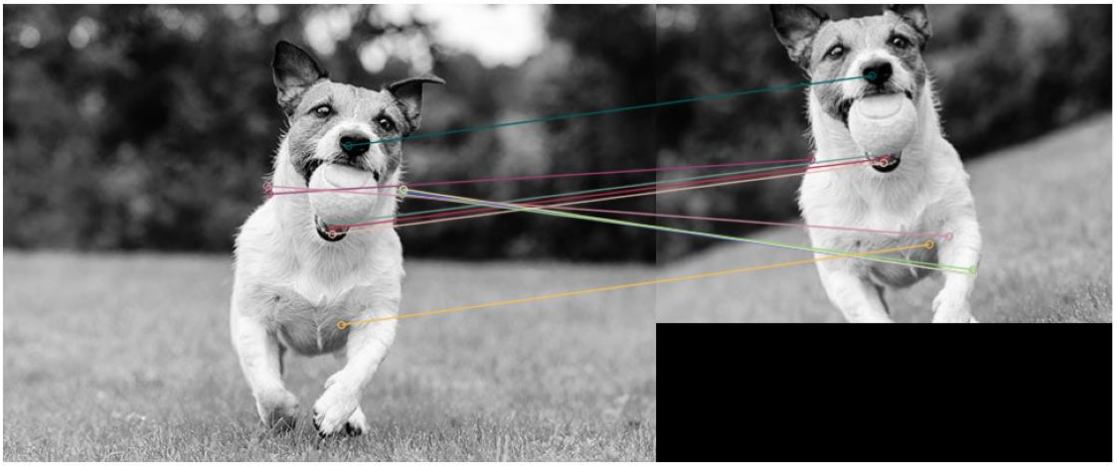

Рис. 1 Результати роботи з використання Brute-Force Matching

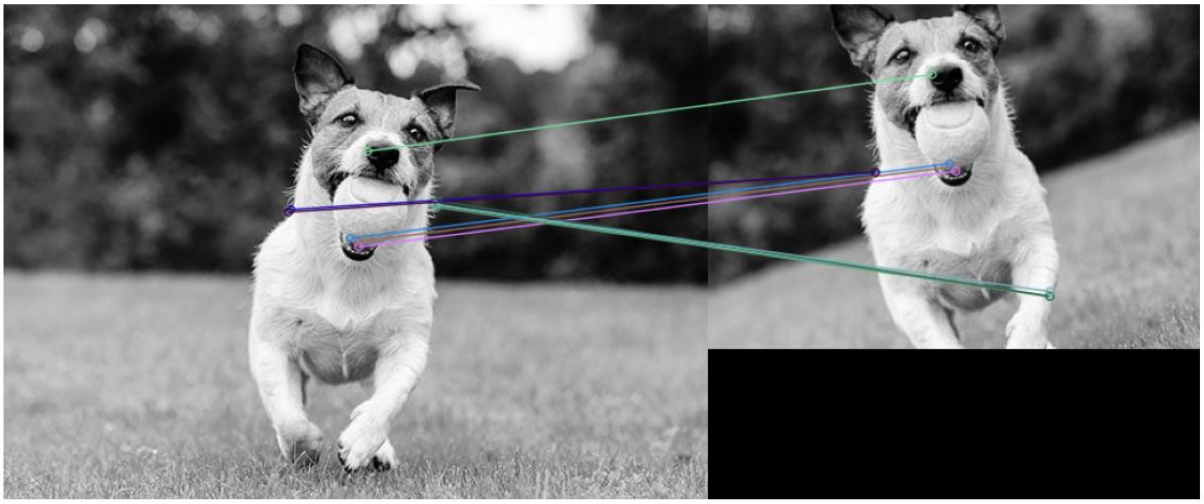

Рис. 2 Результати роботи з використання власного матчінгу

Висновки: Я навчився вирішувати задачу суміщення зображень засобом видобування особливих точок і використав їх в процедурах матчінгу.