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-408

Бурак Марко

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.

**Теоретичні відомості**

Метод SIFT.

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 алгоритм 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 побудови пірамід гаусіанів (Gaussian) і різниць гаусіанів (Difference of Gaussian, DoG).

2. Локалізація ключових точок (Keypoint Localization) - основним завданням етапу є подальше уточнення 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.

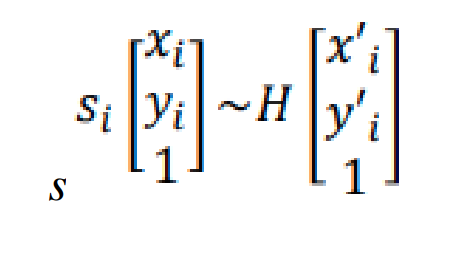
4. Дескриптор ключових точок (Keypoint Descriptor) - завданням етапу є побудова дескрипторів, які містяь інформацію про окіл особливої точки для задачі подальшого порівння на збіг.

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

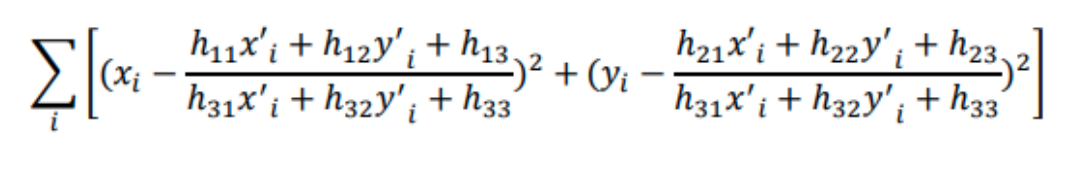
Алгоритм RANSAC - Random sample consensus

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

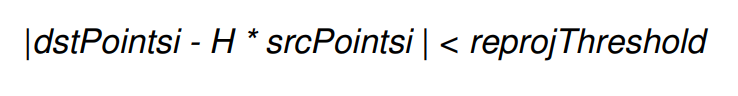
Схема роботи алгоритму RANSAC полягає в циклічному повторенні пошуку матриці трансформації 𝐻H між чотирма особливими точками , які випадково обираються *i* на одному зображенні, і відповідними їм точками на другому:



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



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



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

**Хід роботи**

Варіант – 4

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 геметричних перетворень (кут повороту, зміщень в напрямку х і напрямку y).

AKAZE.

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:

!pip3 uninstall -y opencv-contrib-python

!pip3 uninstall -y opencv-python

!pip3 install opencv-contrib-python

!pip3 install opencv-python

# !pip install opencv-contrib-python==3.4.2.17

# !pip3 install opencv-python==3.4

import numpy as np

from numpy.linalg import norm

import cv2 as cv

import matplotlib.pyplot as plt

import argparse

import math

import cv2

from google.colab.patches import cv2\_imshow

img1\_name = 'cn-tower.jpg'

img2\_name = 'cn-tower2.jpg'

img1 = cv2.imread(img1\_name)

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

img2 = cv2.imread(img2\_name)

gray2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

detector = cv2.AKAZE\_create()

(kps1, descs1) = detector.detectAndCompute(gray1, None)

(kps2, descs2) = detector.detectAndCompute(gray2, None)

img\_1\_key = cv2.drawKeypoints(gray1,kps1,img1)

plt.imshow(img\_1\_key)

img\_2\_key = cv2.drawKeypoints(gray2,kps2,img2)

plt.imshow(img\_2\_key)

def BF\_FeatureMatcher(des1,des2):

brute\_force = cv2.BFMatcher(cv2.NORM\_HAMMING,crossCheck=True)

no\_of\_matches = brute\_force.match(des1,des2)

no\_of\_matches = sorted(no\_of\_matches,key=lambda x:x.distance)

return no\_of\_matches

def display\_output(pic1,kpt1,pic2,kpt2,best\_match):

output\_image = cv2.drawMatches(pic1,kpt1,pic2,kpt2,best\_match,None,flags=2)

cv2\_imshow(output\_image)

number\_of\_matches = BF\_FeatureMatcher(descs1,descs2)

tot\_feature\_matches = len(number\_of\_matches)

# printing total number of feature matches found

print(f'Total Number of Features matches found are {tot\_feature\_matches}')

display\_output(gray1,kps1,gray2,kps2,number\_of\_matches)

def my\_matcher(kps1, descs1, kps2, descs2,img1, img2):

matches = []

for i in range(len(descs1)):

for j in range(len(descs2)):

distance = norm(descs1[i] - descs2[j])

if distance < 800:

matches.append((distance,i,j))

match\_to\_draw = []

for match in matches:

match\_to\_draw.append(cv.DMatch(\_distance=match[0], \_imgIdx=0, \_queryIdx=match[1], \_trainIdx=match[2]))

return match\_to\_draw

def draw\_my\_matcher(kps1, kps2, match\_points\_to\_draw, gray1, gray2):

img\_with\_matches = cv.drawMatches(gray1, kps1, gray2, kps2, match\_points\_to\_draw, None,

flags=2)

plt.figure(figsize=(10,20))

plt.imshow(img\_with\_matches)

plt.show()

matches\_to\_draw = my\_matcher(kps1, descs1, kps2, descs2,gray1,gray2)

draw\_my\_matcher(kps1, kps2, matches\_to\_draw, gray1, gray2)

img3\_name = 'monument.jpg'

img4\_name = 'monument2.jpg'

img3 = cv2.imread(img3\_name)

gray3 = cv2.cvtColor(img3, cv2.COLOR\_BGR2GRAY)

img4 = cv2.imread(img4\_name)

gray4 = cv2.cvtColor(img4, cv2.COLOR\_BGR2GRAY)

detector = cv2.AKAZE\_create()

(kps3, descs3) = detector.detectAndCompute(gray3, None)

(kps4, descs4) = detector.detectAndCompute(gray4, None)

number\_of\_matches = BF\_FeatureMatcher(descs3,descs4)

tot\_feature\_matches = len(number\_of\_matches)

# printing total number of feature matches found

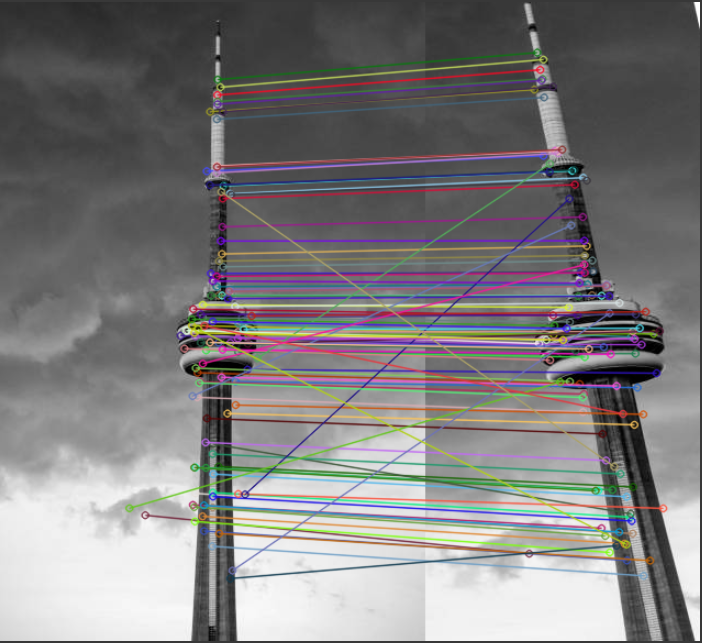
print(f'Total Number of Features matches found are {tot\_feature\_matches}')

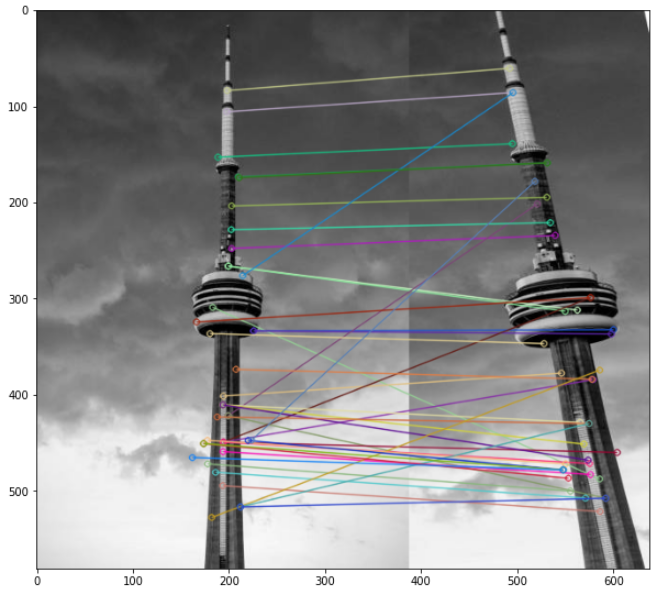
display\_output(gray3,kps3,gray4,kps4,number\_of\_matches)

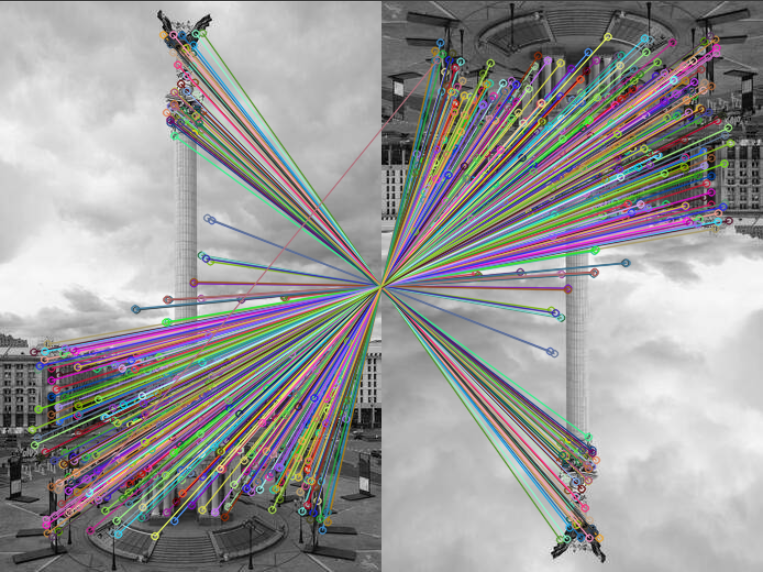
matches\_to\_draw = my\_matcher(kps3, descs3, kps4, descs4, gray3, gray4)

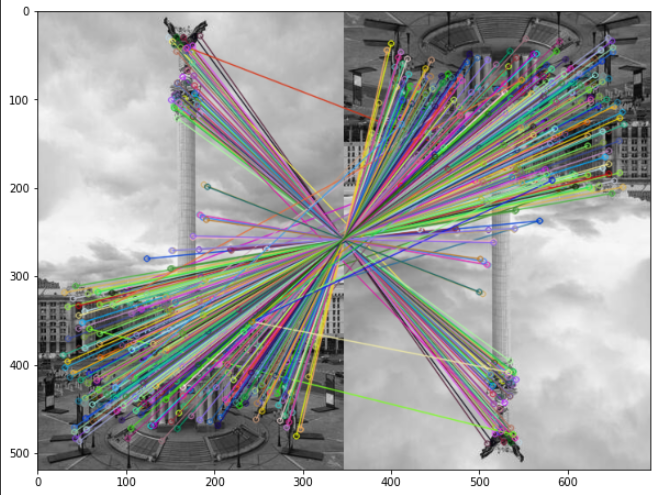
draw\_my\_matcher(kps3, kps4, matches\_to\_draw, gray3, gray4)

Результати роботи програми:









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