МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра систем штучного інтелекту

Лабораторна робота №2

З курсу “Обробка зображень методами штучного інтелекту”

Виконала:
студентка групи КН-408
Жук Анастасія

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 містять 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.

Хід роботи

Код роботи:

```
# -*- coding: utf-8 -*-
"""Untitled3.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1kRJbQNGYfoZKRm4QMvXmSarTHS0S2cT8
"""

!pip3 uninstall -y opencv-contrib-python
!pip3 uninstall -y opencv-python

!pip3 install opencv-contrib-python
!pip3 install opencv-python

!wget -c https://i.imgur.com/K74Rsq2.jpg -O painting.jpg
!wget -c https://i.imgur.com/HnwPrgi.jpg -O painting_in_life.jpg

import cv2 as cv
import sys
import io
import numpy as np
import pandas as pd
from PIL import Image
```

```
from sklearn.decomposition import PCA
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
import imutils
from google.colab.patches import cv2_imshow

def sift(img1, img2):


  sift_detector = cv.SIFT_create()

  keypoints1, descriptions1 = sift_detector.detectAndCompute(img1, None)
  keypoints2, descriptions2 = sift_detector.detectAndCompute(img2, None)



  return keypoints1, descriptions1, keypoints2, descriptions2

def brute_force_opencv_matcher(kps1, descs1, kps2, descs2,img1, img2):
    bf = cv.BFMatcher(cv.NORM_L1)
    matches = bf.knnMatch(descs1,descs2, k=2)

    good = []
    for m,n in matches:
        if m.distance < 0.75*n.distance:
            good.append([m])

    img3 = cv.drawMatchesKnn(img1, kps1, img2, kps2, good[1:20], None, flags=2)
    plt.figure(figsize=(20,20))
```

```
    plt.imshow(img3)
    plt.show()

img1  = cv.imread('/content/king_shine.jpeg', cv.IMREAD_GRAYSCALE)
img2 = cv.imread('/content/king_shine_with_another.jpg', cv.IMREAD_GRAYSCALE)

cv2_imshow(img1)

kps1, descs1, kps2, descs2 = sift(img1, img2)

brute_force_opencv_matcher(kps1, descs1, kps2, descs2,img1, img2)

def matcher(kp1, des1, kp2, des2, img1, img2):
    matches = []
    for i, k1 in enumerate(des1):
        for j, k2 in enumerate(des2):
            matches.append(cv.DMatch(_distance=np.linalg.norm(k1 - k2), _imgIdx=0,
_queryIdx=i, _trainIdx=j))

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

    img3 = cv.drawMatches(img1, kp1, img2, kp2, matches[:10], None,
flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    return img3

def draw_my_matches(img):

  plt.figure(figsize=(20,20))
  plt.imshow(img)
```

```
  plt.show()

img_answer = matcher(kps1, descs1, kps2, descs2,img1, img2)
draw_my_matches(img_answer)

img5 = cv.imread('boom.jpg', cv.IMREAD_GRAYSCALE)  # queryImage
img6 = cv.imread('choc.png', cv.IMREAD_GRAYSCALE)  # trainImage

kps5, descs5, kps6, descs6 = sift(img5, img6)

brute_force_opencv_matcher(kps5, descs5, kps6, descs6,img5, img6)
```

2. Обрані фотографії

Рис 1. Фотографія для першого експерименту

Рис 2. Фотографія для першого експерименту

Рис 3. Фотографія для другого експерименту

Рис 4. Фотографія для дроугого експерименту

3. Пошук декскрипторів

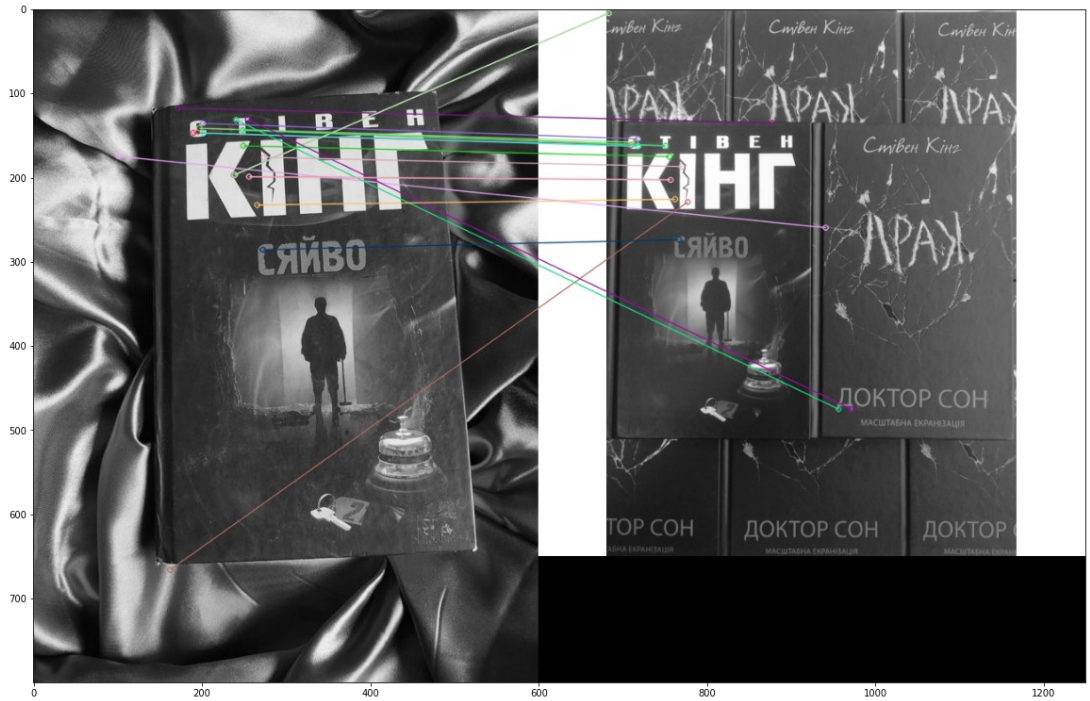

Рис 5. Фотографія першого експерименту з встроєним пошуком

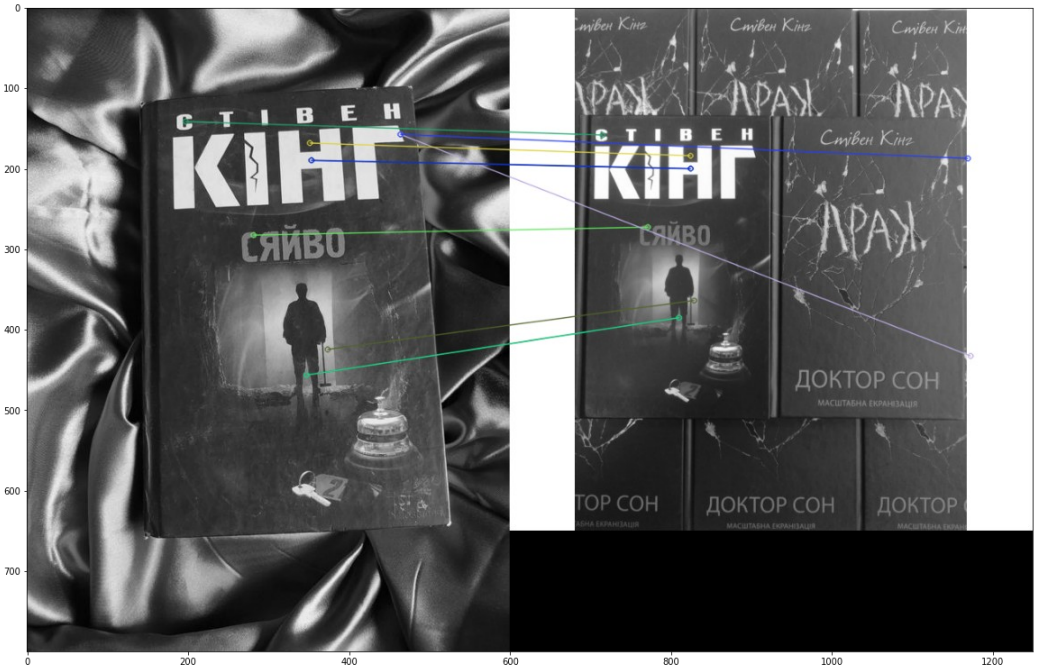

Рис 5. Фотографія першого експерименту з власним пошуком

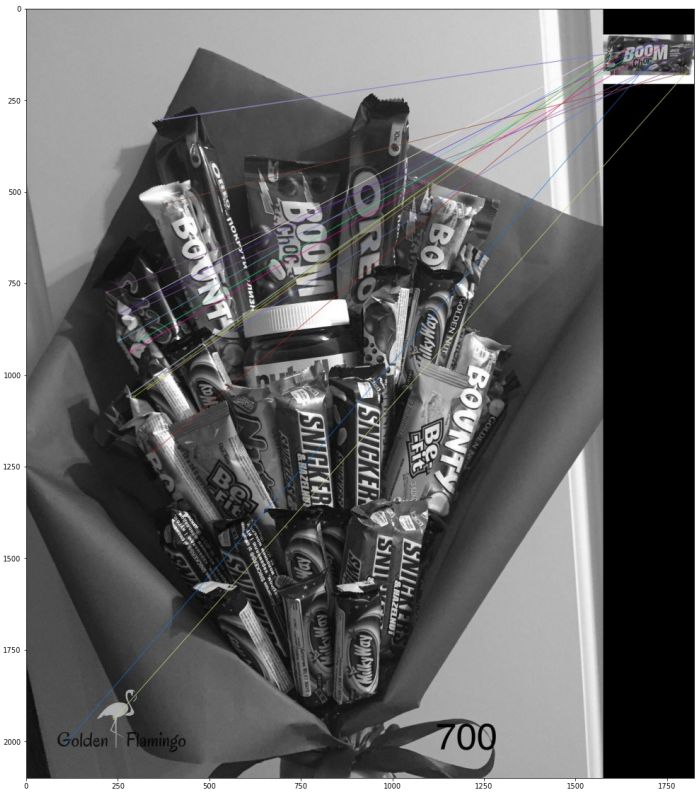

Рис 5. Фотографія першого експерименту з встроєним пошуком

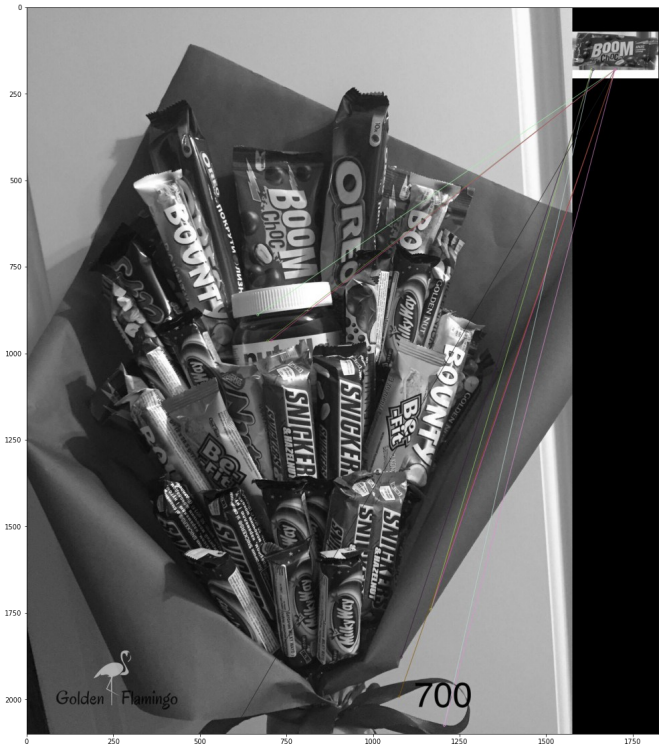

Рис 5. Фотографія першого експерименту з власним пошуком

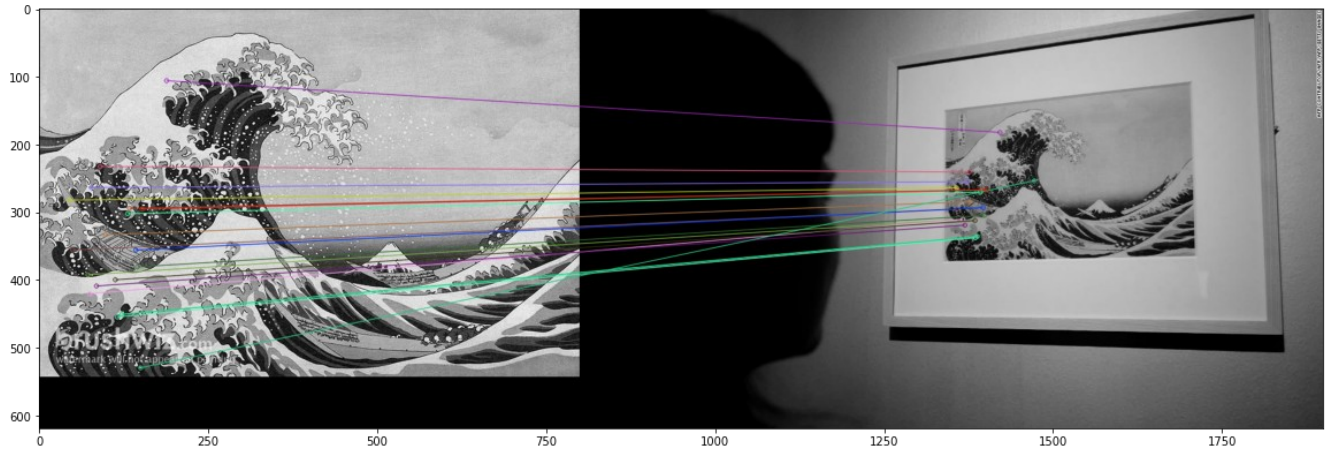

Рис 6. Фотографія третього експерименту з встроєним пошуком

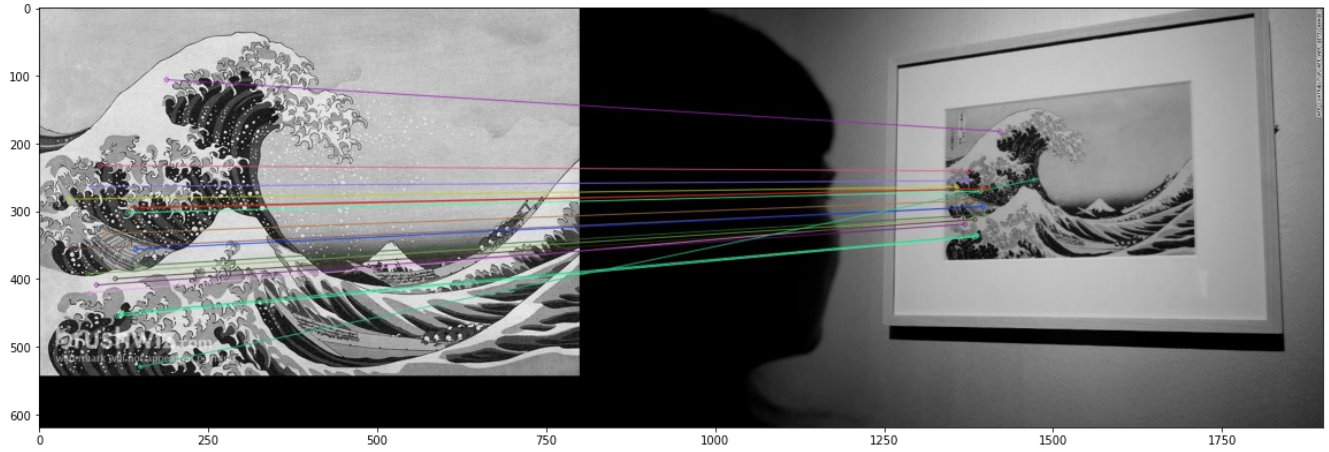

Рис7. Фотографія третього експерименту з власним пошуком

Висновки: Я навчилася вирішувати задачу суміщення зображень засобом видобування особливих точок і використав їх в процедурах матчінгу. Порівнюючи результати роботи вбудованого матчера та власного, можна сказати, що працюють вони доволі схоже. Зображення, які ми отримали в результаті, дуже схожі між собою, матчінг майже однаковий, проте є деякі відмінності, адже у нашому алгоритмі матчінгу використовується інша шкала вибору пошукових точок.