**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №6**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОРІВНЯННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ**

**ДЛЯ СТЕЖЕННЯ ЗА ОБ’ЄКТАМИ У ВІДЕОПОТОЦІ**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

Дослідити принципи та особливості практичного застосування технологій порівняння цифрових зображень для стеження за об’єктами у відеопотоці з використанням спеціалізованих програмних бібліотек.

## **ІІ. Завдання:**

**ГРУПА ВИМОГ 1.**

Удосконалити скрипт із завдання Лр\_5 додавши можливість реалізації порівняння об’єкта ідентифікації (оперативні – високоточні дані ДЗЗ, див. таблицю нижче) з використанням дескриптора зображень.

Вимоги та обмеження:

Дескриптор зображення має стосуватись об’єкта ідентифікації.

Алгоритм та технологія визначення дескриптора зображень – за власним вибором.

Результат порівняння – кількість збігів особливих точок;

Кількість збігів особливих точок дескриптора зображень перевести в ймовірність

ідентифікації.

**Завдання Лр\_5, що потребує удосконаленню.**

Розробити програмний скрипт, що забезпечує цифрову обробку зображень для розрізнення та ідентифікації обраних об’єктів на цифровому знімку земної поверхні з низькою роздільною здатністю за цифровими зображеннями відкритих джерел даних дистанційного зондування Землі (ДЗЗ) із космосу.

Порядок організаційних дій та функціонал програмного скрипта:

1. Обрати район спостереження та об’єкти ідентифікації – однакові за оперативними

та високоточними джерелами даних ДЗЗ – див. табл.

2. Отримати цифрові растрові знімки обраного району земної поверхні з оперативних та високоточних джерел даних ДЗЗ із збереженням їх у файлі відповідного типу.

3. За допомогою програмного скрипта провести кольорову корекцію та / або фільтрацію даних ДЗЗ від оперативних та високоточних джерел відносно об’єкта ідентифікації.

4. Реалізувати програмно кольорову кластеризацію покращених в п.3 зображень об’єкта ідентифікації на даних ДЗЗ від оперативних та високоточних джерел.

5. Здійснити сегментацію кластеризованих в п.4 цифрових зображень від оперативних та високоточних джерел даних ДЗЗ із виділенням контуру об’єкта ідентифікації.

6. Шляхом візуального та / або програмного порівняння контурів обраних об’єктів векторизованих зображень від оперативних та високоточних джерел даних ДЗЗ здійснити ідентифікацію цих об’єктів.

Вимоги та обмеження:

Об’єктами для ідентифікації можуть бути площадні або точкові об’єкти: лісові насадження, вирубки лісів, водойми, площі ерозії поверхні Землі, сільськогосподарські угіддя, посівні площі, будівлі, автівки, техногенні / критичні об’єкти.

Ідентифікація – полягає у встановленні лінгвістичної назви об’єкта та здійснюється за геометрією контура.

Всі процеси обробки повинні бути спрямовані та реалізовані відносно об’єкта ідентифікації.

Порядок, зміст, методи і технології етапів обробки цифрового зображення, вказаних у п.3,4,5 – є результатами обґрунтованих власних R&D досліджень, відносно обраних на даних ДЗЗ об’єктів та спрямовані на головний результат – об’єктова ідентифікація.

Дозволяється змінювати джерела оперативних та еталонних даних ДЗЗ за власним обґрунтованим рішення.

Джерело оперативних даних: <https://livingatlas2.arcgis.com/landsatexplorer/>

Джерело високоточних даних: <https://www.bing.com/maps>

Район зйомки: поблизу села Підріччя Камінь-Каширського району Волинської області.

Дата зйомки: 4 травня 2024 р.

Об'єкт ідентифікації: водойма

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичні моделі операцій над структурами вхідних графічних об’єктів.

Модель реалізує виділення контурів графічних об’єктів (водойм) на цифровому зображенні. Для цього вона покращує зображення з допомогою конвертації у відтінки сірого, Гаусівського розмиття, визначення порогу бінаризації з допомогою інвертованого бінарного порогу, щоб розділити зображення на фон та об’єкти. Далі застосовується згладжування для зменшення шуму. Після цього відбувається кластеризація з допомогою k-means та сегментація. Далі на сегментованому зображенні знаходяться кути Гарріса і за допомогою SIFT обчислюються дескриптори зображення. Після цього з допомогою FLANN знаходяться збіги по особливих точках дескрипторів, які потім відображаються на зображеннях. Наостанок визначається ймовірність ідентифікації об’єкта.

Кластеризацію та сегментацію описано у ЛР5.

Детектор кутів Гарріса – це оператор виявлення кутів та ознак зображення. Спочатку потрібно перетворити зображення. Далі обчислити просторову похідну за та за . Потім треба побудувати структурний тензор . І тепер обчислити відгук за формулою:

де слід .

І нарешті, знаходяться локальні максимуми для обрання оптимальних значеннь для вказування кутів. Кути Гарріса знаходитимуться на сегментованому зображенні.

SIFT (Scale-Invariant Feature Transform) використовується для обчислення особливих точок дескрипторів зображень. Для цього навколо особливої точки береться окіл 16х16, що розділяється на 16 підблоків розміром 4х4. Для кожного підблоку створюється гістограма орієнтацій з 8 бінами. Це дає 128 значень бінів. Ці значення представляються у вигляді вектора, щоб сформувати дескриптор особливої точки.

FLANN (Fast Library for Approximate Nearest Neighbors) – це інструмент, що виконує пошук найближчих сусідів у великих наборах даних. Для цього вона використовує такі структури даних, як kd-дерева, LSH (Locality-Sensitive Hashing), тощо. У цій моделі вона використовується для пошуку збігів по особливих точках дескрипторів зображень та їх подальшій фільтрації.

Ймовірність ідентифікації обчислюється за формулою:

де – кількість збігів, а – кількість особливих точок.

Обробка зображення відбувається з допомогою засобів бібліотеки OpenCV з використанням Python.

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до моделі до програмного скрипту:

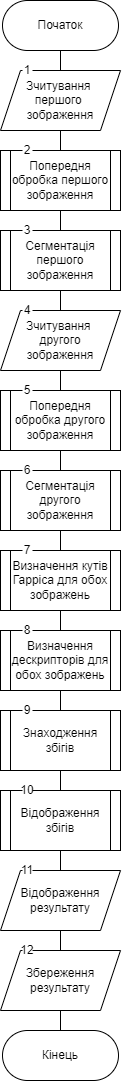


Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається із зчитування першого зображення з допомогою бібліотеки OpenCV. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується попередня обробка першого зображення. Тут зображення перетворюється у відтінки сірого, застосовується Гаусівське розмиття, визначення порогу бінаризації для розділення зображення на фон та об’єкти, та застосовується розмиття для зменшення шуму.

У блоці 3 виконується сегментація другого зображення з допомогою k-means.

У блоці 4 виконується зчитування другого зображення.

У блоці 5 виконується попередня обробка другого зображення. Дії аналогічні до блоку 2.

У блоці 6 відбувається сегментація другого зображення з допомогою k-means.

У блоці 7 відбувається визначення кутів Гарріса на обох сегментованих зображеннях.

У блоці 8 визначаються дескриптори для обох зображень з допомогою SIFT.

У блоці 9 знаходяться та фільтруються збіги по особливих точках дескрипторів з допомогою FLANN.

У блоці 10 знайдені збіги відображаються на результуючому зображенні (воно поєднує перше і друге зображення).

У блоці 11 відображається результат виконання алгоритму.

У блоці 12 виконується збереження результату. На цьому виконання алгоритму завершено.

* 1. **Опис структури проекту програми**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

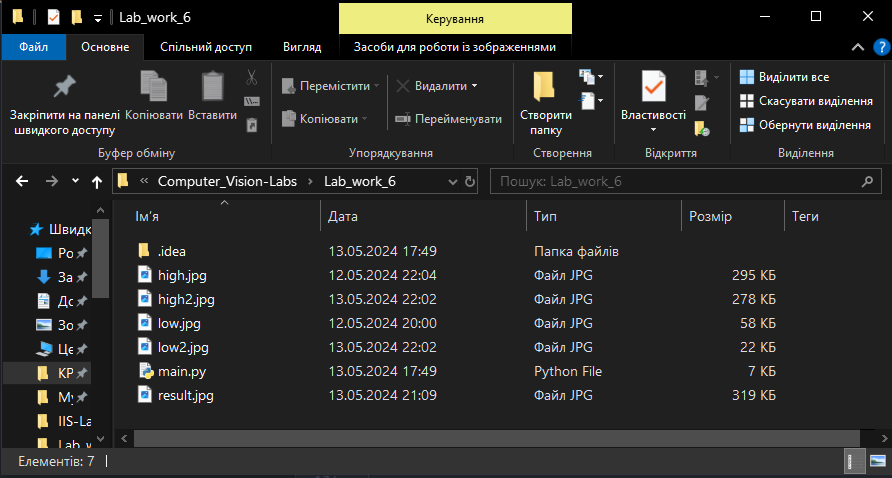


Рисунок 2 – Структура проєкту

Lab\_work\_6 – головний каталог проєкту

main.py – файл програмного коду

high.jpg – високоточний знімок з декількома водоймами

high2.jpg – високоточний знімок з однією водоймою

low.jpg – оперативний знімок з декількома водоймами

low2.jpg – оперативний знімок з однією водоймою

result.jpg – результат порівняння об’єктів ідентифікації

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. Початкове оперативне зображення:

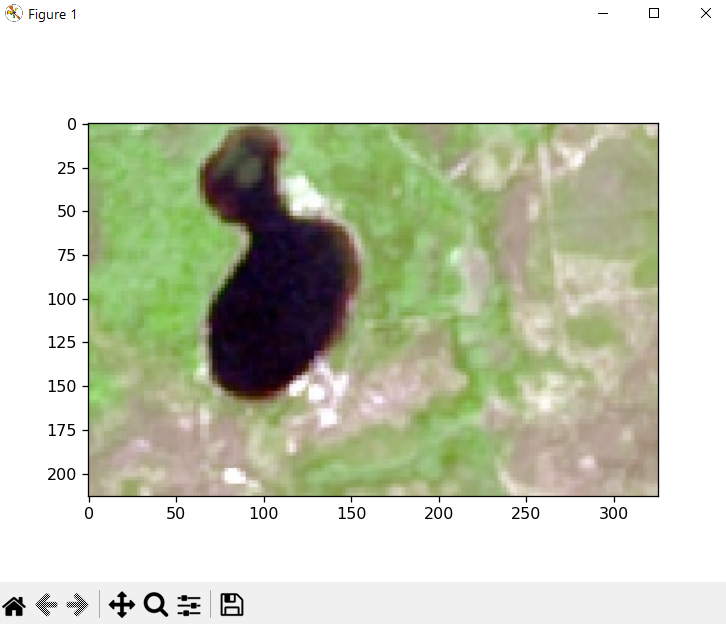


Рисунок 3 – Початкове оперативне зображення

1. Оброблене та сегментоване оперативне зображення:

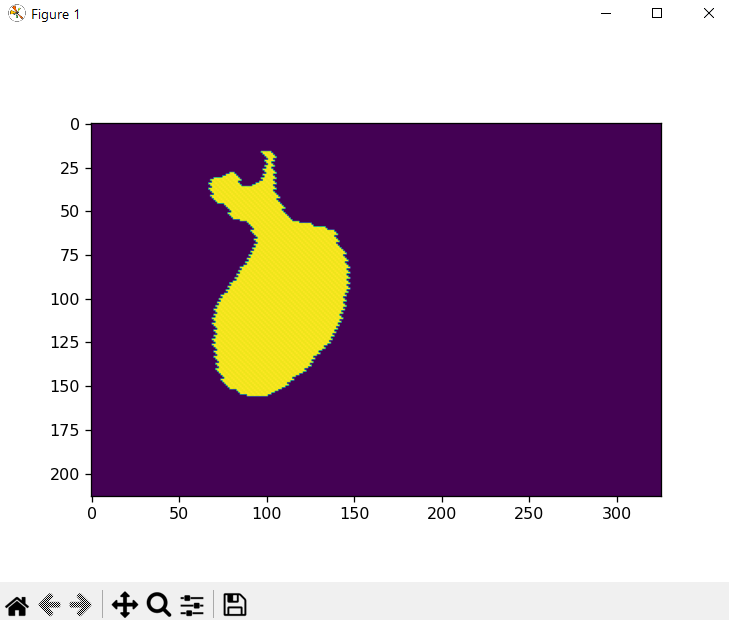


Рисунок 4 – Оброблене та сегментоване оперативне зображення

1. Початкове високоточне зображення:

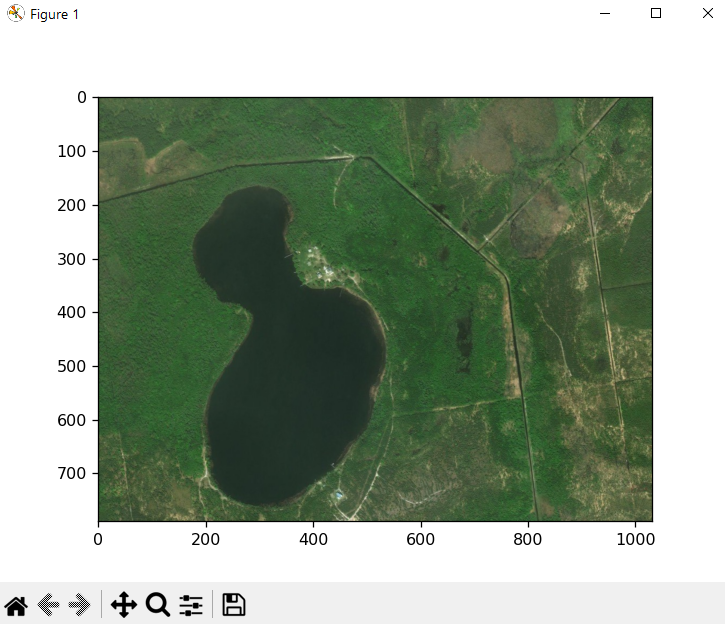


Рисунок 5 – Початкове високоточне зображення

1. Оброблене та сегментоване високоточне зображення:

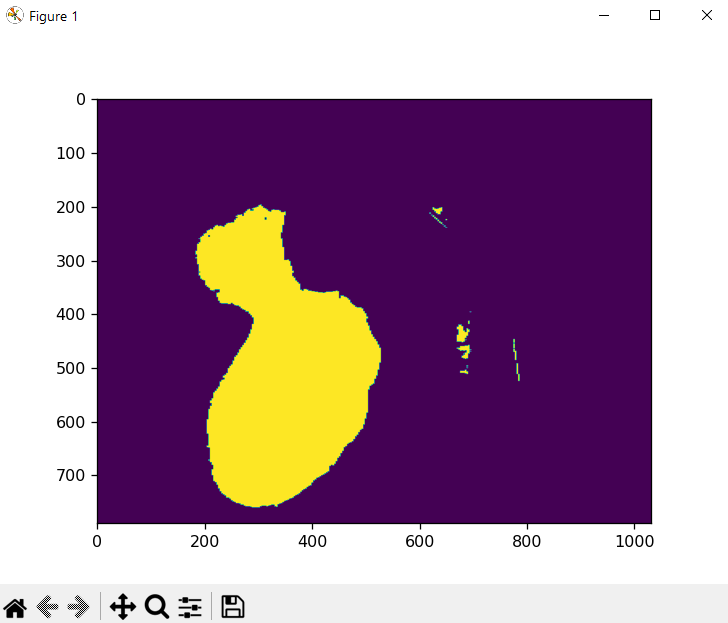


Рисунок 6 – Оброблене та сегментоване високоточне зображення

1. Виділення кутів Гарріса на оперативному зображенні:

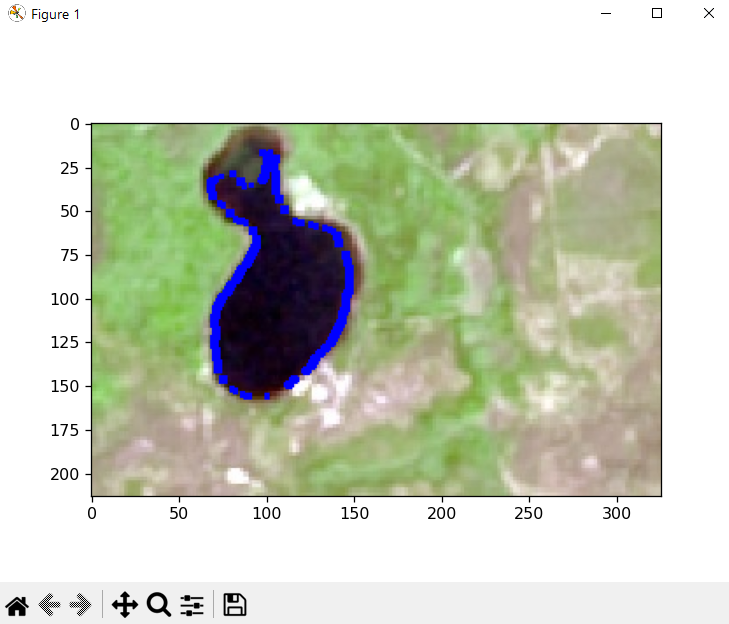


Рисунок 7 – Кути Гарріса на оперативному зображенні

1. Виділення кутів Гарріса на високоточному зображенні:

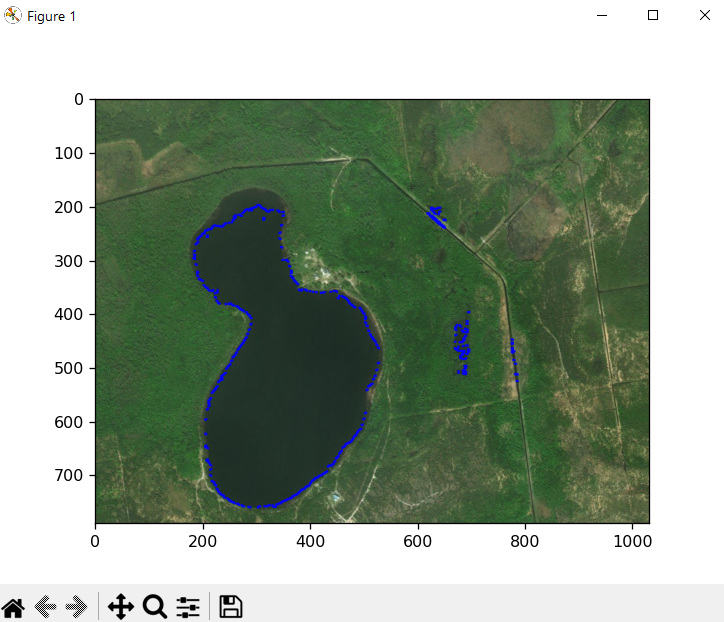


Рисунок 8 – Кути Гарріса на високоточному зображенні

1. Виділення особливих точок дескриптора на оперативному зображенні:

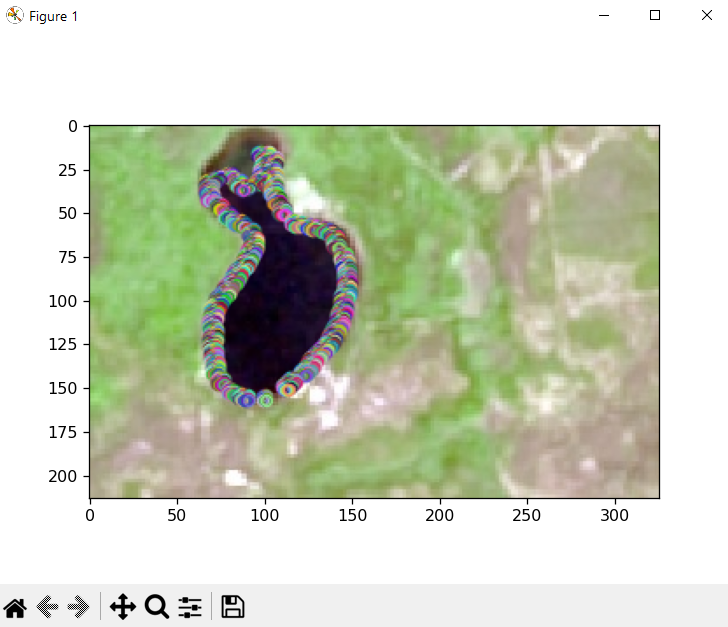


Рисунок 9 – Дескриптор на оперативному зображенні

1. Виділення особливих точок дескриптора на високоточному зображенні:

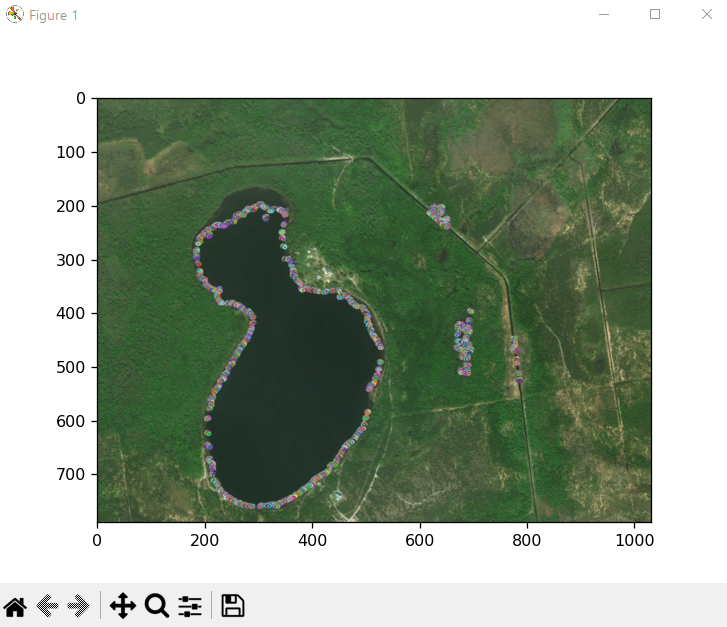


Рисунок 10 – Дескриптор на високоточному зображенні

1. Знаходження збігів по дескрипторах:

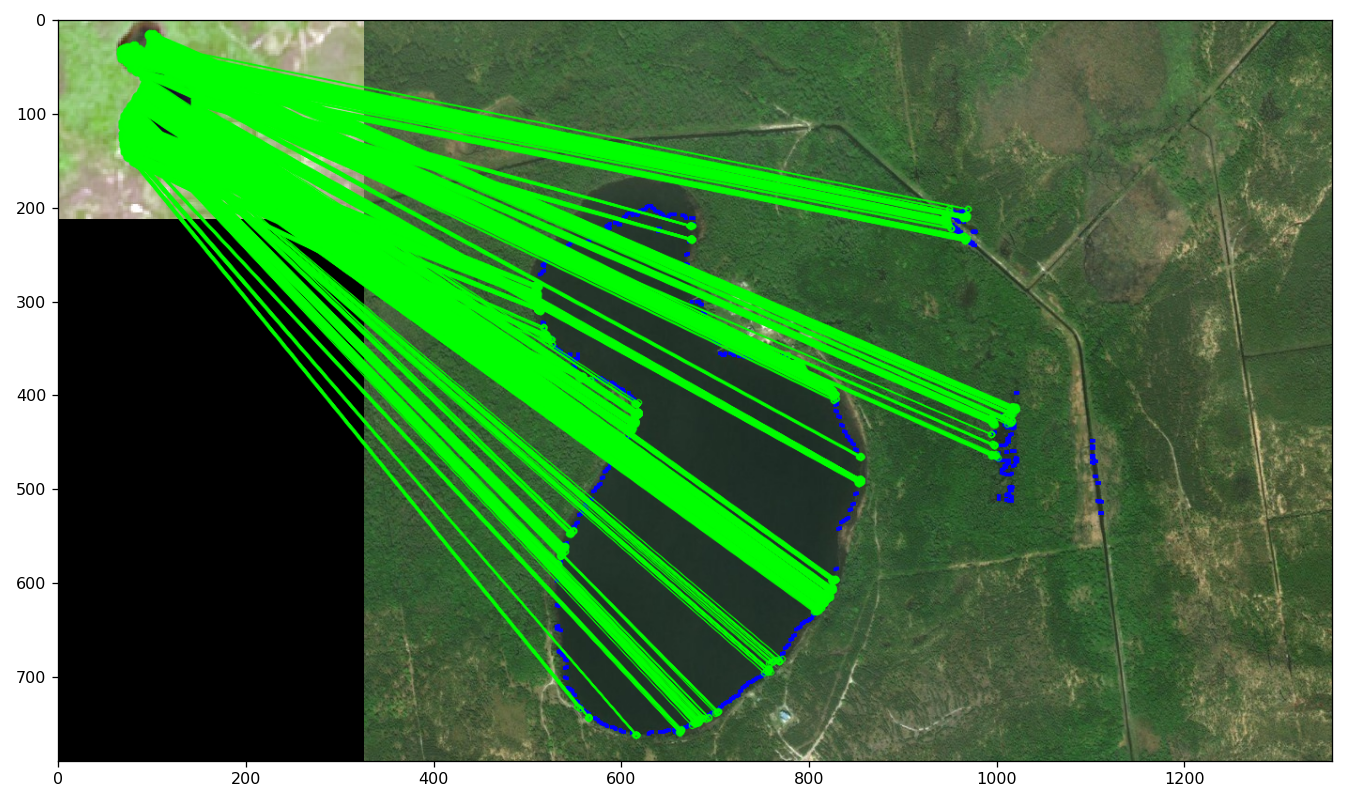


Рисунок 11 – Збіги по дескрипторах зображень

1. Підрахунок результатів:



Рисунок 12 – Результати

Бачимо, що по не всі збіги по дескрипторах є коректними, але загалом, багато дескрипторів утворили правильну пару. Ймовірність ідентифікації склала майже 0.5 (тобто для половини дескрипторів знайдено відповідні їм на іншому зображенні), що є непоганим результатом. На результат доволі сильно вплинула якість зображень, адже при різній якості дескриптори можуть трохи відрізнятися і значенням, і кількістю особливих точок.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-12.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: matplotlib, OpenCV, numpy.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*"""  
Порівняння водойм на зображеннях  
"""  
  
from* matplotlib *import* pyplot *as* plt  
*import* numpy *as* np  
*import* cv2  
  
  
*def* show\_image(image):  
 *""" Показ зображення """* plt.imshow(image)  
 plt.show()  
  
  
*def* image\_read(file\_image):  
 *""" Зчитування зображення з файлу """* image = cv2.imread(file\_image)  
 image\_in\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 *return* image\_in\_rgb  
  
  
*def* image\_processing(image, is\_blurred=*False*):  
 *""" Обробка зображення """  
 # Конвертуємо зображення у відтінки сірого* gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)  
 *# Застосовуємо Гаусівське розмиття, якщо потрібно  
 if* is\_blurred:  
 gray = cv2.GaussianBlur(gray, (5, 5), 5)  
  
 *# Виділяємо темні області на зображенні* \_, threshold\_image = cv2.threshold(gray, 50, 255, cv2.THRESH\_BINARY\_INV)  
  
 *# Застосуємо згладжування* threshold\_image = cv2.medianBlur(threshold\_image, 5)  
  
 *return* threshold\_image  
  
  
*def* image\_contours(image):  
 *""" Знаходження контурів """* contours, \_ = cv2.findContours(image.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 *return* contours  
  
  
*def* save\_result(file\_name, image\_in\_rgb):  
 *""" Збереження результату у файл """* image\_in\_bgr = cv2.cvtColor(image\_in\_rgb, cv2.COLOR\_RGB2BGR)  
 cv2.imwrite(file\_name, image\_in\_bgr)  
  
  
*def* image\_recognition(image, contours):  
 *""" Відображення контурів на зображенні """* cv2.drawContours(image, contours, -1, (0, 255, 0), 2)  
 *return* image  
  
  
*def* segmentation(image):  
 *""" Сегментація з допомогою k-means """  
 # Перетворюємо зображення у двовимірний масив* two\_dimension = image.reshape((-1, 3))  
 two\_dimension = np.float32(two\_dimension)  
  
 *# Критерії зупинки для алгоритму k-means* criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 10, 1.0)  
  
 *# Кількість кластерів та спроб* k = 2  
 attempts = 10  
  
 *# Застосовуємо алгоритм k-means* ret, label, center = cv2.kmeans(two\_dimension, k, *None*, criteria, attempts, cv2.KMEANS\_PP\_CENTERS)  
 center = np.uint8(center)  
  
 *# Отримуємо сегментаційні результати та відновлюємо форму зображення* res = center[label.flatten()]  
 edged\_img = res.reshape(image.shape)  
 *return* edged\_img  
  
  
*def* harris\_corner\_detector(image, segmented\_image):  
 *""" Детектор кутів Гарріса """* floated\_image = np.float32(segmented\_image)  
 *# Застосування детектора кутів Гарріса* dst = cv2.cornerHarris(floated\_image, 2, 3, 0.04)  
  
 *# Диляція результату для позначення кутів* dst = cv2.dilate(dst, *None*)  
  
 *# Поріг для оптимального значення* threshold\_value = 0.01 \* dst.max()image[dst > threshold\_value] = [0, 0, 255]  
  
 *# Отримання координат кутів* key\_points = np.argwhere(dst > threshold\_value)  
 *# Створення ключових точок* key\_points = [cv2.KeyPoint(float(x[1]), float(x[0]), 13) *for* x *in* key\_points]  
 *return* key\_points, image  
  
  
*def* sift\_matching(image1, segmented\_image1, image2, segmented\_image2):  
 *# Знаходження кутів Гарріса для сегментованих зображень* key\_points1, harris\_corner\_image1 = harris\_corner\_detector(image1.copy(), segmented\_image1)  
 key\_points2, harris\_corner\_image2 = harris\_corner\_detector(image2.copy(), segmented\_image2)  
 show\_image(harris\_corner\_image1)  
 show\_image(harris\_corner\_image2)  
  
 *# Створення SIFT детектору* sift = cv2.SIFT\_create()  
  
 *# Обчислення дескрипторів* key\_points1, descriptors1 = sift.compute(segmented\_image1, key\_points1)  
 key\_points2, descriptors2 = sift.compute(segmented\_image2, key\_points2)  
 show\_image(cv2.drawKeypoints(image1, key\_points1, image1))  
 show\_image(cv2.drawKeypoints(image2, key\_points2, image2))  
  
 *# Параметри FLANN* flann\_index\_kdtree = 1  
 index\_params = dict(algorithm=flann\_index\_kdtree, trees=5)  
 search\_params = dict(checks=100)  
 flann = cv2.FlannBasedMatcher(index\_params, search\_params)  
 *# Знаходження збігів* matches = flann.knnMatch(descriptors1, descriptors2, k=2)  
 matches = list(matches)  
  
 *# Фільтрація збігів  
 for* match *in* matches:  
 m, n = match  
 *if* m.distance < n.distance:  
 matches.remove(match)  
  
 *# Параметри зображення збігів* draw\_params = dict(matchColor=(0, 255, 0),  
 singlePointColor=(255, 0, 0),  
 flags=cv2.DrawMatchesFlags\_NOT\_DRAW\_SINGLE\_POINTS)  
 *# Зображення збігів* img3 = cv2.drawMatchesKnn(harris\_corner\_image1, key\_points1, harris\_corner\_image2, key\_points2, matches, *None*, \*\*draw\_params)  
 matches\_number = len(matches)  
 probability = matches\_number / len(key\_points1)  
 *return* img3, matches\_number, probability  
  
  
*# Обробка високоякісного зображення*image\_entrance\_1 = image\_read("low2.jpg")  
show\_image(image\_entrance\_1)  
image\_exit = image\_processing(image\_entrance\_1.copy(), *True*)  
img\_segment\_1 = segmentation(image\_exit)  
show\_image(img\_segment\_1)  
  
*# Обробка низькоякісного зображення*image\_entrance\_2 = image\_read("high2.jpg")  
show\_image(image\_entrance\_2)  
image\_exit = image\_processing(image\_entrance\_2.copy(), *False*)  
img\_segment\_2 = segmentation(image\_exit)  
show\_image(img\_segment\_2)  
  
result, matches\_number, probability = sift\_matching(image\_entrance\_1, img\_segment\_1, image\_entrance\_2, img\_segment\_2)  
print("Кількість збігів особливих точок =", matches\_number)  
print("Ймовірність ідентифікації =", probability)  
show\_image(result)  
save\_result("result.jpg", result)

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження принципів та особливостей практичного застосування технологій порівняння цифрових зображень для ідентифікації об’єктів з використанням спеціалізованих програмних бібліотек. Було реалізовано алгоритм, що покращує вхідне зображення, виконує кластеризацію та сегментацію, з допомогою детектора Гарріса знаходить особливі точки на зображеннях, обчислює дескриптори з допомогою SIFT та знаходить збіги дескрипторів з допомогою FLANN і зображує їх на результуючому зображенні з використанням засобів мови програмування Python. Було проведено порівняння водойми (озера) на зображеннях різної якості та визначено, що ймовірність ідентифікації складає майже 0.5.

Виконав: студент Трикош І. В.