**Міністерство освіти і науки України**

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

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №5**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ СЕГМЕНТАЦІЇ ТА КЛАСТЕРИЗАЦІЇ**

**ЦИФРОВИХ ЗОБРАЖЕНЬ ДЛЯ ЗАДАЧ COMPUTER VISION**

**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 сегментації та кластеризації цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

## **ІІ. Завдання:**

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: водойми

## **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 відбувається пошук контурів на сегментованому зображенні. Знайдений контур і буде ідентифікованим об’єктом.

Кластеризація k-means – це один з методів кластеризації, що впорядковує множини об’єктів в однорідні групи. Цей метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто мінімізується функція

де – метрика, – об’єкт даних, – центр кластера, якому належить об’єкт на j-ій ітерації.

Сегментація відбувається наступним чином: кожному пікселю присвоюється кластерний центр, до якого він належить, а далі масив з пікселями перетворюється у форму початкового зображення.

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 зображення з допомогою бібліотеки OpenCV. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується попередня обробка зображень. 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 зображення з допомогою k-means.

У блоці 4 виконується виділення зовнішніх контурів об’єктів.

У блоці 5 виконується відображення контурів на початковому зображенні.

У блоці 6 відображається результат виділення контурів.

У блоці 7 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.

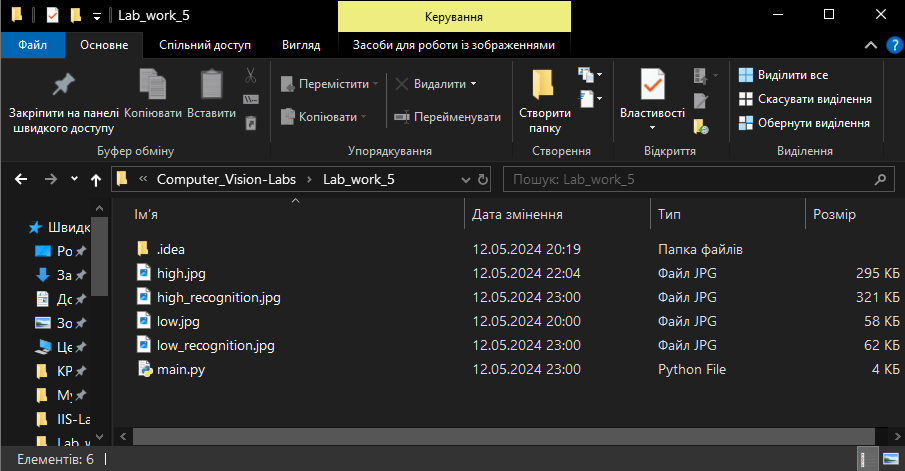


Рисунок 2 – Структура проєкту

Lab\_work\_5 – головний каталог проєкту

main.py – файл програмного коду

high.jpg – високоточний знімок

high\_recognition.jpg – результат виділення контурів об’єктів на високоточному знімку

low.jpg – оперативний знімок

low\_recognition.jpg – результат виділення контурів об’єктів на оперативному знімку

* 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 високоточне зображення:

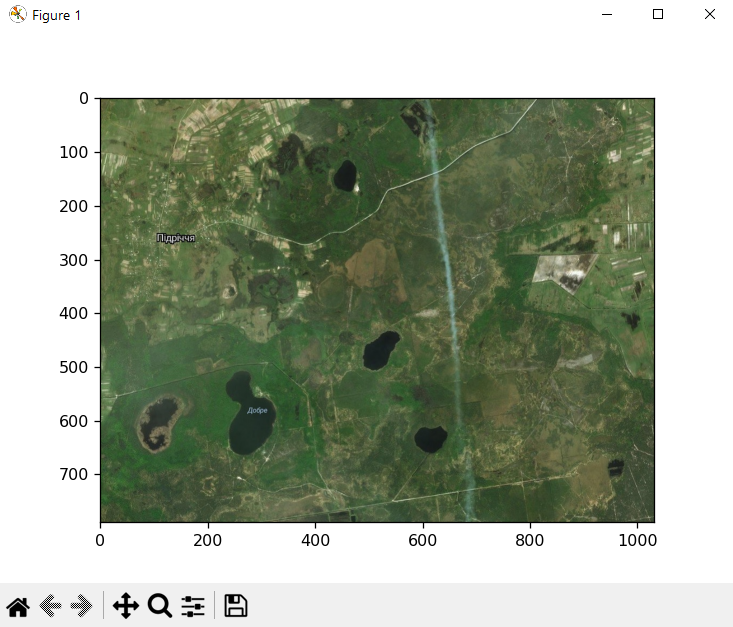


Рисунок 3 – Початкове високоточне зображення

1. Оброблене високоточне зображення:

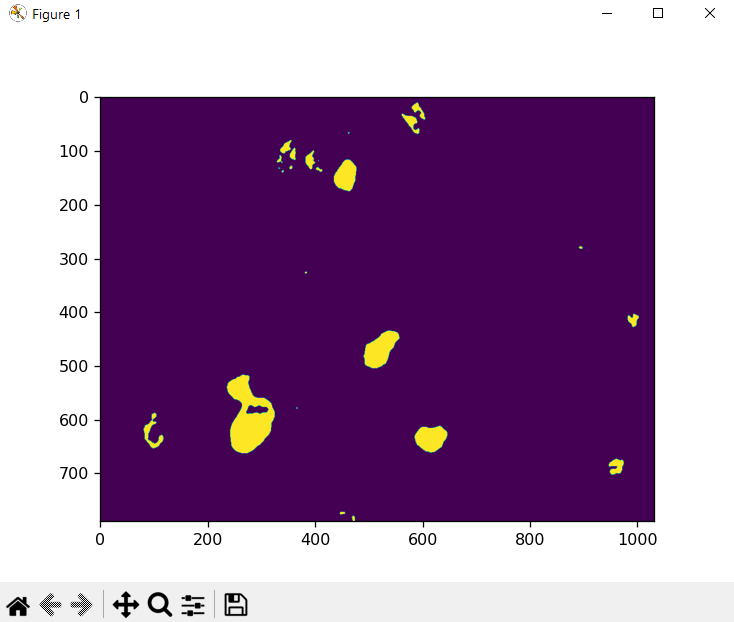


Рисунок 4 – Оброблене високоточне зображення

1. Сегментоване високоточне зображення:

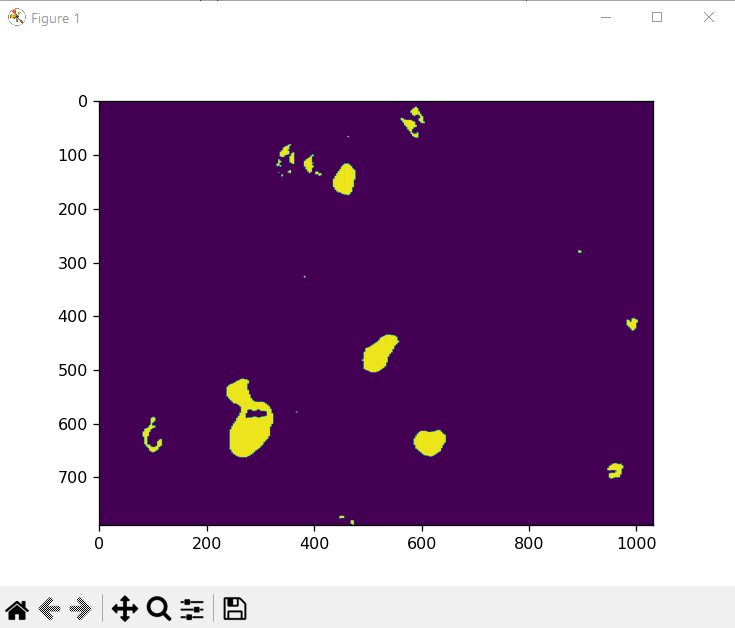


Рисунок 5 – Сегментоване високоточне зображення

1. Виділення контурів на високоточному зображенні:

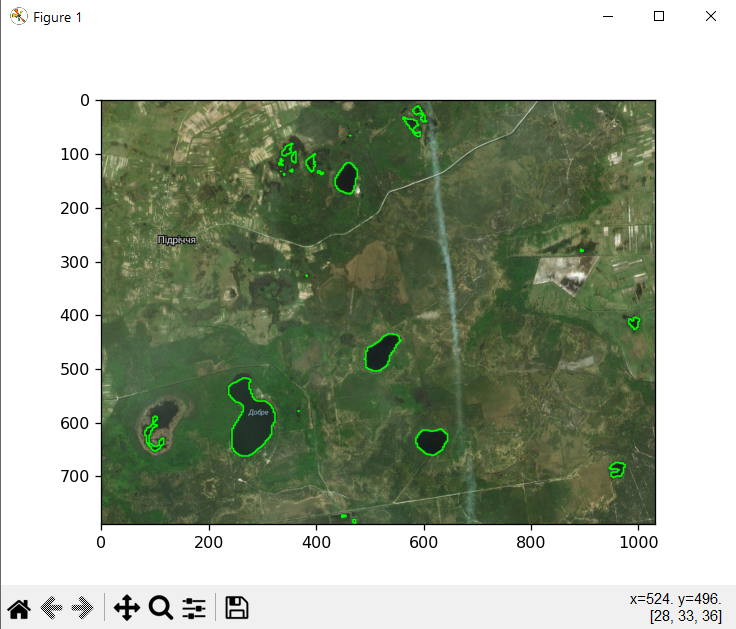


Рисунок 6 – Виділені контури на високоточному зображенні

Бачимо, що озера виділились доволі добре, проте також виділились деякі темні ділянки лісу. Тепер обробимо оперативне зображення:

1. Початкове оперативне зображення:

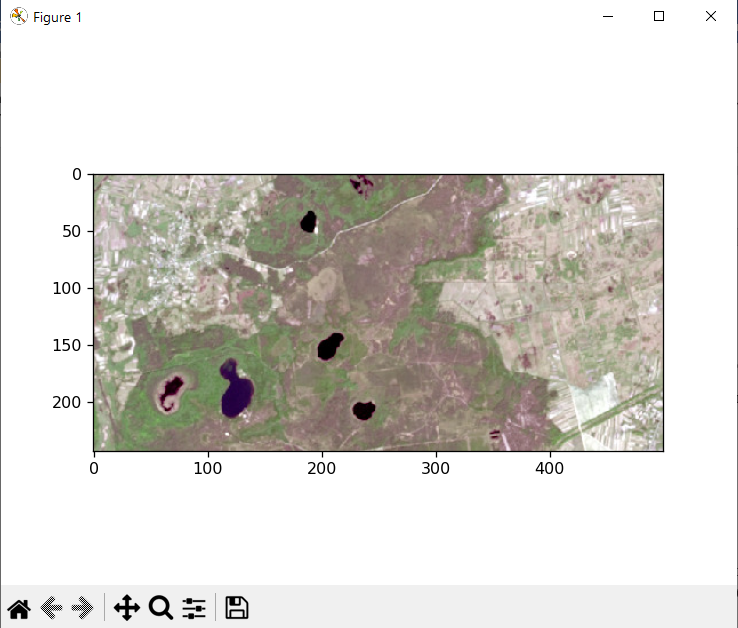


Рисунок 7 – Початкове оперативне зображення

1. Оброблене оперативне зображення:

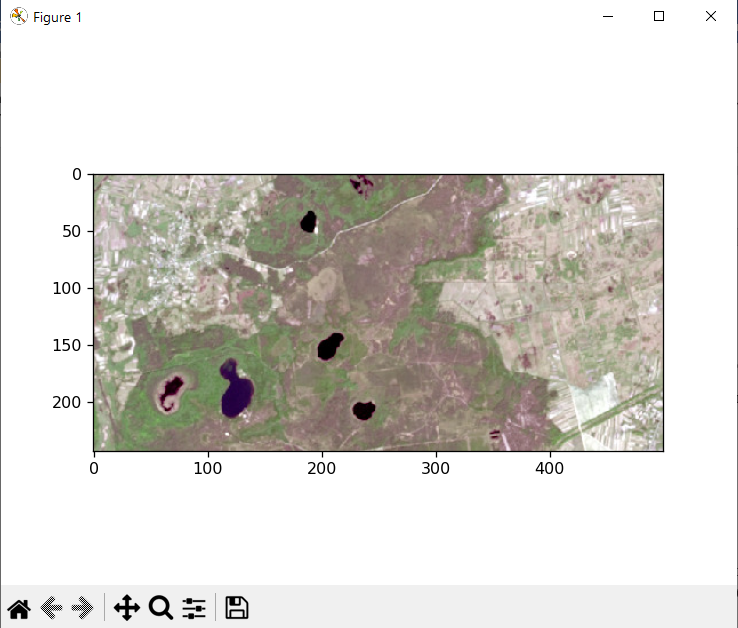


Рисунок 8 – Оброблене оперативне зображення

1. Сегментоване оперативне зображення:

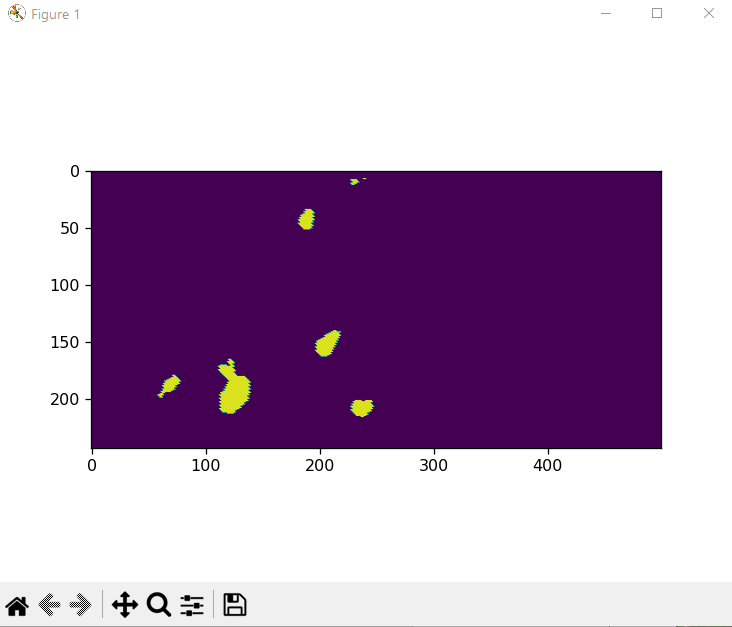


Рисунок 9 – Сегментоване оперативне зображення

1. Виділення контурів на оперативному зображенні:

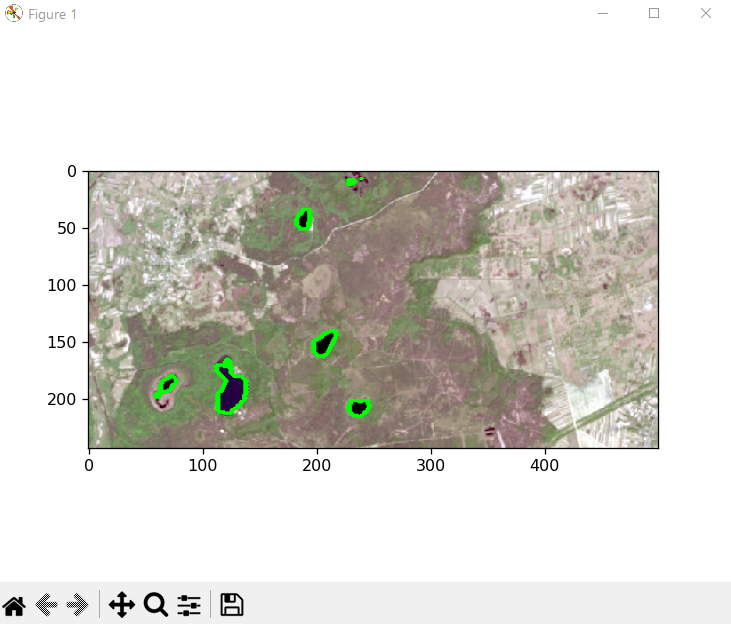


Рисунок 10 – Виділені контури на оперативному зображенні

Бачимо, що на оперативному зображенні контури водойм виділились навіть краще. Це через те, що на цьому знімку ліс позначено однотонно. Тому загалом, представлені результати відповідають завданню лабораторної роботи.

* 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.3-10.

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

*"""  
Ідентифікація водойм на зображеннях різної якості  
"""  
  
import* cv2  
*import* numpy *as* np  
*from* matplotlib *import* pyplot *as* plt  
  
  
*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  
  
  
*# Обробка високоякісного зображення*image\_entrance = image\_read("high.jpg")  
show\_image(image\_entrance)  
image\_exit = image\_processing(image\_entrance, *True*)  
show\_image(image\_exit)  
img\_segment = segmentation(image\_exit)  
show\_image(img\_segment)  
img\_contours = image\_contours(img\_segment)  
result = image\_recognition(image\_entrance, img\_contours)  
save\_result("high\_recognition.jpg", result)  
show\_image(result)  
  
  
*# Обробка низькоякісного зображення*image\_entrance = image\_read("low.jpg")  
show\_image(image\_entrance)  
image\_exit = image\_processing(image\_entrance, *False*)  
show\_image(image\_exit)  
img\_segment = segmentation(image\_exit)  
show\_image(img\_segment)  
img\_contours = image\_contours(img\_segment)  
result = image\_recognition(image\_entrance, img\_contours)  
save\_result("low\_recognition.jpg", result)  
show\_image(result)

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження принципів та особливостей практичного застосування технологій сегментація та кластеризації цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек. Було реалізовано алгоритм, що покращує вхідне зображення, виконує кластеризацію та сегментацію і знаходить та позначає контури водойм на зображенні з використанням засобів мови програмування Python.

Виконав: студент Трикош І. В.