Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №5

з дисципліни «Технології Computer Vision»

на тему

«ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ СЕГМЕНТАЦІЇ ТА КЛАСТЕРИЗАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ ДЛЯ ЗАДАЧ COMPUTER VISION»

Виконала:
студентка групи ІС-12
Павлова Софія

Перевірив:
Баран Д. Р.

Київ – 2024

1. Постановка задачі

Мета роботи:

Дослідити принципи та особливості практичного застосування технологій сегментації та кластеризації цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

Завдання II рівня:

Здійснити виконання завдання лабораторної роботи (**кольорова кластеризація**, **виявлення контурів** та **ідентифікація об'єктів**), ідентифікацію об'єктів реалізувати шляхом ***програмного порівняння контурів***.

1-6	1. Оперативні: https://apps.sentinel-hub.com/eo-browser/?zoom=14&lat=52.04212&lng=29.27444&themeId=WILDFIRES-NORMAL-MODE&visualizationUrl=https%3A%2F%2Fservices.sentinel-hub.com%2Fogc%2Fwms%2Faae18701-6b25-4001-8b2a-b98a1b3806c1&datasetId=S2L2A&fromTime=2022-03-16T00%3A00%3A00.000Z&toTime=2022-03-16T23%3A59%3A59.999Z&layerId=1_FALSE-COLOR 2. Високоточні: https://www.google.com.ua/maps	Район спостереження – обрати самостійно. Об'єкти ідентифікації – обрати самостійно. Дата оперативних даних – обрати самостійно. Метод і технологія кластеризації / сегментації – повинні забезпечувати можливість розрізнення та ідентифікацію обраних об'єктів спостереження.

2. Виконання

2.1. Постановка задачі

Сформулюємо задачу, подібну до попередньої лабораторної роботи.

Задача:

Необхідно ідентифікувати великі за площою поселення міського типу з супутникових знімків.

Об’єкти ідентифікації – міська забудова.

2.2. Імпорт даних

Використаємо оперативні дані ДЗЗ (дистанційне зондування Землі) з сайту https://apps.sentinel-hub.com/ .

За *параметри пошуку зображення* візьмемо (з попередньої лабораторної):

Район: поселення Lazio поблизу Риму, Італія.

Дата: 29.04.2024

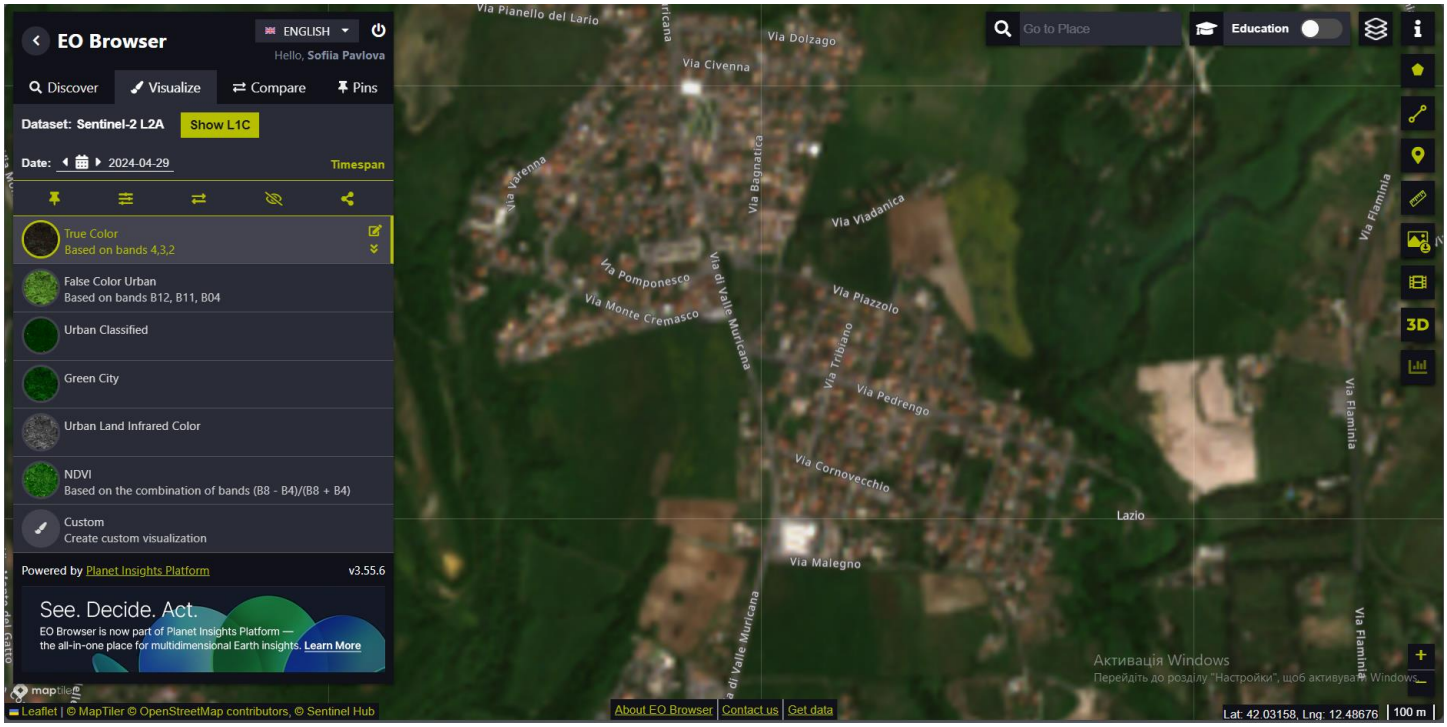

Рисунок 1 – Експорт даних ДЗЗ

2.3. Кольорова кластеризація

Задача програми – покращити фото так, щоб на вихідному фото лишилась лише міська забудова.

Для цього використаємо **метод аналізу кольорових просторів** за допомогою **HSV** (Hue, Saturation, Value) або ж *(відтінок, насиченість, яскравість)*.

Алгоритм програми

Програма буде використовувати маску для порівняльного аналізу кольорових просторів усього зображення та цікавого для нас фрагмента, відсікати непотрібний діапазон яскравості та прибирати шуми за допомогою блюру.

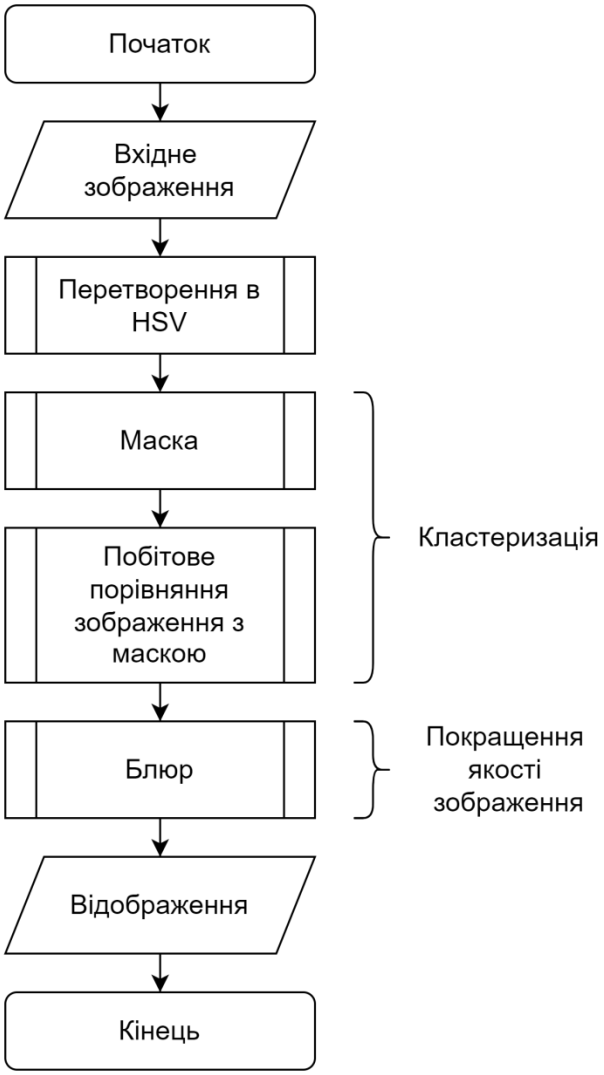

Рисунок 2 – Блок схема програми

Програмна реалізація

Експериментально визначимо параметри міської забудови для *!конкретного* зображення в HSV. Найкращими виявились:

```
# Визначення об'єктів міської забудови в HSV
lower_city = np.array([0, 10, 0])
upper_city = np.array([25, 170, 120])
```

Створимо маску та відфільтруємо непотрібну інформацію.

Лістинг коду:

```
import cv2
import numpy as np
import imutils

# Перетворення зображення в HSV
def img_to_hsv(image_name):
    image = cv2.imread(image_name)
    image = imutils.resize(image, width=600)
    # Перетворення в HSV
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    return image, hsv

# Кольорова кластеризація по HSV гістограмі яскравості
def color_clasterising(image, hsv):
    '''
    HSV (відтінок, насиченість, яскравість)

        відтінок (0-360) - 0 червоний, 120 зелений, 240 синій
        насиченість (0-255)
        яскравість (0-255)
    '''

    # Визначення об'єктів міської забудови в HSV
    lower_city = np.array([0, 10, 0])
    upper_city = np.array([25, 170, 120])
    # Створення маски
    mask = cv2.inRange(hsv, lower_city, upper_city)

    # Побітове порівняння кольорового образу - "маски" із поточними кадрами
    img_bitwise_and = cv2.bitwise_and(image, image, mask=mask)
    # Median blurring
    img_median_blurred = cv2.medianBlur(img_bitwise_and, 5)

    return mask, img_median_blurred

# Візуалізація результатів
def print_result(image, contour_image, img_median_blurred, img_identification):
    # Відображення вхідного зображення, зображення з контурами та обробленого зображення
    cv2.imshow('Input Image', image)
    [...]
    cv2.imshow('Segment Image', img_median_blurred)

# Блок головних викликів
if __name__ == "__main__":
```

```
image_name = 'Sentinel-2_L2A_True_Color.jpg'

# Перетворення зображення в HSV
image, hsv = img_to_hsv(image_name)

[...]

# Сегментація і виділення контурів
contour_image = contours(image, mask)
```

Результат:

Бачимо, що за таких параметрів алгоритм **доволі точно кластеризує міську забудову**. Однак, усе одно в силу схожості пікселів, програма кластеризує деякі поля до міста.

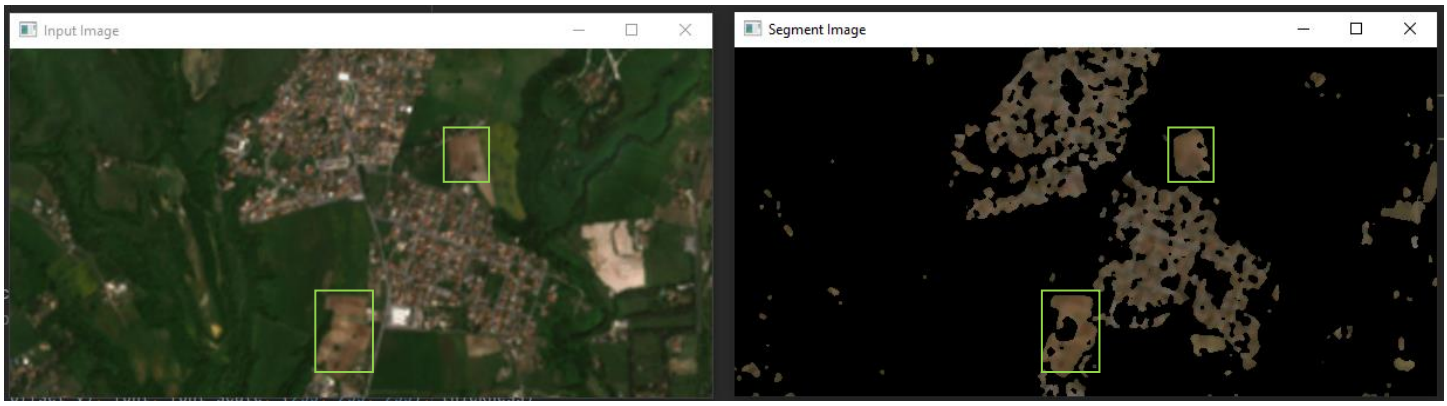

Рисунок 3 – Результат покращення зображення та кольорової кластеризації

2.4. Виявлення контурів

Виконаємо операцію виділення контурів на основі визначеної раніше маски.

Алгоритм програми

Програма буде використовувати функцію бібліотеки $OpenCV$: $findContours()$, яка буде виявляти лише зовнішні контури, які задаються двома точками. Тобто ті, які не містяться всередині інших контурів.

.

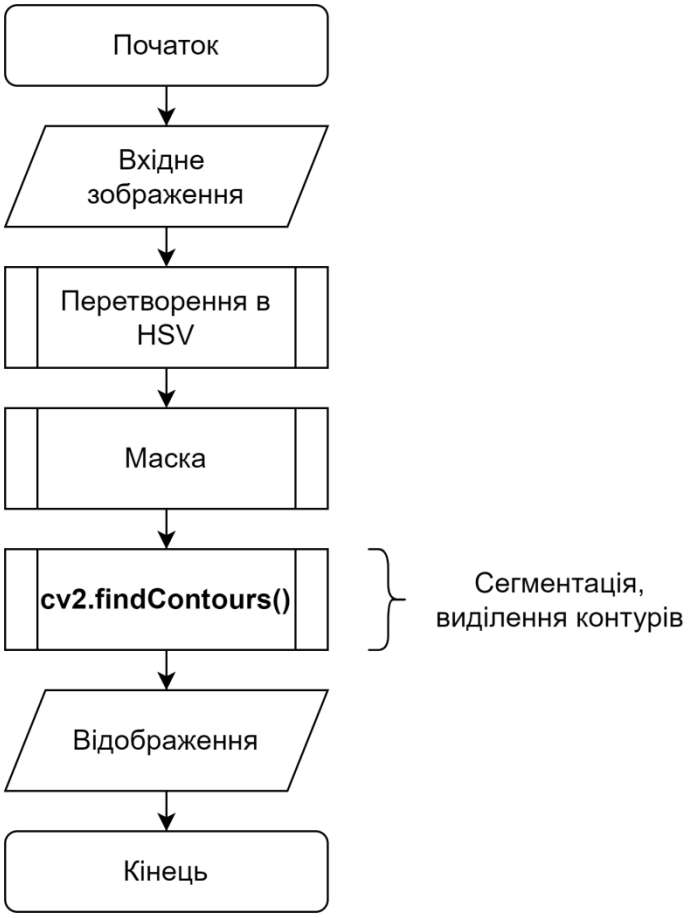

Рисунок 4 – Блок схема програми

Програмна реалізація

Лістинг коду:

```
[...]

# Сегментація і виділення контурів
def contours(image, mask):
    # Виявлення контурів
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Малювання контурів на вхідному зображенні
    contour_image = image.copy()
    cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)

    return contour_image

# Візуалізація результатів
def print_result(image, contour_image, img_median_blurred, img_identification):
    # Відображення вхідного зображення, зображення з контурами та обробленого зображення
    cv2.imshow('Input Image', image)
    cv2.imshow('Contours', contour_image)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
# Блок головних викликів
if __name__ == "__main__":
    [...]
    # Сегментація і виділення контурів
    contour_image = contours(image, mask)
```

Результат:

У результаті бачимо, що програма **доволі точно визначає контури як великих, так і малих скупчень об'єктів** міської забудови.

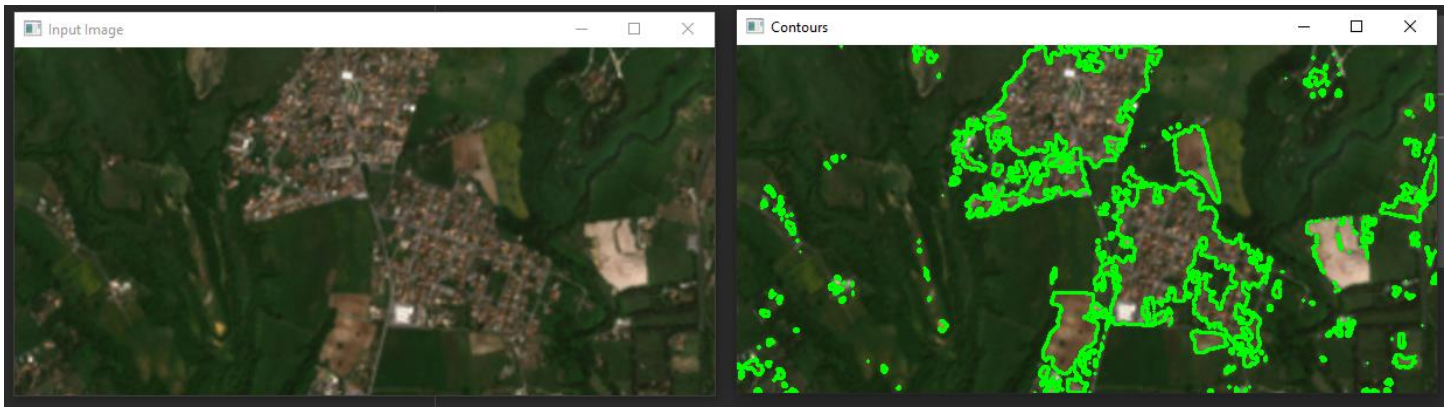

Рисунок 5 – Сегментація та виділення контурів усіх об'єктів міської забудови

2.5. Ідентифікація об'єктів

Наша задача – ідентифікувати великі поселення людей. Для цього порівняємо площі сегментованих об'єктів і для найбільших поселень задамо лінгвістичний ідентифікатор.

Алгоритм програми

Використаємо функцію $OpenCV : dilate()$ для того, щоб об'єднати суміжні сегментовані об'єкти в одну групу. Після чого повторно виявимо контури.

Обчислимо площі виявлених поселень і відобразимо лише ті, які входять у 95% поселень з найбільшою площою.

Для ідентифікації розрахуємо **центри мас контурів** найбільших поселень і пронумеруємо їх від 1 до n, де 1 – найбільше поселення.

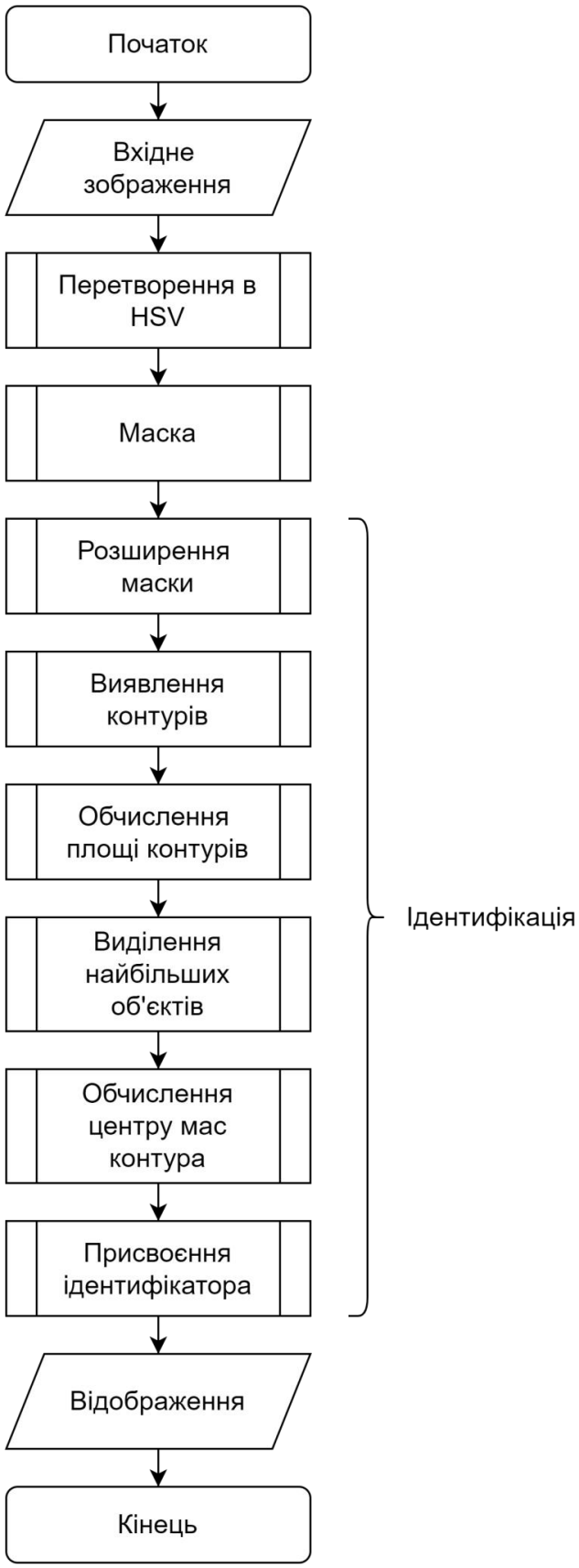

Рисунок 6 – Блок схема програми

Програмна реалізація

Використаємо **програмне порівняння** контурів на основі їх площ описане в алгоритмі.

Для більш точного розрахунку контурів поселень, розширимо маску за допомогою ядра $kernel$ $(5, 5)$ у 2 ітерації. Саме такі параметри дають максимально чітко згруповані селища. *При вказуванні меншого ядра, великі селища ідентифікуються як купка менших.*

А для красивого «лейблування» ідентифікованих об’єктів, використаємо центр мас контуру щоб зпозиціонувати ідентифікатор (текст) по центру об’єкту.

Лістинг коду:

```
[...]

# Ідентифікація великих поселень
def identification(image, mask):
    # Розширення маски
    kernel = np.ones((5, 5), np.uint8)
    mask_dilated = cv2.dilate(mask, kernel, iterations=2)
    # Виявлення контурів
    contours_identification, _ = cv2.findContours(mask_dilated, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    # Обчислення площ контурів
    areas = [cv2.contourArea(contour) for contour in contours_identification]
    # Визначення порогової площі за квантилем 0.95
    threshold_area = np.quantile(areas, 0.95)

    # Малювання всіх контурів
    img_identification = image.copy()
    i = 0
    for contour in contours_identification:
        # Ідентифікація об'єкта, якщо його площа більша за 95%
        if cv2.contourArea(contour) > threshold_area:
            i += 1
            cv2.drawContours(img_identification, [contour], -1, (0, 255, 0), 2)
            # Обчислення моменту контуру
            M = cv2.moments(contour)
            # Ценрт масс контуру
            if M['m00'] != 0:
                cX = int(M['m10'] / M['m00'])
                cY = int(M['m01'] / M['m00'])
                # Отримання розмірів тексту
                text = f'City {i}'
                font = cv2.FONT_HERSHEY_SIMPLEX
                font_scale = 0.5
                thickness = 1
                text_size, _ = cv2.getTextSize(text, font, font_scale, thickness)
                # Зміщення позиції тексту, щоб його центр співпадав з центром об'єкту
                text_offset_x = cX - text_size[0] // 2
                text_offset_y = cY + text_size[1] // 2
                cv2.putText(img_identification, text, (text_offset_x, text_offset_y),
font, font_scale, (255, 255, 255), thickness)

    return img_identification
```

```python
# Візуалізація результатів
def print_result(image, contour_image, img_median_blurred, img_identification):
    # Відображення вхідного зображення, зображення з контурами та обробленого зображення
    cv2.imshow('Input Image', image)
    [...]
    cv2.imshow('Identification', img_identification)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Блок головних викликів
if __name__ == "__main__":
    [...]
    # Ідентифікація
    img_identification = identification(image, mask)

    # Результати
    print_result(image, contour_image, img_median_blurred, img_identification)
```

Результат:

У результаті бачимо, що програма **чудово ідентифікувала великі селища** на зображенні з супутника і присвоїла їм ідентифікатори в залежності від їх площі.

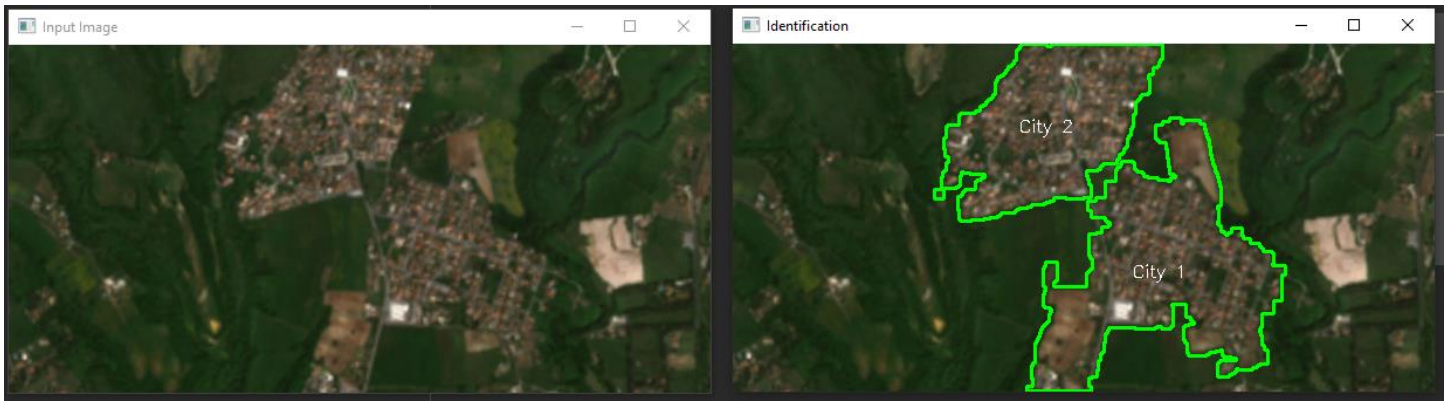

Рисунок 7 – Ідентифікація великих поселень

2.6. Аналіз отриманих результатів

Покращено якість зображення та виконану кольорову кластеризацію об’єктів міської забудови – підтверджено рисунком.

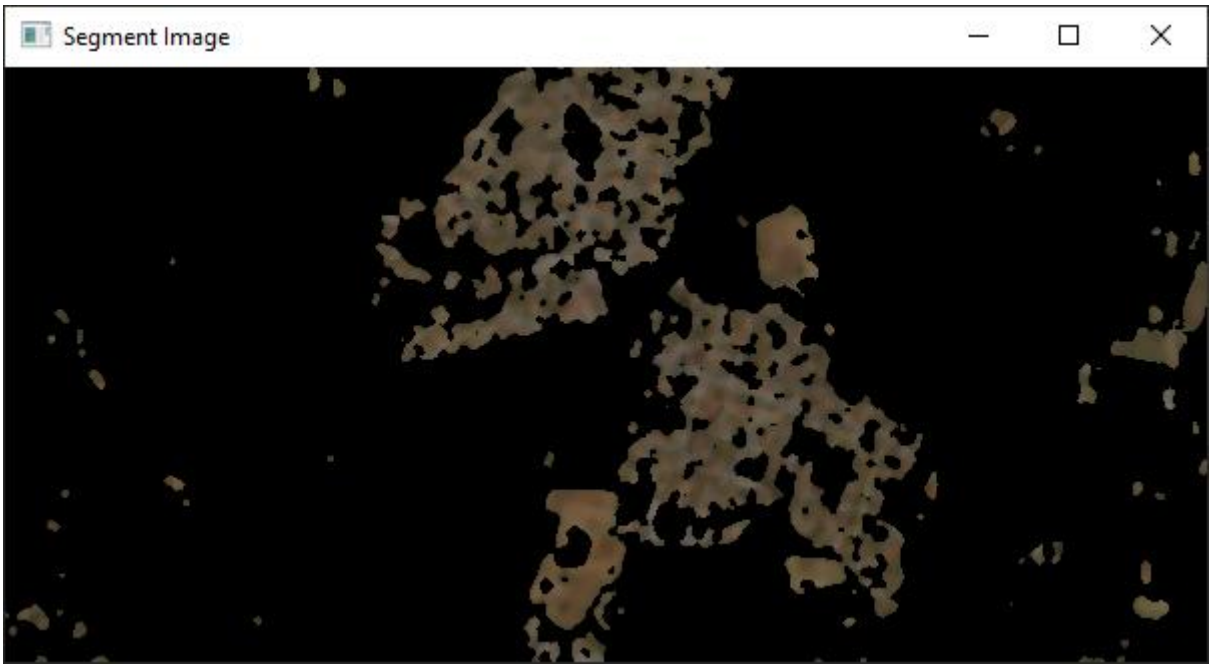

Рисунок 8 – Підтвердження покращення якості та кластеризації зображення

Сегментовано об’єкти міської забудови та виділено контури скупчень об’єктів міської забудови різної щільності – підтверджено рисунком.

Рисунок 9 – Підтвердження сегментації та виділення контурів

Ідентифіковано великі поселення – підтверджено рисунком.

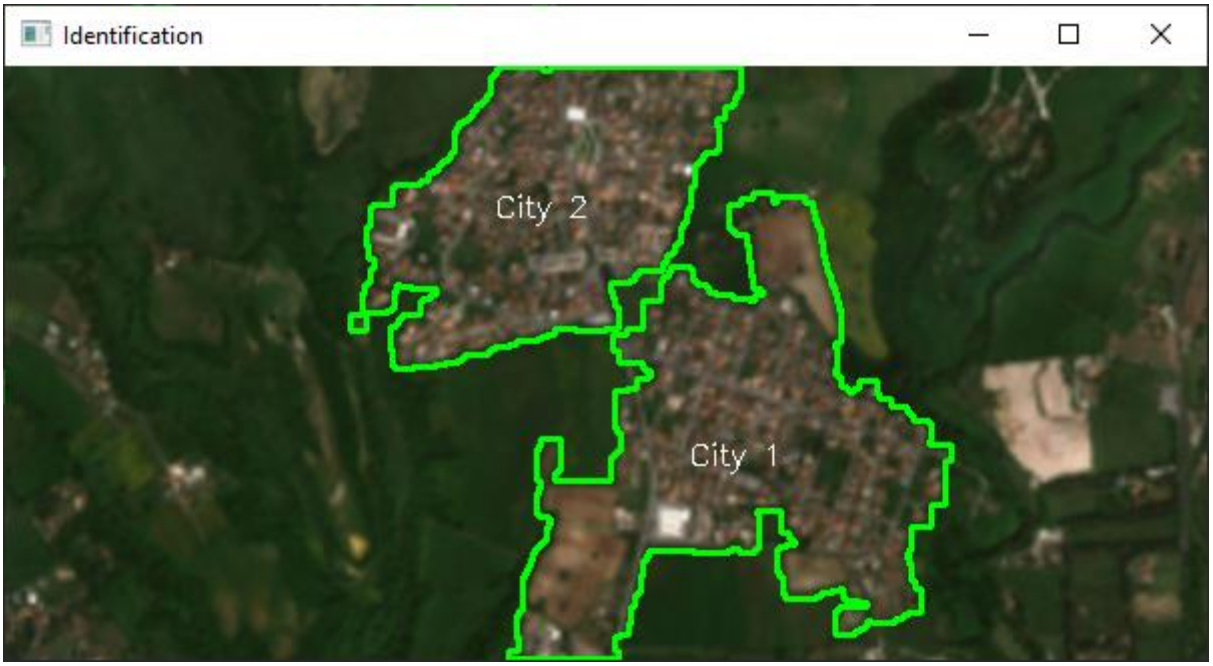

Рисунок 10 – Підтвердження ідентифікації великих поселень

Висновок:

У результаті виконання лабораторної роботи отримано практичні та теоретичні навички роботи з даними дистанційного зондування Землі, а саме фото з супутників.

Отримано досвід кольорової кластеризації, сегментації об’єктів на фото, виділення контурів шуканих об’єктів та їх ідентифікації на основі аналізу кольорових просторів зображення у HSV.

Реалізовано програмний скрипт який виконує усі розглянуті функції.

За допомогою такого скрипта можна ідентифікувати великі поселення.