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

Кафедра інформаційних систем та технологій

Лабораторна робота №4

з дисципліни «Технології Computer Vision»

на тему

«ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ПОКРАЩЕННЯ ЯКОСТІ 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 покращення якості цифрових зображень для задач Computer Vision з використанням спеціалізованих програмних бібліотек.

Завдання II рівня:

Здійснити виконання завдання лабораторної роботи (покращення якості цифрового зображення) для відеопотоку за варіантами таблиці додатку.

3	Дані ДЗЗ	Об’єкти міської забудови

2. Виконання

2.1. Імпорт даних

За варіантом необхідно працювати з даними ДЗЗ (дистанційне зондування Землі). Тому візьмемо дані з сайту https://apps.sentinel-hub.com/ .

Авторизуємось і матимемо змогу вивантажувати зображення та відео з даного ресурсу.

Імпорт зображення

Для імпорту зображення необхідно вибрати «*Download image*».

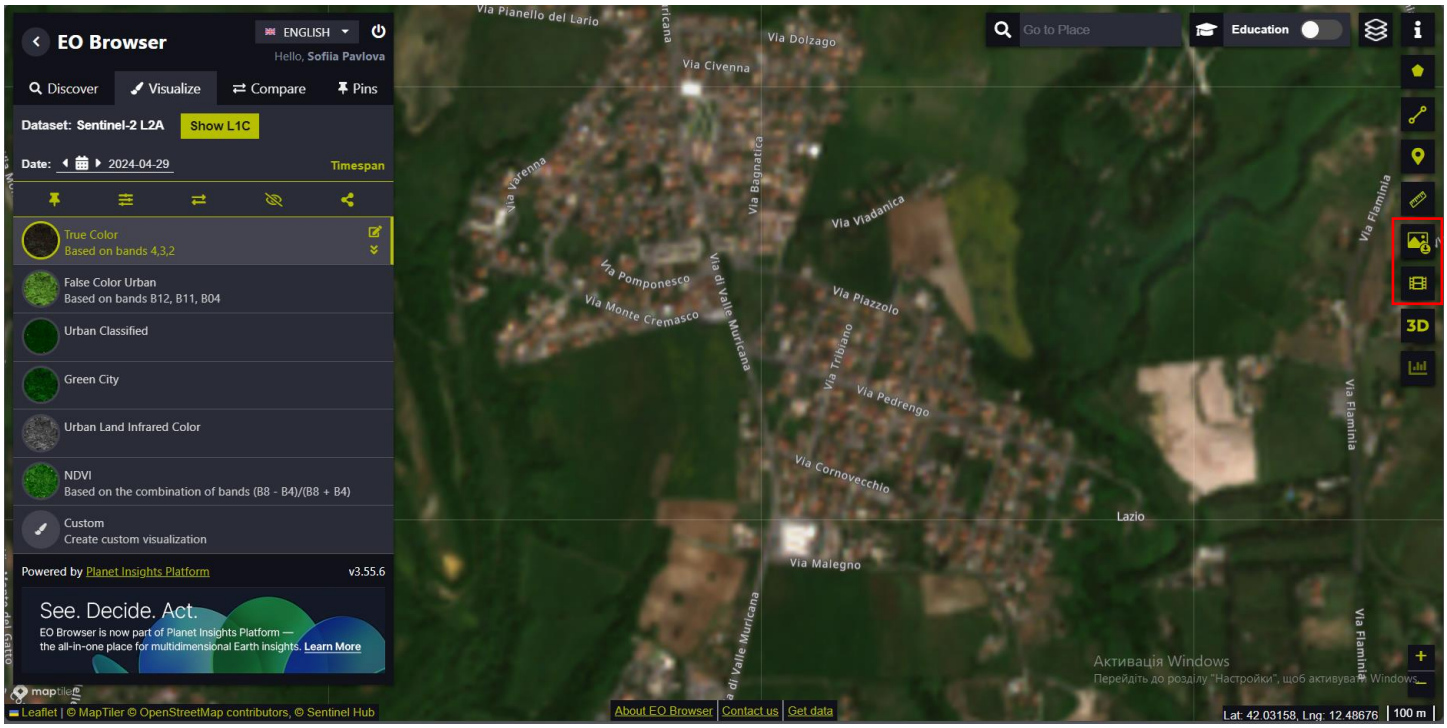

Рисунок 1 – Інструменти для імпорту фото/відео

Імпорт відео

Для імпорту відео необхідно вибрати «*Create timelapse animation*». Після чого обрати хмарність та період, за яким будуть фільтруватись зображення з супутника. Серед знайдених зображень можна включити, або виключити деякі з анімації.

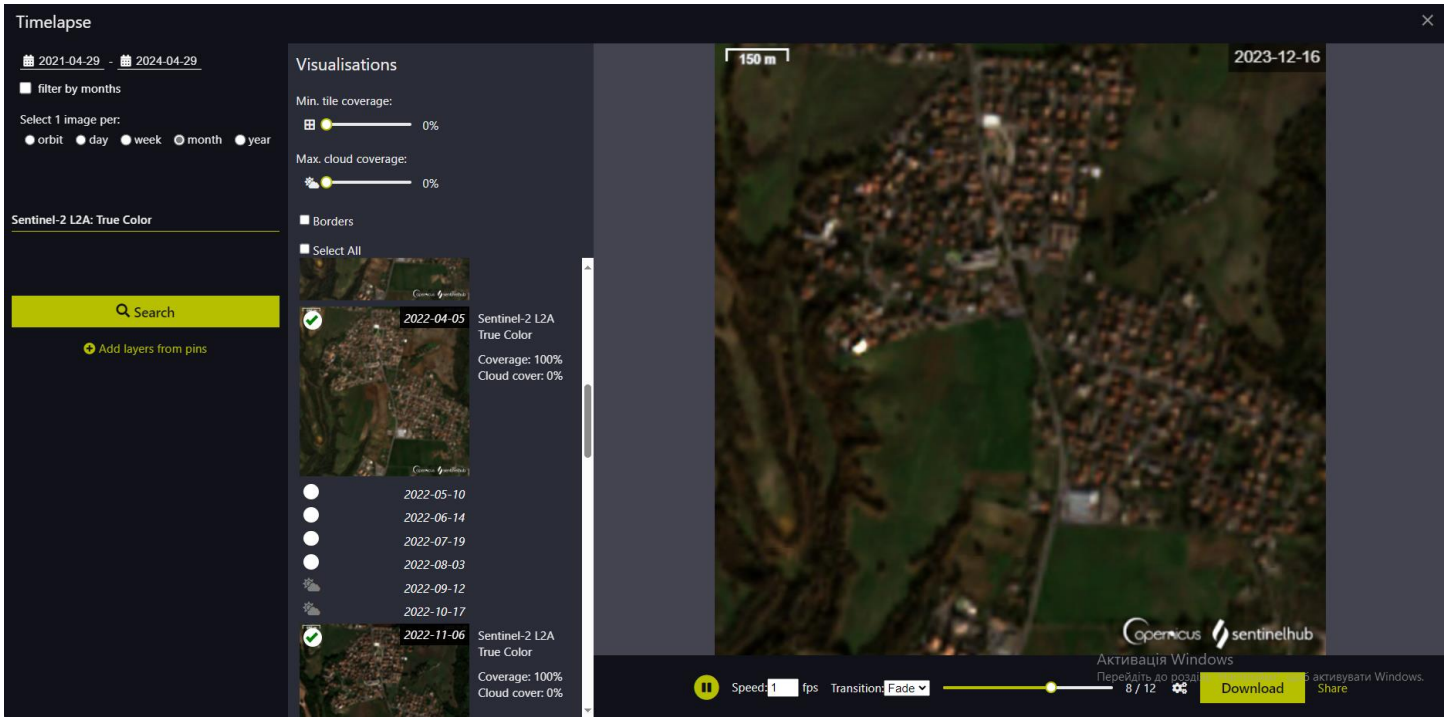

Рисунок 2 – Налаштування параметрів імпорту відео

2.2. Визначення характеристик об’єктів міської забудови

Задача програми – покращити якість відео так, щоб на вихідному відео можна було легко сегментувати міську забудову.

Для цього раціонально використати корекцію яскравості кожного кадру відео потоку. Утім для цього потрібно аналізувати гістограму яскравості кожного кадру. Це довго і неефективно.

Щоб цього не робити, можна попередньо провести аналіз яскравості на одному кадрі, а потім застосувати відшуканий «фільтр» для всього відео.

Напишемо окрему програму для аналізу фото.

Алгоритм програми

Програма буде використовувати маску для порівняльного аналізу гістограм яскравості усього зображення та цікавого для нас фрагмента, відсікати непотрібний діапазон яскравості та нормалізувати отриману гістограму.

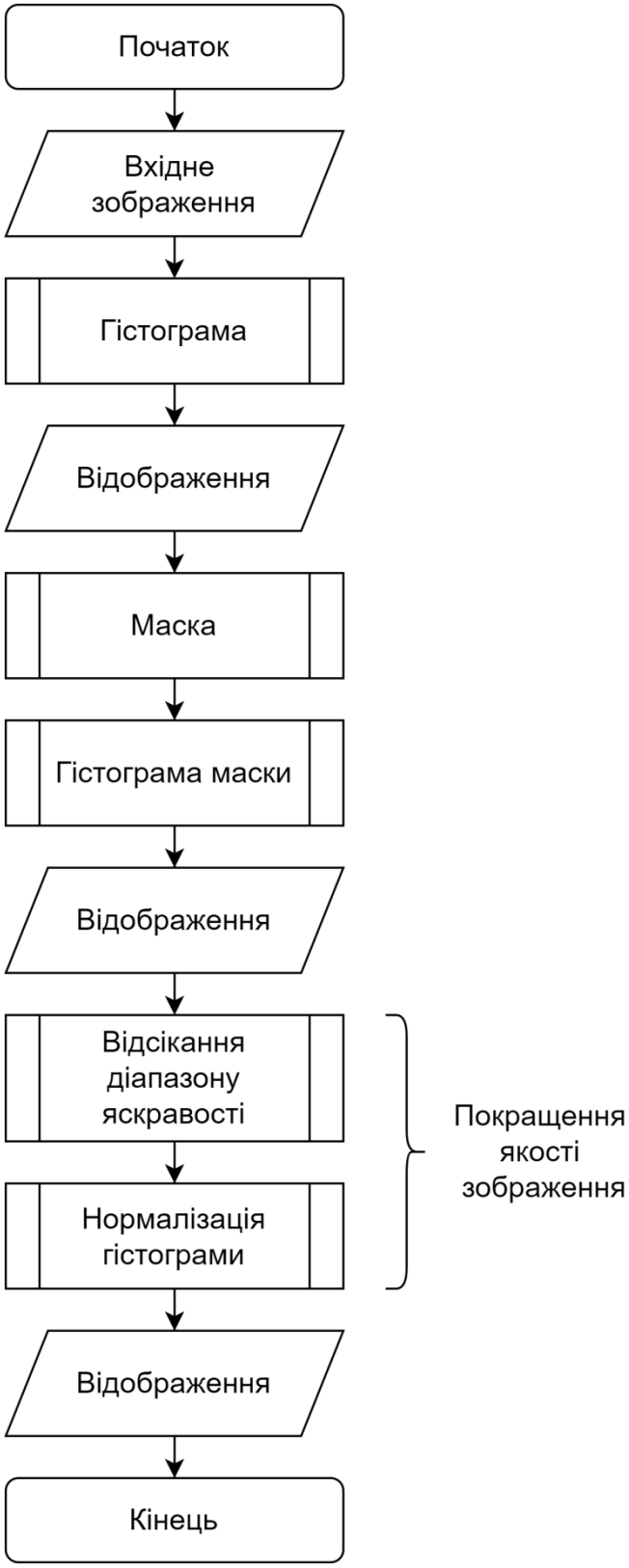

Рисунок 3 – Блок схема програми

Математична модель (Покращення якості зображення)

Спотворене зображення – має шум в значеннях яскравості та в геометрії пікселей:

$$f(x,y) = m(x,y) \cdot Q \cdot s(x,y) + n(x,y),$$

де $f(x,y)$ – спотворене зображення;

$m(x,y)$ – мультиплікативний стохастичний шум, що змінює яскравість;

$s(x,y)$ – ідеальне зображення;

Q – функціонал, що описує стохастичні геометричні, радіометричні спотворення зображення, а також раз фокусування;

$n(x,y)$ – адитивний стохастичний шум зображення.

Покращення зображення – зменшення стохастичного шуму за складовими:

$$m(x,y), Q, n(x,y).$$

Програмна реалізація

Напишемо програмний скрипт, який слугуватиме як інструмент для визначення характеристик об’єктів міської забудови.

Гістограма вхідного зображення

Проаналізуємо спочатку гістограму яскравості вхідного зображення.

<u>Лістинг коду:</u>

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Гістограма яскравості вхідного зображення
def input_image(img_name):
    # Зображення
    img = cv2.imread(img_name)
    imS = cv2.resize(img, (600, 500))
    cv2.imshow("img", imS)

    # Гістограма яскравості
    plt.hist(img.ravel(), 256, [0, 256])
    plt.show()

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

```
    return img

# Блок головних викликів
if __name__ == "__main__":
    image_name = 'Sentinel-2_L2A_True_Color'
    image = image_name + '.jpg'
    result_image = image_name + '_Result.jpg'

    # Гістограма яскравості вхідного зображення
    img = input_image(image)
```

Результат:

Гістограма має два піки. Більший скоріш за все відповідає за зелений колір на зображенні (природа), а другий, імовірно, за світло-коричневий (об'єкти міської забудови).

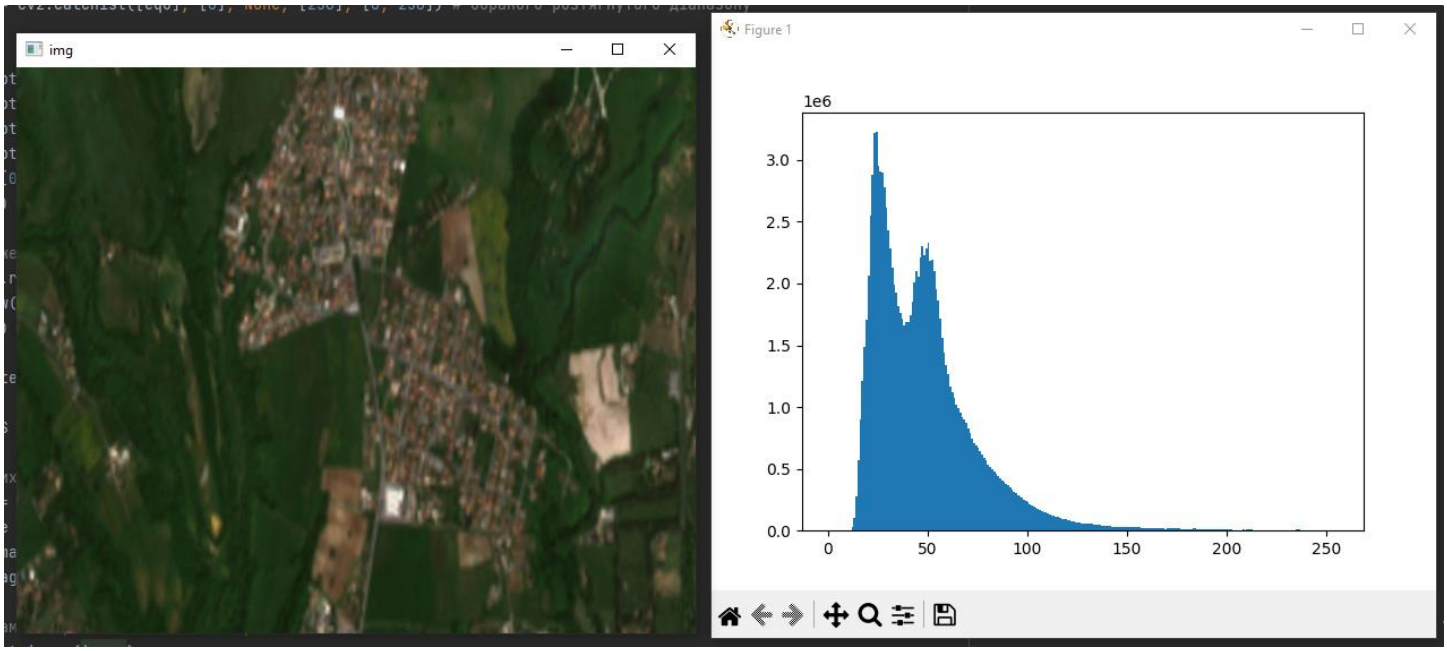

Рисунок 4 – Гістограма яскравості вхідного зображення

Накладання маски

Виділимо вручну на вхідному зображенні область, яка містить потрібну нам інформацію і порівняємо піки на гістограмі вхідного зображення з піками на виділеній області.

Результат зобразимо у вигляді графіка з 4-х субграфіків.

Лістинг коду:

```
# Накладання маски на вхідне зображення
def mask(img):
    # Створення маски
    mask = np.zeros(img.shape[:2], np.uint8)
    # Розміри маски (висота, ширина)
    mask[0:1200, 4000:5000] = 255
    # Зображення під маскою
    masked_img = cv2.bitwise_and(img, img, mask=mask)

    # Гістограми яскравості
    hist_full = cv2.calcHist([img], [0], None, [256], [0, 256]) # Вхідного зображення
    hist_mask = cv2.calcHist([img], [0], mask, [256], [0, 256]) # Зображення під маскою

    # Графік
    plt.subplot(221), plt.imshow(img, 'gray') # Вхідне зображення
    plt.subplot(222), plt.imshow(mask, 'gray') # Маска
    plt.subplot(223), plt.imshow(masked_img, 'gray') # Зображення під маскою
    plt.subplot(224), plt.plot(hist_full), plt.plot(hist_mask) # Гістограми: вхідного
зображення та зображення під маскою
    plt.xlim([0, 256])
    plt.show()

# Блок головних викликів
if __name__ == "__main__":
    image_name = 'Sentinel-2_L2A_True_Color'
    image = image_name + '.jpg'
    result_image = image_name + '_Result.jpg'

    # Гістограма яскравості вхідного зображення
    img = input_image(image)
    # Накладання маски на вхідне зображення
    mask(img)
```

Результат:

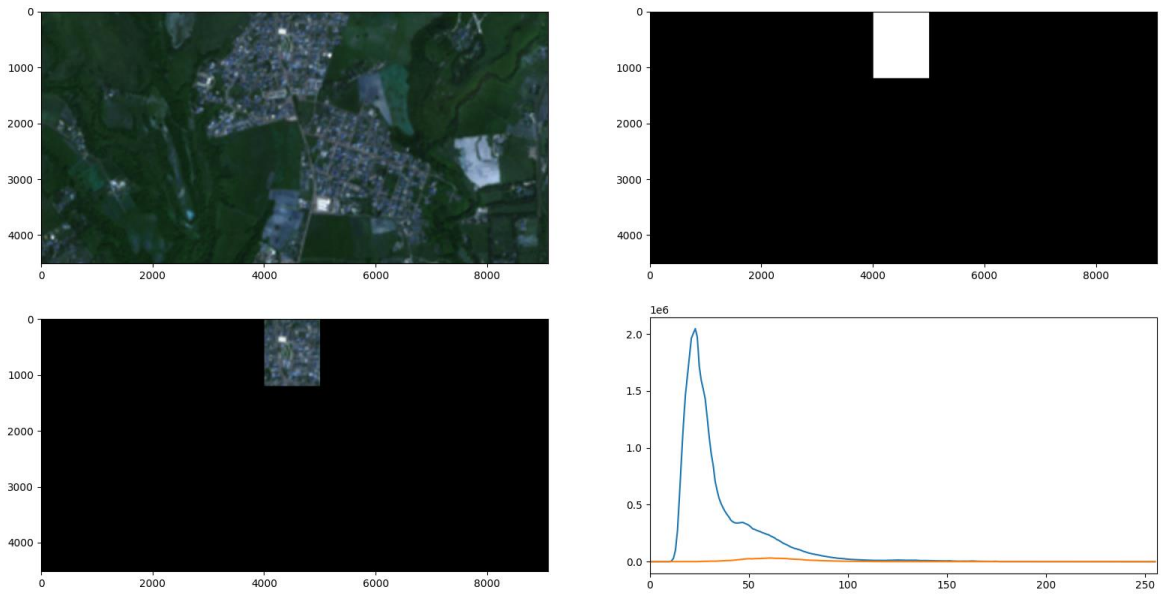

Рисунок 5 – Накладання маски на область з міською забудовою та гістограма маски

Бачимо, що перший пік взагалі відсутній на другій гістограмі. А отже наші припущення, щодо належності цього піку до зеленого кольору істинні.

Якщо збільшити гістограму в районі другого піку, то бачимо, що на нашій масці відсутній сам пік, але присутній спуск з піку. Тому логічно припустити, що другий пік належить білому кольору, а спуск з піку світло-коричневому (будівлям).

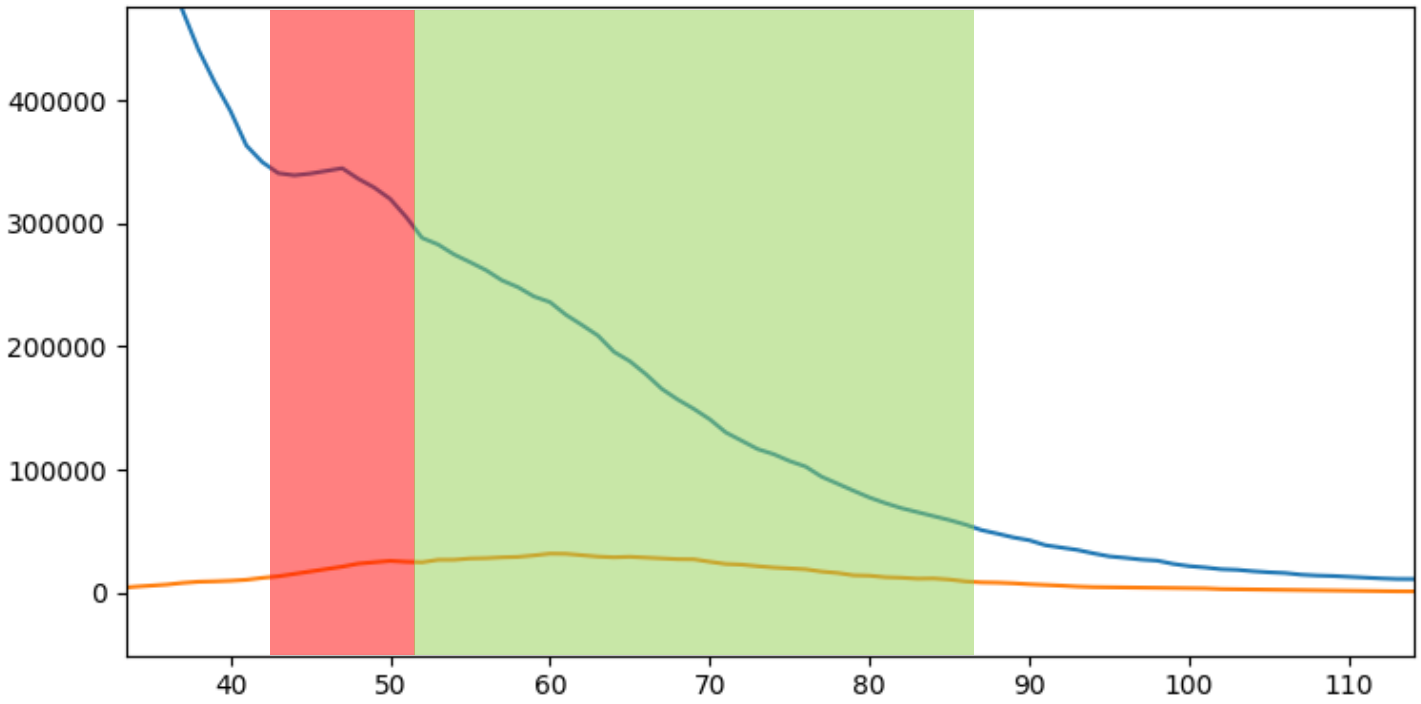

Рисунок 6 – Яскравість міської забудови

Відсікання діапазону яскравості та розтягнення гістограми

Якщо ми будемо відсікати яскравість в діапазоні 53-87, який знайшли на попередньому кроці, то ми захопимо багато шумів. Це чудово видно з гістограми.

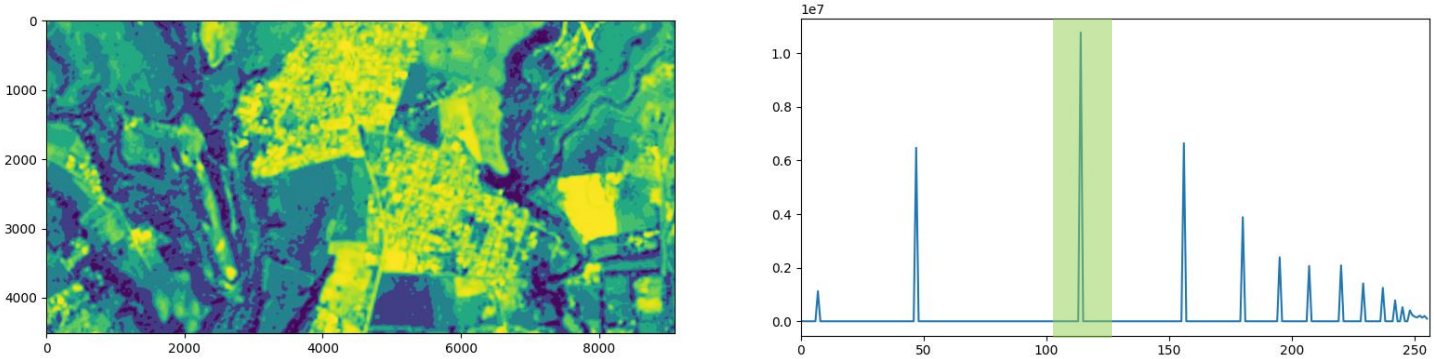

Рисунок 7 – Зашумлений результат відсікання

Для того, щоб прибрати шуми, необхідно відкинути зайві піки на гістограмі і залишити найінтенсивніший (жовтий).

Для цього експериментально підберемо вдалий діапазон: 70-72.

Результат:

Бачимо, що гістограма має один пік і максимально контрастно виділяє об’єкти міської забудови.

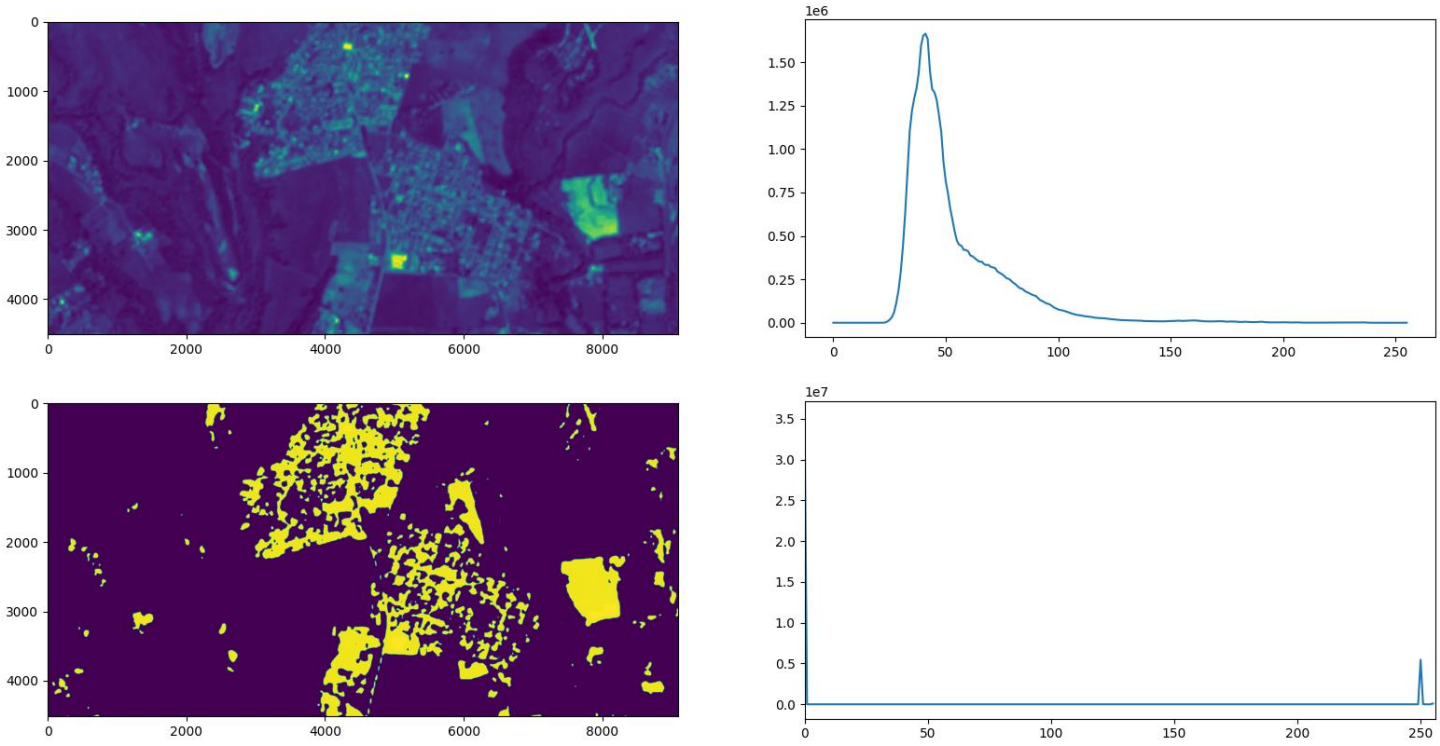

Рисунок 8 – Результат ідентифікації міської забудови

Лістинг коду:

```
# Розтягнення гістограми
def hist_normalise(image_name, result_image_name):
    # ЧБ вхідне зображення
    img = cv2.imread(image_name, 0)

    # Відокремлення діапазону яскравості
    normalized_img = cv2.normalize(img, None, 70, 72, cv2.NORM_MINMAX)
    # Розтягнення обраного діапазону на весь діапазон 0-255
    equ = cv2.equalizeHist(normalized_img)
    # Зклеювання зображень
    res = np.hstack((img, equ))

    # Гістограми яскравості
    hist_full = cv2.calcHist([img], [0], None, [256], [0, 256]) # Вхідного зображення
    hist_equ = cv2.calcHist([equ], [0], None, [256], [0, 256]) # Обраного розтягнутого
діапазону
```

```
    # Графік
    plt.subplot(221), plt.imshow(img) # Вхідне зображення
    plt.subplot(222), plt.plot(hist_full) # Гістограма
    plt.subplot(223), plt.imshow(equ) # Обраний розтягнутий діапазон
    plt.subplot(224), plt.plot(hist_equ) # Гістограма
    plt.xlim([0, 256])
    plt.show()

    # Відображення результатів (зклеєний кадр)
    imS = cv2.resize(res, (800, 300))
    plt.imshow(imS)
    plt.show()

    cv2.imwrite(result_image_name, res)

    return imS

# Блок головних викликів
if __name__ == "__main__":
    image_name = 'Sentinel-2_L2A_True_Color'
    image = image_name + '.jpg'
    result_image = image_name + '_Result.jpg'

    # Гістограма яскравості вхідного зображення
    img = input_image(image)
    # Накладання маски на вхідне зображення
    mask(img)
    # Розтягнення гістограми
    result = hist_normalise(image, result_image)
```

Поріняння з вхідним зображенням

Для оцінки ступеня покращення зображення для ідентифікації об’єктів міської забудови, порівняємо чи можна виділити контури цікавої для нас області у вхідному та результуючому зображеннях.

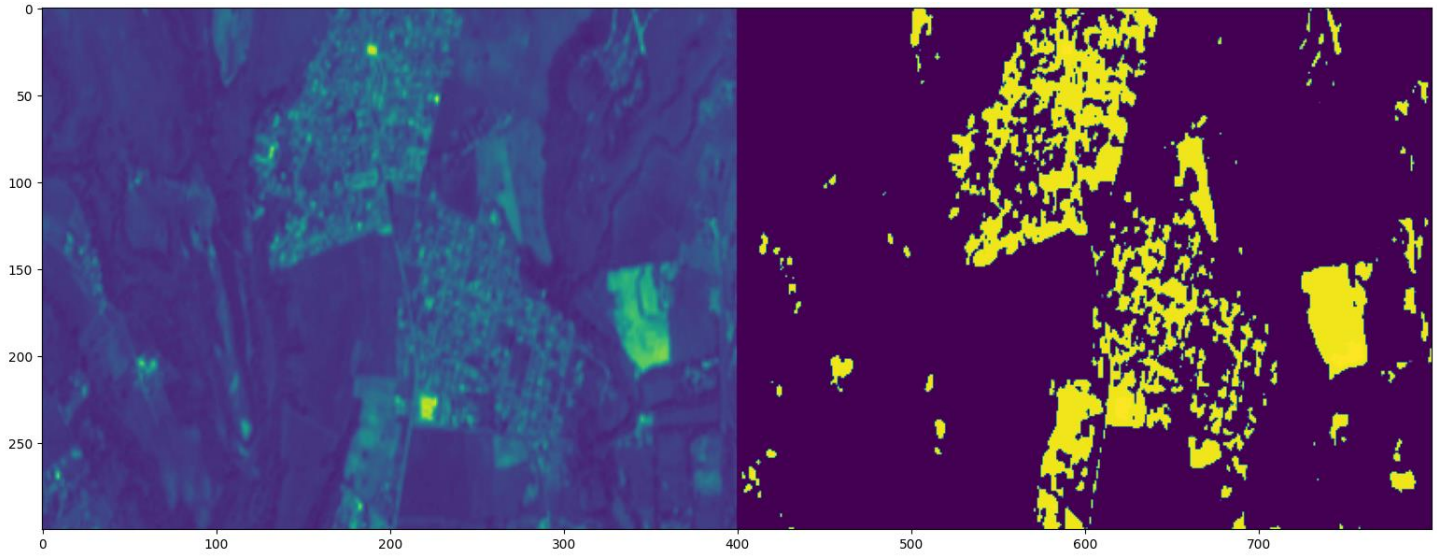

Рисунок 9 – Порівняння чіткості контурів міської забудови

Бачимо, що на вхідному зображенні навряд чи вдалось би виокремити контури об’єктів міської забудови, а на результуючому контури об’єктів чітко видно неозброєним оком. З таким зображенням можна в подальшому працювати.

Отже, **виділення «ознак міської забудови» виконано успішно**. Можна переходити до обробки відео.

2.3. Покращення якості відео

Тепер, коли визначені параметри для відсікання яскравості для відео, можна перейти до покадрового покращення відео.

Алгоритм програми

Для цього напишемо ще одну програму, яка буде відсікати за визначеною ознакою яскравості шуми з кожного кадру відео.

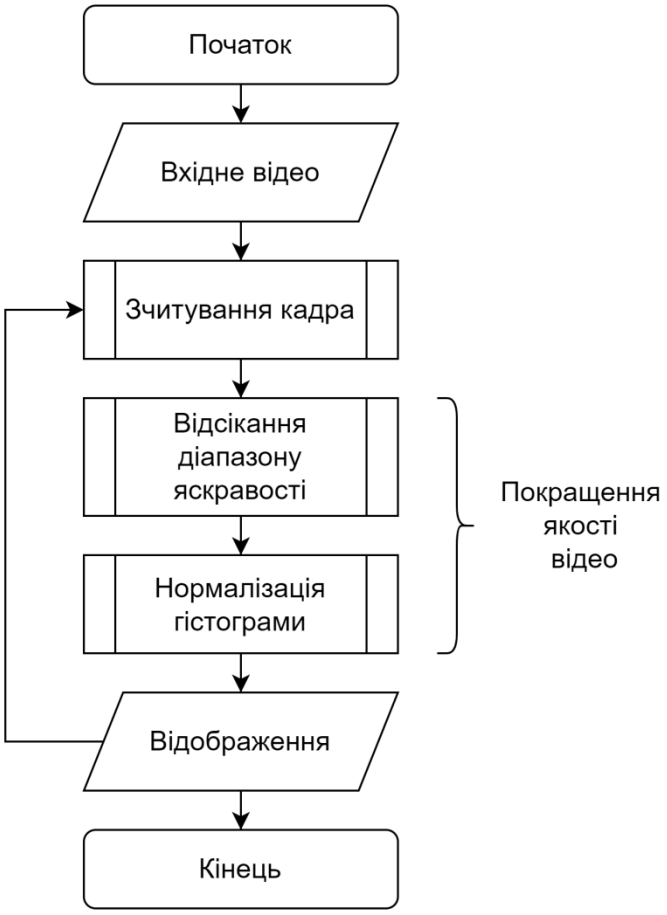

Рисунок 10 – Блок схема програми

Програмна реалізація

Додамо затримку для більш плавного відображення кадрів та збережемо отриманий відео-результат.

Лістинг коду:

```
import cv2
import time

# Відео з файлу
cap = cv2.VideoCapture('timelapse.mp4')

# Параметри вихідного відео (назва файлу, чотирьохсимвольний кодек, кадрова швидкість,
розмір кадру)
out = cv2.VideoWriter('result_video.mp4', cv2.VideoWriter_fourcc(*'avc1'), 30,
(int(cap.get(3)), int(cap.get(4))))


while True:
    ret, frame = cap.read()
    if not ret:
        break  # Вийти з циклу, якщо відео закінчилося

    # Затримка для плавності відео
    time.sleep(0.03)

    # Кадр в ЧБ
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Перетворення яскравості (відсікання діапазону)
    normalized_frame = cv2.normalize(gray, None, 70, 72, cv2.NORM_MINMAX)
    # Перетворення яскравості (розтягування діапазону на весь спектр)
    equ_frame = cv2.equalizeHist(normalized_frame)

    # Відображення
    cv2.imshow('Video', frame)
    cv2.imshow('Frame', equ_frame)

    # Запис кожного кадру у вихідне відео
    out.write(equ_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Результат:

Бачимо, що алгоритм спрацьовує по-різному, в залежності від пори року на знімку місцевості та від змін у оточенні.

Так, наприклад, зоране поле на другому знімку по діапазону яскравості ідентифікується як будівля.

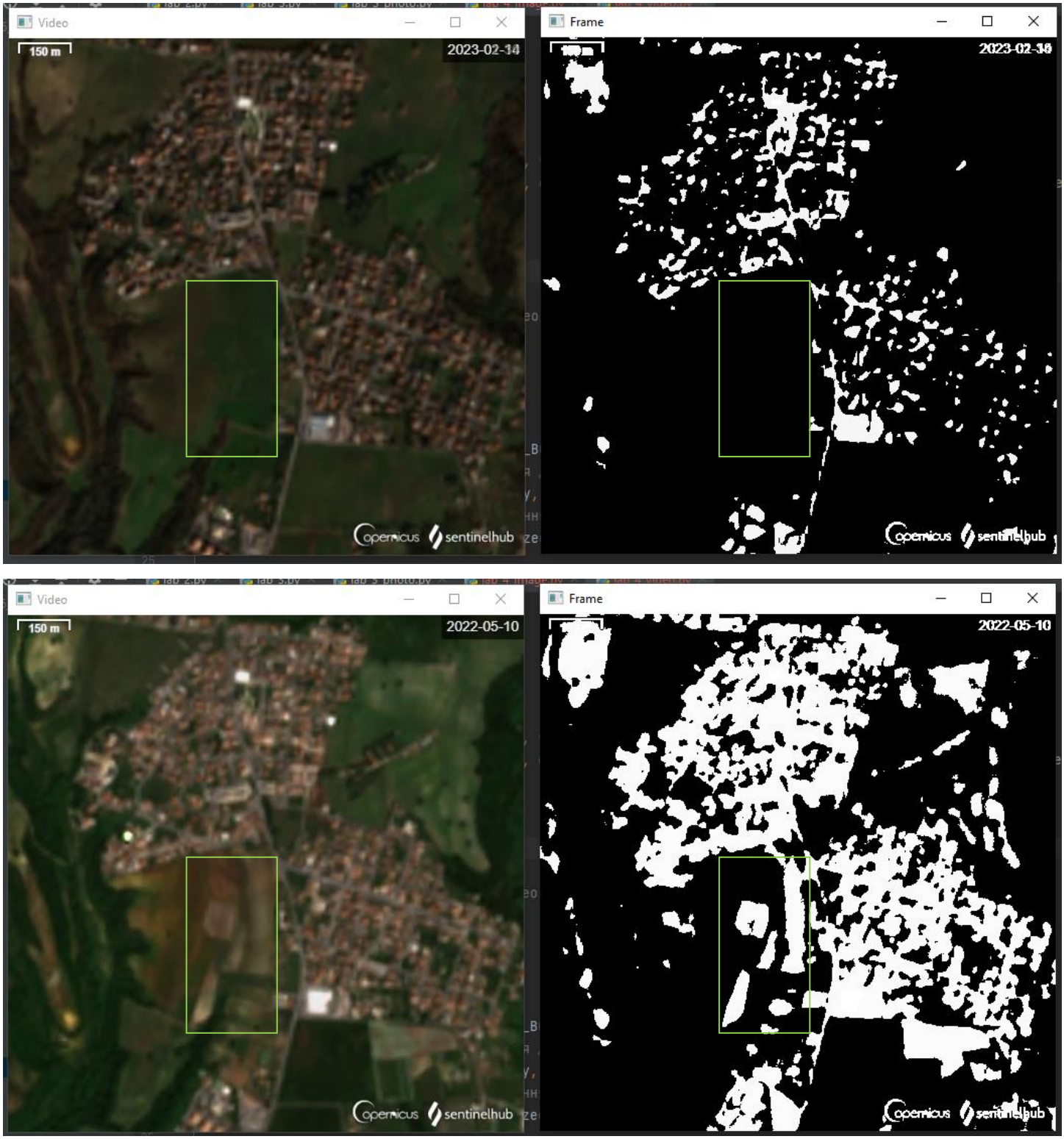

Рисунок 11 – Виділення об’єктів міської забудови на відео

Хоч, де-не-де алгоритм захоплює зайві об’єкти, загалом, **отримано покращення** ідентифікації міської забудови на відео.

Повне відео результату можна побачити в папці проекту на github.

2.3. Аналіз отриманих результатів

Знайдено діапазон яскравості, який відповідає об’єктам міської забудови – підтверджено рисунком.

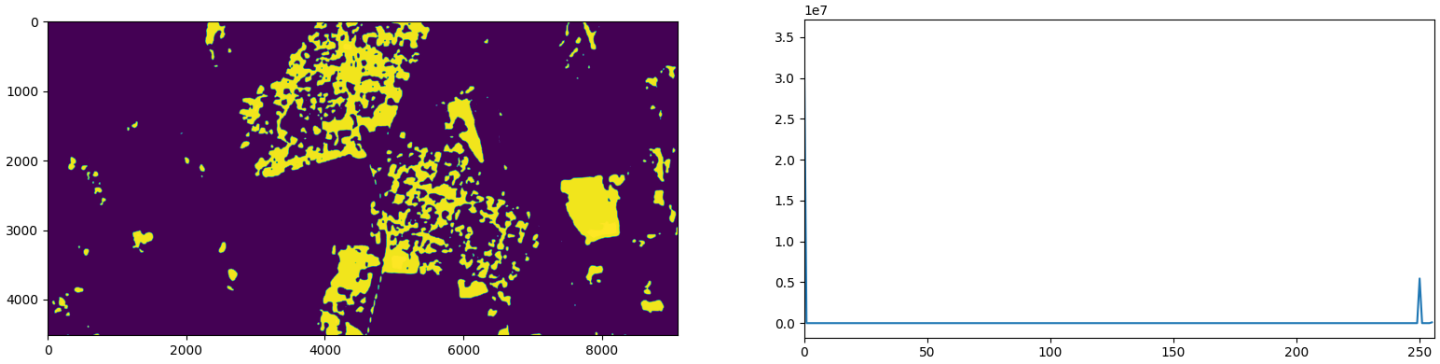

Рисунок 12 – Підтвердження знаходження потрібного діапазону яскравості

Покращено якість ідентифікації об’єктів міської забудови на відео – підтверджено рисунком і відео.

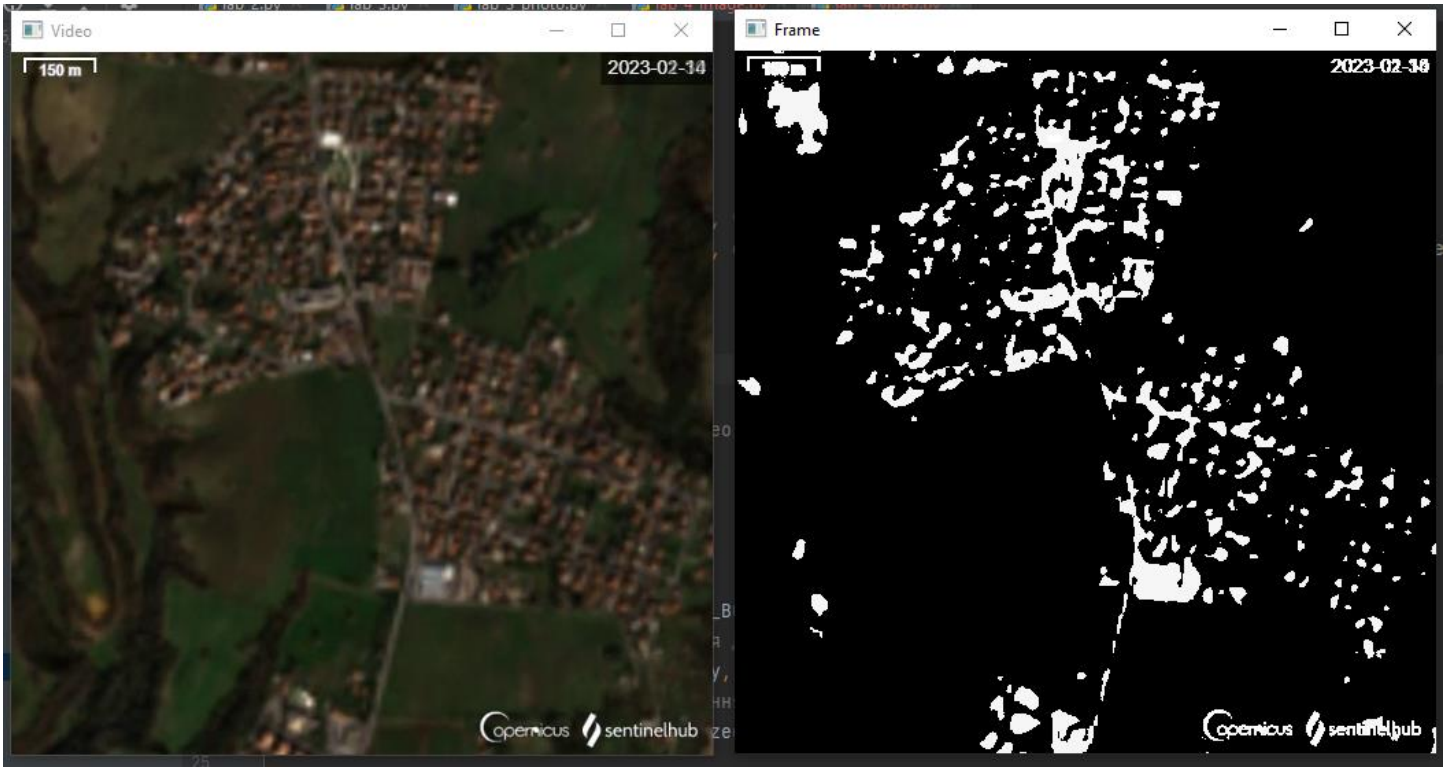

Рисунок 13 – Підтвердження покращення якості відео

Помічено, що **сезонність фото та відео впливає на діапазон яскравості шуканих об’єктів**.

Рішення: шукати діапазон яскравості для кожної пори року окремо.

Висновок:

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

Отримано досвід підвищення якості фото та відео шляхом змінення гістограми яскравості зображення.

Реалізовано програмний скрипт №1, за допомогою якого можна аналізувати характерні діапазони яскравості для обраних груп об’єктів.

Реалізовано програмний скрипт №2, який виконує покращення якості відео на основі заданих параметрів зміни гістограми яскравості.

Отримані результати покращення якості відео далекі від ідеальних. Очевидно, **для збільшення точності ідентифікації шуканих об’єктів** на зображенні **слід застосовувати інші методи сегментації об’єктів**.