Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем управління
(повна назва кафедри)

Бакалаврська дипломна робота
на тему:

«Тема «Автоматизована система дистанційного зондування земної поверхні з можливістю класифікації об'єктів»»

Виконав: студент 4-го курсу, групи
2АКІТ-20б
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

_____ Дмитро ПАРХОМЕНКО
(ім'я та прізвище)

Керівник: к.т.н., доцент каф. КСУ
_____Олена НИКИТЕНКО
(ім'я та прізвище)
«10» 06 2024 р.

Рецензент: к.т.н., доц. каф. АІІТ
_____Юрій ІВАНОВ
(ім'я та прізвище)
«11» 06 2024 р.

Допущено до захисту
Завідувач кафедри КСУ
д.т.н., проф.
_____В'ячеслав КОВТУН
(ім'я та прізвище)
«14» 06 2024 р.

Вінниця ВНТУ - 2024 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Факультет Інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти І (бакалаврський)
Галузь знань – 15 Автоматизація та приладобудування
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ
д.т.н., проф.
_____ Вячеслав КОВТУН
«14» 05 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Пархоменко Дмитру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизована система дистанційного зондування земної поверхні з можливістю класифікації об'єктів»
керівник роботи Никитенко Олена Дмитрівна, к.т.н., доцент кафедри КСУ
затверджені наказом вищого навчального закладу від "13" 05 2024 року № 90

1. Термін подання студентом роботи 10 06 2024 року

2. Вихідні дані до роботи: 1. J. R. Jensen, "Introductory Digital Image Processing: A Remote Sensing Perspective," 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2015; 2. P. M. Mather and M. Koch, "Computer Processing of Remotely-Sensed Images: An Introduction," 4th ed. Chichester, UK: Wiley-Blackwell, 2011; 3. R. C. Gonzalez and R. E. Woods, "Digital Image Processing," 4th ed. New York, NY, USA: Pearson, 2018.

3. Зміст текстової частини: техніко-економічне обґрунтування доцільності розробки, багатокритеріальний аналіз способів та сервісів для побудови веб-системи, вимоги до умови, розробка архітектури системи, розробка програмного продукту, аналіз вимог до тестування програмного забезпечення.

4. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
1 УМЛ – діаграма варіантів використання, основні екранні форми програмного забезпечення.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Виконання прийняв
1-3	д.т.н., проф. Коатуз В.В.		

6. Дата видачі завдання 29.05.2024 рік

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів завдання	Примітка
1.	Постановка задачі дослідження	04.05.2024 р.	
2.	Визначення технічних характеристик системи	15.05.2024 р. - 19.05.2024 р.	
3.	Розробка програмного забезпечення системи	20.05.2024 р. - 25.05.2024 р.	
4.	Тестування програмного забезпечення системи	26.05.2024 р. - 28.05.2024 р.	
5.	Графічні матеріали	29.05.2024 р. - 31.05.2024 р.	
6.	Оформлення пояснювальної записки та презентації	01.06.2024 р. - 08.06.2024 р.	
7.	Попередній захист КДР	10.06.2024 р.	

Студент _____ Дмитро ПАРХОМЕНКО
(підпис)

Керівник роботи _____ Оксана ПРИХОДЬКО
(підпис)

АНОТАЦІЯ

У цій роботі було розроблено програмну систему для виявлення об'єктів за даними дистанційного зондування Землі (ДДЗ) з відкритих інформаційних джерел, використовуючи технології Computer Vision. Метою даного дослідження є підвищення ефективності та надійності ідентифікації даного об’єкта за даними ДЗЗ на основі відкритих джерел інформації. Система здатна обробляти супутникові знімки та виявляти на них об'єкти. Крім того, програмний функціонал дозволяє порівнювати зображення та візуально відстежувати їх зміни у часових інтервалах. Програмну частину написано на мові програмування Python, що забезпечує інструменти для обробки зображень через бібліотеку OpenCV. Графічний інтерфейс розроблено з використанням бібліотеки CustomTkinter, яка розширює можливості бібліотеки Tkinter. Демонстрація програмного забезпечення підтверджує ефективність реалізації та досягнення мети дипломної роботи.

ABSTRACT

In this work, a software system was developed for detecting objects based on the data of remote sensing of the Earth from open information sources, using Computer Vision technologies. The system is able to process satellite images and detect objects on them. In addition, the software functionality allows you to compare images and visually monitor their changes in time intervals. The software part is written in the Python programming language, which provides tools for image processing through the OpenCV library. The GUI is developed using the CustomTkinter library, which extends the capabilities of the Tkinter library. The software demonstration confirms the effectiveness of the implementation and achievement of the thesis goal.

‘

ЗМІСТ

‘

‘

ВСТУП

Актуальність дослідження. У сучасному світі дистанційне зондування Землі з використанням відкритих джерел інформації є одним із ключових інструментів отримання цінних даних про Землю.

Однак обробка великої кількості даних, отриманих від дистанційного зондування, вимагає багато зусиль і ефективних інструментів, таких як ті, які здатні ідентифікувати певні об’єкти дослідження на зображеннях.

Об’єкт дослідження – процес дистанційного зондування земної поверхні з можливістю класифікації об’єктів.

Предмет дослідження – методи й засоби автоматизованої класифікації конкретних об’єктів на зображеннях, отриманих в результаті дистанційного зондування земної поверхні.

Метою даного дослідження є підвищення ефективності та надійності ідентифікації даного об’єкта за даними ДЗЗ на основі відкритих джерел інформації.

Для досягнення поставленої мети основними завданнями дослідження є:

1. Проаналізувати відомі технічні рішення виявлення даного об’єкта за даними ДЗЗ з відкритих джерел інформації.

2. Проаналізувати відомі методи розпізнавання об’єктів у задачах комп’ютерного зору.

3. Розробити спеціальні програмні системи виявлення об’єктів на основі даних ДЗЗ із загальнодоступних джерел інформації.

4. Продемонструвати практичні можливості розробленої програмної системи.

Об’єктом даної роботи є процес розпізнавання цілей на супутникових зображеннях у задачах комп’ютерного зору.

Інноваційність даного дослідження полягає у розробці та впровадженні нових методів і засобів автоматизованої класифікації об'єктів на зображеннях, отриманих в

результаті дистанційного зондування земної поверхні (ДЗЗ). Відмінною рисою цього дослідження є:

1. Використання відкритих джерел інформації: Впровадження методик, які використовують загальнодоступні дані, що забезпечує більшу доступність та знижує витрати на отримання даних.

2. Адаптація сучасних методів комп'ютерного зору: Інтеграція передових алгоритмів машинного навчання та глибокого навчання для підвищення точності ідентифікації та класифікації об'єктів.

3. Розробка спеціалізованих програмних систем: Створення нових програмних рішень, що забезпечують автоматизацію процесу ідентифікації об'єктів, зменшуючи необхідність ручної обробки даних.

4. Інтеграція мультиспектральних та гіперспектральних даних: Використання даних з різних спектральних діапазонів для підвищення якості ідентифікації об'єктів.

Практичне значення цього дослідження полягає у значному підвищенні ефективності та надійності ідентифікації об'єктів на основі даних ДЗЗ, що має широке застосування у різних галузях:

1. Екологічний моніторинг: Поліпшення процесів моніторингу екологічного стану територій, включаючи виявлення змін у ландшафті, забруднення водних ресурсів та зсувів ґрунту.

2. Сільське господарство: Автоматизована класифікація сільськогосподарських культур та оцінка їхнього стану для оптимізації процесів вирощування та збору врожаю.

3. Урбаністика та містобудування: Аналіз міської інфраструктури, ідентифікація нових будівель та моніторинг зміни міського середовища для ефективного планування розвитку міст.

4. Лісове господарство: Виявлення та моніторинг лісових пожеж, хвороб дерев, а також незаконних вирубок.

5. Безпека та оборона: Використання у задачах національної безпеки для ідентифікації об'єктів інтересу на супутникових зображеннях.

Дослідження має потенціал суттєво зменшити затрати часу та ресурсів на обробку даних, забезпечуючи високу точність і надійність результатів, що є критично важливим

для прийняття рішень в різних галузях.

‘

1 АНАЛІЗ ПІДХОДІВ ДО ВИЯВЛЕННЯ КОНКРЕТНИХ ОБ'ЄКТІВ ЗА ДАНИМИ ВІДКРИТИХ ДЖЕРЕЛ ДИСТАНЦІЙНОГО ЗОНДУВАННЯ ЗЕМЛІ

Дистанційне зондування Землі (ДДЗ) має великий потенціал для розуміння та вивчення нашої планети, але для досягнення цієї мети необхідні ефективні рішення для обробки та аналізу отриманих даних.

Тому в цьому розділі будуть розглянуті різні техніко-методичні підходи до виявлення того чи іншого об'єкта за допомогою дистанційного зондування Землі, а також їх ефективність і можливість застосування в окремих галузях. Крім того, також будуть переглянуті відкриті джерела даних для ДЗЗ та його супутникових систем.

1.1 Аналіз відомих технічних рішень виявлення даного об'єкта за даними ДЗЗ з відкритих джерел інформації.

1.1.1 Супутникові системи.

Супутникові системи — це технологічні рішення для збору даних із систем дистанційного зондування з великими можливостями виявлення певного об’єкта.

Супутникові системи включають штучні супутники, які обертаються навколо Землі та надають нам зображення та дані про Землю з великих висот. Ці супутники оснащені датчиками, здатними збирати інформацію про поверхню Землі, ось кілька прикладів:

• Оптичний датчик. Оптичні датчики використовуються для вловлювання світла, відбитого від поверхні Землі. Це можуть бути супутникові камери, які записують видиме світло та інфрачервону інформацію. Вони пропонують можливість отримання зображень земної поверхні з високою роздільною здатністю, що полегшує виявлення та дослідження об’єктів з високою деталізацією, таких як будівлі, дороги, ліси тощо [1].

• Радарні системи. Радарні системи використовують радіохвилі для виявлення об'єктів і отримання інформації про їх розташування та властивості. Супутникові радарні системи, такі як радар, вимірюють радіацію, відбиту радіосигналами, і надають інформацію про форму, розмір і топографію об’єктів. Ці системи ефективно працюють

навіть за несприятливих погодних умов і вночі, що робить їх ефективними при виявленні об’єктів навіть у складних умовах [2].

• Тепловізор. Теплові камери, також відомі як інфрачервоні камери, використовуються для виявлення інфрачервоного випромінювання, що випромінюється об’єктами. Вони здатні виявляти теплове випромінювання від об’єктів навколишнього середовища незалежно від освітлення та погодних умов.

Супутникові системи забезпечують зображення з високою просторовою роздільною здатністю та можуть охоплювати великі території. За допомогою супутникових систем можна виявляти об'єкти, вивчати зміни в ландшафті, визначати особливості поверхні і виконувати ряд інших завдань.

Сьогодні існує багато публічних ресурсів, які надають безкоштовний доступ до супутникових знімків. Аналізуючи певну кількість джерел професійної інформації у сфері ДЗЗ, можна виділити такі зручні сервіси:

• Макса. MAXAR має такі супутники: WorldView-1, WorldView-2, WorldView-3 і WorldView-4. Їх супутники використовують різні спектральні канали, включаючи видиме світло, Можна отримати інфрачервону та панхроматичну спектроскопію, різноманітну інформацію про наземні об'єкти. Цей ресурс включає збір даних, обробку зображень, цифрове моделювання рельєфу, аналіз змін, виявлення об’єктів, класифікацію та інші аналітичні послуги. Приклад роботи сервісу наведено на рисунку (1.1).

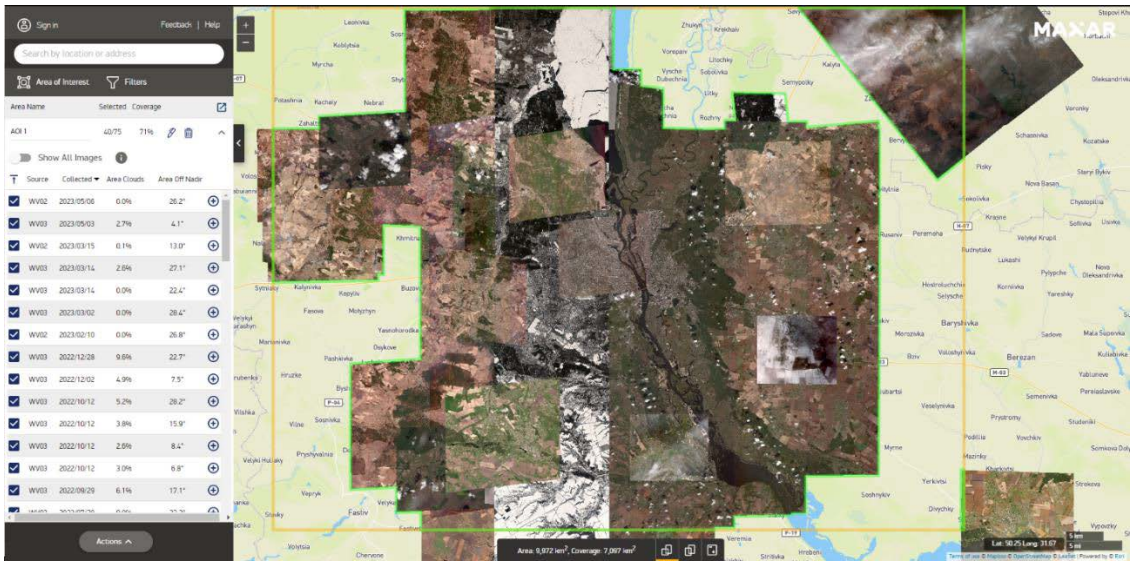


Рисунок 1.1 – Скріншот прикладу сервісу MAXAR для вибору конкретного напряму дослідження.

• Сторожовий центр. Платформа надає доступ до даних із різних супутників, включаючи зображення високої роздільної здатності з Sentinel-2, радіолокаційні дані з Sentinel-1, характеристики атмосфери з Sentinel-3 і дані про склад атмосфери з Sentinel-5P [3]. Приклад роботи Sentinel Hub показано на рисунку (1.2).

Рисунок 1.2 – Скріншот прикладу сервісу Sentinel Hub.

Користувачі можуть використовувати різні фільтри, алгоритми та моделі для отримання інформації із зображень і даних дистанційного зондування. Sentinel Hub надає можливість розширювати функціональність і створювати власні аналітичні рішення за допомогою сценаріїв і власного програмування.

1.1.2 Географічна інформаційна система (ГІС).

ГІС – це система, яка забезпечує збір, аналіз, зберігання та візуалізацію просторових даних і пов’язаних з ними об’єктів, які потребують відстеження. ГІС може включати бази даних (БД) DZZ, просторові бази даних, растрові та векторні графічні редактори та різноманітні інструменти для аналізу просторових даних. Такі системи використовуються в багатьох галузях, таких як геологія, картографія, метеорологія, де потрібні розширені можливості обробки та аналізу просторових даних.

Цей інструмент для аналізу та обробки географічних даних містить ГІС-методи та функції, які допомагають виявляти певні об’єкти [4]:

1. Класифікація зображень. ГІС може реалізувати машинне навчання та алгоритми обробки зображень для автоматичної ідентифікації об’єктів на зображеннях. Наприклад, за допомогою суперпіксельних методів сегментації та класифікації можна виділити лісові масиви, водойми, міські будівлі тощо.

2. Аналіз текстури та рисунка. ГІС може виявляти об’єкти за допомогою аналізу текстури та візерунка. Наприклад, поля, ліси, ставки та інші об’єкти на зображеннях можна автоматично ідентифікувати шляхом визначення типових текстур, властивих різним об’єктам.

3. Просторовий аналіз. ГІС надає функціональність для виконання просторового аналізу, наприклад буферизації, перетину та злиття регіонів. Ці інструменти дозволяють ідентифікувати об’єкти, які відповідають певним географічним критеріям. Наприклад, ви можете виявити об’єкти, які перетинають задану область інтересу або знаходяться на певній відстані від інших об’єктів.

4. Моделювання та прогнозування. ГІС може служити засобом моделювання та прогнозування розподілу об'єктів на основі наявних даних. Наприклад, аналізуючи історичні дані про зміни землекористування, можна передбачити майбутні зміни та визначити потенційні території для розширення об’єкта.

5. Візуалізуйте та співпрацюйте. ГІС дозволяє відображати географічні дані та результати аналізу у вигляді карт, діаграм та інших візуальних елементів. Це сприяє легкій інтерпретації інформації та спільній роботі з іншими користувачами.

Наведено приклади таких ресурсів, які є найбільш зручними та мають достатній список для задоволення різноманітних вимог. Деякі з них безкоштовні, але є також ГІС, які коштують додатково, щоб розширити дослідницькі можливості:

1. Геоінформаційна система:

^ ArcGIS, розроблена Esri, є однією з найпопулярніших у світі комерційних ГІС.

^ Надає розширені можливості для обробки, аналізу та візуалізації географічних даних.

^ ArcGIS пропонує різноманітні модулі та розширення для різних галузей, зокрема картографії, міського планування, екології, геології тощо.

^ Підтримує створення тематичної карти, аналіз просторових зв’язків,

моделювання та прогнозування[5].

2. Геоінформаційна система:

^ QGIS — це відкрита та безкоштовна ГІС, доступна для різних операційних систем (Windows, macOS, Linux).

^ Він має широкий спектр можливостей для обробки та аналізу географічних даних, включаючи векторний і растровий аналіз, геообробку, створення тематичних карт і зображень [5].

^ QGIS підтримує використання плагінів для розширення своєї функціональності та інтеграції з іншими ГІС і форматами даних.

/ Він має активну спільноту користувачів, яка постійно розробляє нові розширення та вдосконалення.

3. Google Earth:

/ Google Earth – це веб-програма ГІС, яка надає доступ до тривимірних зображень Землі, супутникових зображень і аерофотознімків.

^ Це дозволяє користувачам взаємодіяти з географічними даними, використовувати інтерактивні карти, вимірювати відстані та висоти, а також додавати маркери та теги.

/ Google Планета Земля також надає можливість реконструювати зміни в географічних даних з часом за допомогою історичних знімків.

^ Він використовується в різноманітних сферах, включаючи наукові дослідження, освіту, туризм і містобудування.

4. Геоінформаційна система пасовищ:

S GRASS GIS (Geographic Resources Analysis Support System) — це відкрита безкоштовна ГІС, призначена для аналізу та обробки географічних даних.

^ Він забезпечує універсальне середовище для географічного аналізу, геостатистики, моделювання поверхні, обробки зображень, статистичного аналізу та інших операцій із географічною інформацією.

^ GRASS GIS має потужний командний рядок і графічний інтерфейс, а також розширені функції для автоматизованої обробки географічної інформації.

5. Інформація про карту:

‘

^ MapInfo — комерційна ГІС, розроблена Пітні Боузом.

^ Він надає різноманітні інструменти для обробки та аналізу географічних даних, включаючи векторні та растрові дані.

^ MapInfo має функції візуалізації даних, картографування, аналізу просторових зв’язків, маршрутизації та аналізу мережі.

1.1.3 Мультиспектральний аналіз.

Мультиспектральний аналіз — це метод обробки та аналізу географічних даних, який використовує різні діапазони спектрального випромінювання для отримання додаткової інформації про земну поверхню. Метод передбачає використання даних різних спектральних діапазонів, включаючи видиме світло, інфрачервоне випромінювання та інші спектральні діапазони [6].

Мультиспектральне зображення виконується за допомогою спеціальної камери, здатної розділяти світло на різні спектральні компоненти. Цей процес створює монохромну фотографію кожного кадру, кількість яких залежить від кількості спектральних каналів у камері. Інформація, отримана з цих фотографій, зазвичай аналізується в географічних інформаційних програмах за допомогою різних індексів (наприклад, NDVI, NDRE, SAVI, LAI). Основним застосуванням багатоспектральних зображень є визначення стану рослинності.

Одним із підходів до мультиспектрального аналізу є використання показників відбиття, які розраховуються на основі різних спектральних діапазонів. Наприклад, індекс NDVI (Normalized Difference Vegetation Index) використовується для визначення розподілу рослинності на поверхні Землі. На рисунку (1.3) показано візуальне представлення індексу NDVI.

‘

Рисунок 1.3 – Приклад NDVI з відкритих джерел даних DZZ – Sentinel Hub.

Значення NDVI варіюються від -1 до 1.

NDVI range	HTLM color code	Color
NDVI < -0.2	#000000	
-.2 < NDVI ≤ 0	#a50026	
0 < NDVI ≤ .1	#d73027	
.1 < NDVI ≤ .2	#f46d43	
.2 < NDVI ≤ .3	#fdae61	
.3 < NDVI ≤ .4	#fee08b	
.4 < NDVI ≤ .5	#ffffbf	
.5 < NDVI ≤ .6	#d9ef8b	
.6 < NDVI ≤ .7	#a6d96a	
.7 < NDVI ≤ .8	#66bd63	
.8 < NDVI ≤ .9	#1a9850	
.9 < NDVI ≤ 1.0	#006837	

Рисунок 1.4 – Колірна гамма за діапазоном NDVI.

Значення NDVI в негативному діапазоні (близько -1) вказують на наявність води. Значення, близькі до нуля (від -0,1 до 0,1), зазвичай вказують на безплідні ділянки, такі як камінь, пісок або сніг. Низькі позитивні значення вказують на наявність чагарників і луків (приблизно від 0,2 до 0,4), тоді як високі значення вказують на наявність помірних і тропічних лісів (значення, близькі до 1) [7]. Цей показник є хорошим показником наявності живої зеленої рослинності. Колірну гамму та діапазон зображено на рисунку (1.4)

1.2 Аналіз відомих методів розпізнавання об'єктів у задачах комп'ютерного зору.

Комп’ютерне бачення (CV) або комп’ютерне бачення — це технологія, яка дозволяє розробляти алгоритми для отримання, аналізу та розуміння інформації з візуальних даних, таких як зображення та відео.

Головна мета розв’язування задач із CV — вміти витягувати інформацію із зображень, відео та розуміти її так само, як люди (наприклад, ідентифікувати та класифікувати об’єкти).

1.2.1 Обробка зображень.

Перед застосуванням алгоритмів розпізнавання об'єктів зображення необхідно спочатку обробити. Це важливий крок до досягнення ваших цілей і покращення результатів. Деякі типові етапи обробки зображень описані нижче.

1. Підвищення якості.

Процес включає набір методів і алгоритмів, призначених для виправлення дефектів або помилок у зображенні для покращення його чіткості, якості або візуального вигляду [8].

• Фільтрація шуму. Такі фільтри, як медіанний фільтр, фільтр Гауса або фільтр згладжування, використовуються для зменшення шуму на зображеннях. Це допомагає покращити чіткість і деталізацію зображення.

• Корекція освітленості. Застосування методів корекції освітлення регулює рівні яскравості та контрастності зображення, що допомагає покращити його вигляд і деталізацію.

• Видалити розмитість. Використання алгоритмів зменшення розмиття може покращити різкість і чіткість ваших зображень, особливо під час фотографування рухомих об’єктів або в умовах обмеженого освітлення.

• Розширення динамічного діапазону. Використання технології розширення динамічного діапазону може збільшити контрастність і деталізацію зображень,

особливо в сценах із великою різницею між світлими й темними ділянками.

• Виправте спотворення. Алгоритми використовуються для виправлення спотворень, які можуть виникнути через об’єктивні або інші фактори, такі як вигин або розмиття [9].

2. Трансформація колірного простору.

Кольорові простори визначають, як кольори представлені на зображеннях. Це математична модель, яка дозволяє описувати колір за допомогою числових значень.

Найпоширеніші колірні простори включають:

• RGB. RGB — це колірний простір, заснований на комбінації трьох основних каналів — червоного, зеленого та синього. Кожен канал представляє інтенсивність відповідного кольору в кожному пікселі. Завдяки широкому застосуванню RGB використовується у візуальних дисплеях і цифровій фотографії.

• Вірус простого герпесу. HSV — це колірний простір, що складається з трьох основних компонентів: тон, насиченість і значення. Відтінок представляє тональні характеристики кольору, насиченість визначає насиченість кольору, а значення представляє яскравість. Зміни відтінку можна легше виявити та обробити за допомогою HSV.

• Лабораторія. LAB — це колірний простір, що складається з трьох частин: світлота, колірний канал «A» і колірний канал «B». Яскравість відображає ступінь освітлення, а колірні канали «A» і «B» визначають спектральні характеристики. Використання LAB широко використовується в області аналізу та розробки алгоритмів комп'ютерного зору.

• ЮВ. YUV — це колірний простір, що складається з трьох компонентів: яскравості (Luma), кольоровості синього (U) і насиченості червоного (V). YUV широко використовується в телевізійних технологіях і стисненні відео для розділення яскравості та кольору, що дозволяє ефективно зберігати та передавати відеоінформацію.

3. Сегментація.

Під час цього процесу зображення ділиться на окремі фрагменти, щоб спростити його представлення та перетворити його на більш значущі та аналізовані компоненти.

Основною метою сегментації є розділення об’єктів на зображенні та виділення їх

меж.

• Порогова сегментація.

Цей метод використовує порогові значення для поділу зображення на дві або більше областей. Пікселі зі значеннями, що перевищують заданий поріг, призначаються одній області, тоді як пікселі з меншими значеннями призначаються іншій області. Виберіть порогове значення як межу візерунка для гістограми зображення. Приклад наведено на рисунку (1.15).

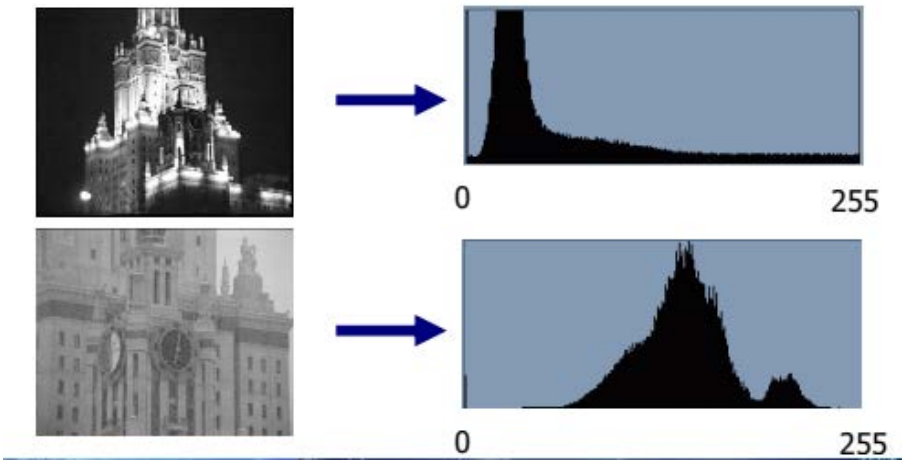


Рисунок 1.5 - Сегментація зображення гістограмами, що містять пересічні максимуми.

• За рахунок збільшення площі.

Цей метод заснований на ідеї об’єднання пікселів в однорідні області на основі певних критеріїв подібності [10].

Основні кроки для додавання методу сегментації регіону такі:

1. Виберіть початковий піксель або область як початкові

точка росту.

2. Встановіть критерії подібності для регіонів, беручи до уваги такі параметри, як інтенсивність пікселів, колір, текстура чи інші атрибути.

3. Порівняння сусідніх пікселів з поточною областю. Якщо вони відповідають критеріям схожості, їх додають до зони.

4. Продовжуйте збільшувати область, додаючи сусідні пікселі, які відповідають критеріям подібності, і повторюйте цей крок, доки область не припинить рости.

‘

5. Повторіть процес для інших необроблених пікселів або областей.

• Вибором меж.

Цей підхід передбачає використання операторів градієнта відповідно до кількох кроків:

1. Для визначення фактів для виявлення меж використовується процес поділу за порогами.

2. Потім пікселі, визначені як межі, з’єднуються, щоб утворити замкнутий контур, що охоплює відповідну область.

3. При прямому методі сегментації шляхом виділення меж передбачається, що вихідне зображення використовує градієнтні фільтри (Роберта, Лапласа та ін.).

4. Для меж простої форми (прямих ліній, дуг тощо) можна використовувати процес апроксимації вихідного зображення градієнта через будь-яку функцію параметра.

5. Векторизація.

Векторизація в обробці зображень у CV відноситься до процесу перетворення растрового зображення (складеного з пікселів) у векторне зображення, яке складається з векторних об’єктів, таких як лінії, криві та багатокутники. Це дозволяє зображенням бути у векторному форматі, що має кілька переваг, зокрема:

^ Масштабованість. Векторні зображення можна масштабувати без втрати якості. Оскільки вони зберігаються як математичні об’єкти, їх можна збільшувати або зменшувати без розмиття чи сплутування країв.

^ Операції з об’єктами. Векторні зображення дозволяють виконувати різноманітні операції над окремими об’єктами, такі як переміщення, обертання, зміна розміру тощо. Це важливо, коли потрібно відредагувати або змінити окремі елементи зображення.

^ Менше пам'яті. Векторні зображення зазвичай займають менше пам'яті, ніж растрові. Це особливо важливо під час роботи з великими наборами даних або передачі зображень через мережу.

^ Інтерактивність. Векторні зображення дозволяють додавати інтерактивні елементи, такі як кнопки, посилання, анімації тощо, що розширює можливості роботи з зображеннями.

‘

6. Морфологічні перетворення.

Це техніка обробки зображень, яка використовується для виділення контурів і заповнення порожніх областей на зображенні після попередньої обробки.

До основних морфологічних операцій належать розмивання, розширення, розкриття та закриття тощо.

Основні способи морфологічного перетворення [11]:

1. Ерозія: шляхом зменшення розміру об’єктів на зображенні. Ця операція видаляє дрібні структури, шуми та дрібні деталі. Ерозія корисна для видалення шуму та відтворення областей об’єктів.

2. Розширення: шляхом збільшення розміру об’єктів на зображенні. Ця операція розширює та з’єднує об’єкти, заповнюючи прогалини та відновлюючи втрачені деталі. Це розширення корисне для об’єднання об’єктів і виправлення ділянок із погано визначеними контурами.

3. Операція відкриття: Операція відкриття - це послідовне застосування корозії та розширення. Використовується для видалення шумів і дрібних деталей, а також розриву зв'язків між об'єктами.

4. Операція закриття: операція закриття – це послідовне застосування розширення та ерозії. Він використовується для заповнення прогалин у великих об'єктах і відновлення зв'язків між об'єктами.

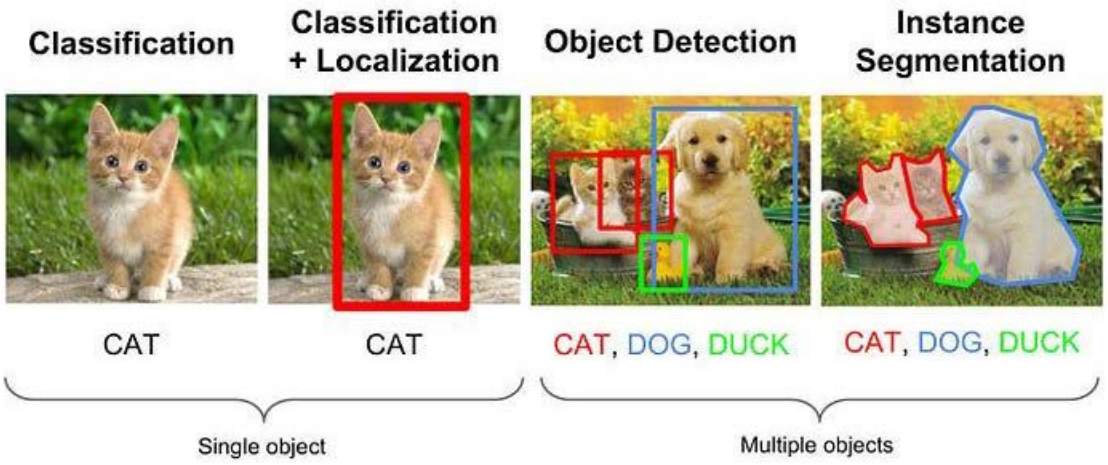


Рисунок 1.6 – Відмінності між класифікацією зображення, класифікацією + локалізацією, виявленням об’єкта та сегментацією [12].

‘

Для більш детального розуміння необхідно проаналізувати кожен процес виявлення об’єктів, який можна візуально порівняти за допомогою рисунка (1.6).

1.2.2 Класифікація.

Коли дані з мітками доступні, є методи, які можуть допомогти вибрати найважливіші функції для завдань машинного навчання. Вони дають змогу визначити ознаки, які найбільше впливають на результати прогнозування, тим самим покращуючи якість моделі.

У випадку немаркованих даних також існують методи вибору ознак, які використовують різні критерії для оцінки важливості ознак. Ці методи дають змогу виділити ті ознаки, які важливі в контексті досліджуваної проблеми.

Крім того, відповідні функції, виявлені за допомогою ненавченої евристики, можуть бути корисними навіть у навчених моделях. Вони допомагають виявити інші закономірності та зв’язки в даних, які можуть бути недоступні для простого співвідношення між функціями та цільовими змінними.

• Найближчі сусіди (k-Nearest Neighbors, k-NN).

Це досить популярний метод для завдань класифікації об’єктів у машинному навчанні. Основна ідея цього методу полягає в тому, що подібні об'єкти зазвичай належать до одного класу. Тому, щоб класифікувати новий об'єкт, спочатку обчисліть відстань між ним і всіма об'єктами в навчальному наборі. Найближчим сусідом є об'єкт з найменшою відстанню до нового об'єкта. Категорія цього найближчого сусіда стає прогнозованою категорією нового об'єкта [13]. Приклад наведено на рисунку (1.8).

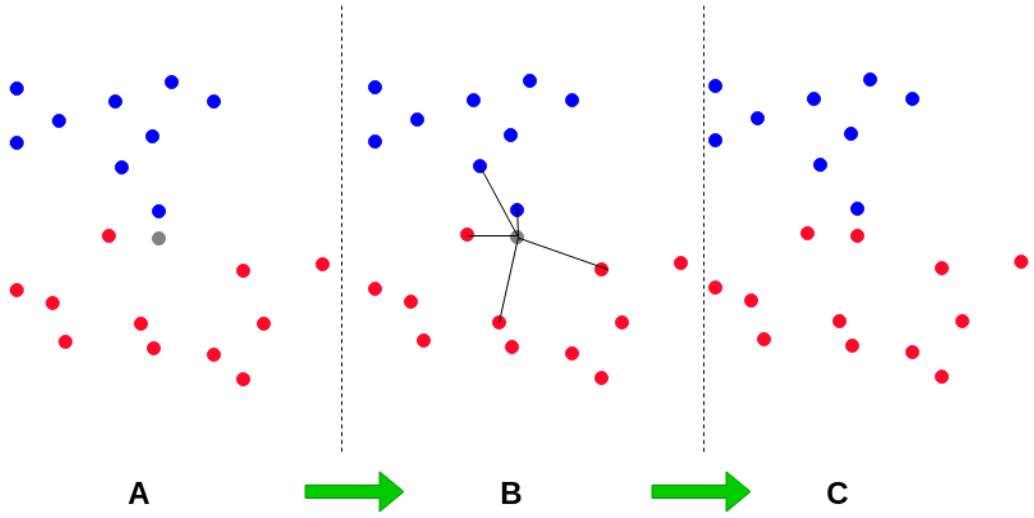


Рисунок 1.8 – Приклад класифікації з використанням методу п’яти найближчих сусідів.

• Методи на основі глибокого навчання.

Глибоке навчання, мабуть, є найефективнішим методом для класифікації об’єктів, виявлення та відстеження.

Основна ідея глибокого навчання полягає в тому, щоб послідовно використовувати шари нейронної мережі для поступового вилучення функцій високого рівня. Ранні шари виявляють прості елементи, такі як грані або кути, тоді як пізніші шари об’єднують ці прості елементи, щоб показати складнішу форму та структуру об’єкта. Як правило, глибокі нейронні мережі мають кілька згорткових шарів, які виконують згортку та об’єднання, щоб зменшити розмірність зображення, і повністю зв’язані шари, які виконують класифікацію на основі отриманих ознак [14].

У терміні «глибоке навчання» слово «глибоке» означає використання багатьох прихованих шарів у моделі нейронної мережі.

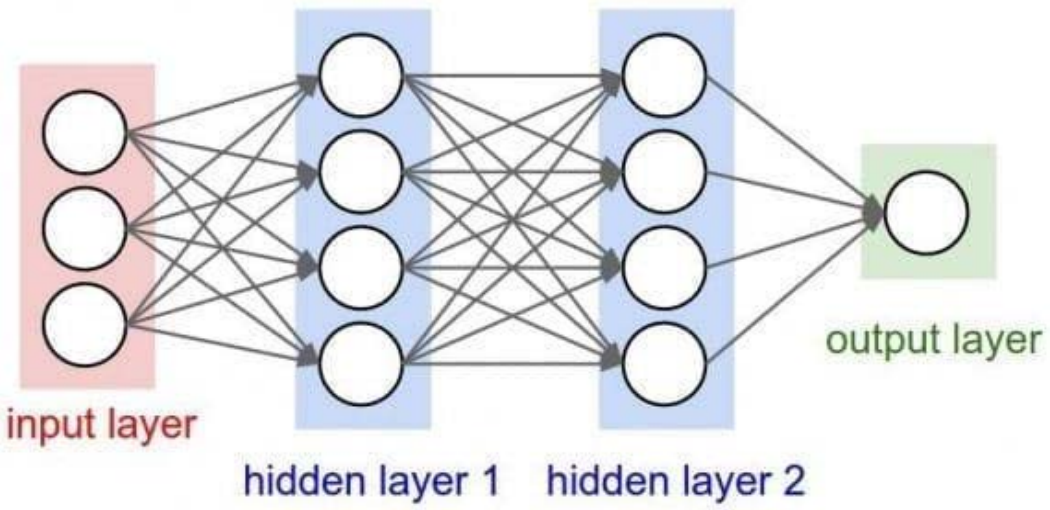


Рисунок 1.9 – Глибока нейронна мережа з двома прихованими шарами.

Давайте ближче розглянемо нейронну мережу, зображену на рисунку (1.9). На початковому етапі обробки вхідних даних моделі нейронної мережі ми маємо вхідний рівень, що містить чотири нейрони, які отримують відповідну вхідну інформацію. Потім дані передаються на перший шар прихованих нейронів, де виконуються математичні розрахунки. Кількість шарів і нейронів вибирається виходячи з конкретного завдання, вимог до точності даних і складності. На завершальному етапі вихідний рівень забезпечує результати прогнозування. Навчання глибоких нейронних мереж вимагає великих обсягів даних з мітками категорії. Процес навчання передбачає оптимізацію вагових коефіцієнтів мережі за допомогою зворотного поширення, яке оновлює вагові коефіцієнти, щоб мінімізувати функцію втрат між прогнозованою та справжньою мітками.

Переваги використання глибоких нейронних мереж і порівняння з машинним навчанням показані на рисунку (1.10):

- Можливість автоматичного вилучення ознак високого рівня з вхідних зображень і використання цих функцій для класифікації об’єктів.

- Здатність працювати з великими наборами даних.

- Здатність до високоточного розпізнавання.

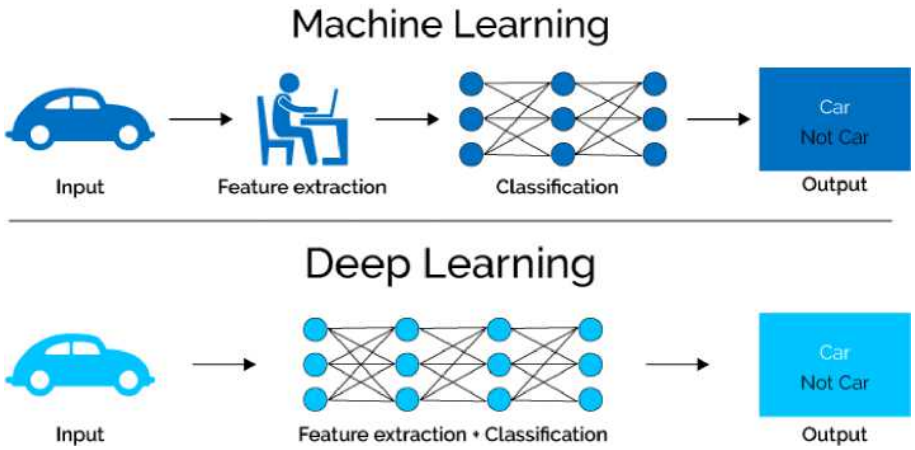


Рисунок 1.10 – Порівняння машинного та глибокого навчання [15].

1.2.3 Локалізація.

Крім того, не менш важливий процес розпізнавання об'єктів. Він передбачає визначення розташування об’єктів на зображенні чи відео. локалізація

У тому числі знайти координати або межі місцевості, де знаходиться об'єкт інтересу.

• Дескрипторний підхід.

Підхід до завдань CV для розпізнавання та локалізації об’єктів на зображеннях. Він використовує спеціальні описові характеристики, які називаються дескрипторами, для представлення статичних і динамічних об’єктів.

Щоб краще зрозуміти цей підхід, можна виділити основні етапи дескрипторних методів [16]:

1. Виділіть особливості. По-перше, функції витягуються із зображення або області інтересу. Алгоритм методу вибирає ключові точки або області зі стабільними характеристиками на зображенні для масштабування, обертання та освітлення.

2. Опис функції. Після визначення ознак опишіть їх за допомогою дескрипторів. Дескриптори містять інформацію про форму, текстуру чи інші характеристики об’єкта. Найпоширеніші дескриптори включають дескриптори SIFT, дескриптори SURF, дескриптори ORB тощо. Ці дескриптори генерують вектори ознак, які можна використовувати для порівняння та ідентифікації об’єктів.

3. Порівняння дескрипторів. Після отримання дескрипторів можна проводити порівняння для виявлення подібності між об’єктами. Різні міри відстані, такі як

евклідова відстань або косинусна відстань, часто використовуються для порівняння векторів ознак. Об'єкти зі схожими дескрипторами вважаються подібними.

4. Позиціонування об'єктів. Останнім кроком є визначення місця розташування об’єкта на зображенні. Цього можна досягти шляхом зіставлення дескрипторів об’єктів у зображенні з дескрипторами, що зберігаються в базі даних або навчальному наборі. Таке відображення дозволяє визначити положення і масштаб об'єктів.

• Методи на основі нейронних мереж.

Нейронні мережі (NM) можуть автоматично визначати розташування та розмір об’єктів на зображенні без ручного вибору функцій. Це досягається створенням спеціальної архітектури NM, налаштуванням кількості шарів і навчанням набору даних із відповідними мітками. Після навчання НМ здатний локалізувати об’єкти на нових зображеннях, надаючи передбачення щодо їх розташування та розміру у формі прямокутників, що описують межі цих об’єктів [17].

Після деяких досліджень вдалося виявити наступні переваги використання НМ, які спостерігалися в задачах локалізації об’єктів:

^ Автоматичне навчання функцій. Тобто NM може автоматично вивчати функції, необхідні для цільової локалізації, не вибираючи функції вручну.

^ Навички узагальнення. Навчання на великих обсягах даних дозволяє нейронним мережам узагальнювати та ефективно локалізуватиОб’єкти на нових зображеннях були речами, яких вона ніколи раніше не бачила.

/ Висока точність. Методи на основі ЯМ можуть досягти високоточної локалізації цілі, особливо при навчанні на великих і репрезентативних наборах даних.

1.2.4 Виявлення об'єктів.

Завдання виявлення об’єктів полягає у виділенні кількох об’єктів на зображенні шляхом визначення координат обмежувальних рамок і класифікації цих об’єктів серед великої кількості раніше відомих категорій. На відміну від класифікаційної локалізації кількість об'єктів на зображенні заздалегідь невідома.

• Двоетапний підхід.

Ці методи також називають методами на основі регіонів і являють собою

‘

двоетапний підхід до виявлення об’єктів. На першому етапі вибірковий пошук використовується для визначення регіонів інтересу (ROI) – областей, які можуть містити об’єкти. На другому етапі вибрані регіони проходять через класифікатор, щоб визначити, що вони належать до певних категорій, і регресор, щоб більш точно встановити розташування обмежувальної рамки.

Приклад двоетапного підходу:

1.R-CNN

R-CNN (Region-based Convolutional Network) — це алгоритм виявлення цілей на основі згорткової нейронної мережі. Алгоритм роботи R-CNN наведено на рисунку (1.11):

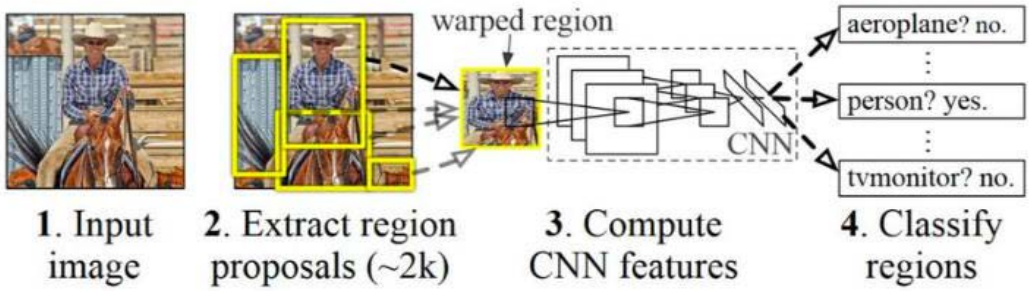


Рисунок 1.11 – Алгоритм роботи R-CNN [18].

2. Швидкий R-CNN.

У Fast R-CNN вхідними даними є зображення, а також запропоновані об’єкти. По-перше, дані передаються через повністю згортову мережу для отримання згорткових карт функцій. Потім для кожного запропонованого об’єкта на карті об’єктів використовується рівень об’єднання регіону інтересу (RoI), щоб отримати вектор ознак фіксованої довжини. На рисунку (1.12) показаний принцип роботи.

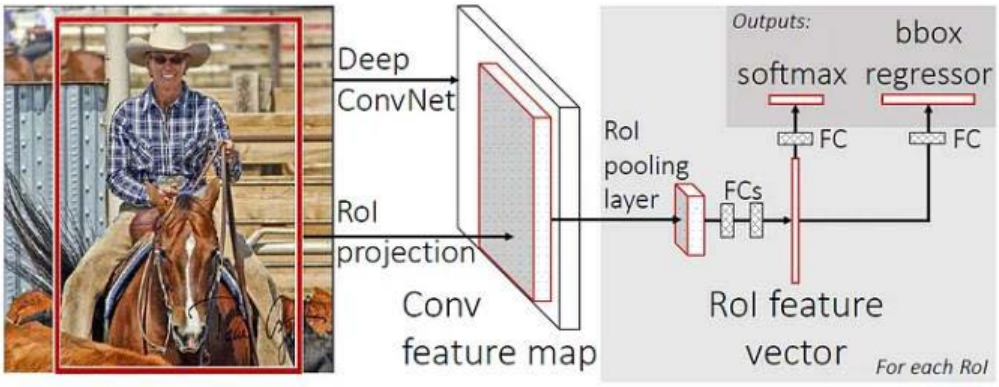


Рисунок 1.12 – Швидкий алгоритм роботи R-CNN [18].

3. Швидший R-CNN

Faster R-CNN складається з двох модулів: Region Proposal Network (RPN) - Region Proposal Network і Fast R-CNN Detector. Два модулі об’єднані в мережу та викладаються наскрізно.

Генерація пропозицій регіону виконується за допомогою розрідженої мережі, яка просувається через останній загальний рівень згорткової мережі. Невеликі мережі приймають вхідні карти функцій розміром (n x n). Кожне ковзне вікно порівнюється з елементами нижчої роздільної здатності та проходить через два повністю пов’язані рівні, як і раніше: рівень класифікації та рівень регресії.

• Однокроковий метод.

У порівнянні з першим методом, однокроковий метод не використовує окремий алгоритм для генерації регіонів. Натомість вони передбачають координати обмежувальної рамки з різними характеристиками, такими як результати класифікації та достовірність. Потім вони регулюють положення цих рамок для отримання кінцевого результату.

Одноетапний приклад:

1. ЙОЛО.

Одним із найпопулярніших методів виявлення об’єктів сьогодні є YOLO (You Only Look Once), який здатний обробляти відео в режимі реального часу з мінімальною затримкою, зберігаючи прийнятну точність. І, як випливає з назви, для виявлення всіх об’єктів на зображенні потрібно лише одне просте уточнення.

За час існування цього алгоритму з'явилося багато версій YOLO, і його автори

підвищували точність і швидкість своєї роботи, які є дуже важливими факторами при виявленні об'єктів.

2. Твердотільний накопичувач.

Через кілька місяців після випуску YOLO з’явилися одноразові детектори з кількома коробками (SSD), які стали цінною альтернативою.

Подібно до YOLO, SSD використовує одноканальну мережу для виявлення об’єктів. Ця модель на основі CNN передає вхідне зображення серії згорткових шарів, які генерують обмежувальні прямокутники різного розміру.

У SSD для навчання використовуються позитивні та негативні мітки об'єктів. Метод жорсткого негативного видобутку використовується після застосування немаксимального придушення (NMS) до позначених об’єктів. Цей метод вибирає негативний приклад із найбільшою втратою довіри та зберігає співвідношення позитивних і негативних прикладів не більше 1:3. Це допомагає пришвидшити оптимізацію та забезпечує стабільну фазу навчання.

1.2.5 Сегментація екземпляра.

Сегментація екземплярів — це метод виявлення, сегментації та класифікації кожного окремого об’єкта на зображенні. Замість простого окреслення області, що містить об’єкт, сегментація екземплярів дозволяє вам розділити кожен об’єкт на окремі сегменти, які відповідають його межам.

Крім того, сегментація екземплярів також може класифікувати кожен фрагмент об’єкта, щоб отримати інформацію про типи об’єктів, присутніх на зображенні, як показано на рисунку (1.13).

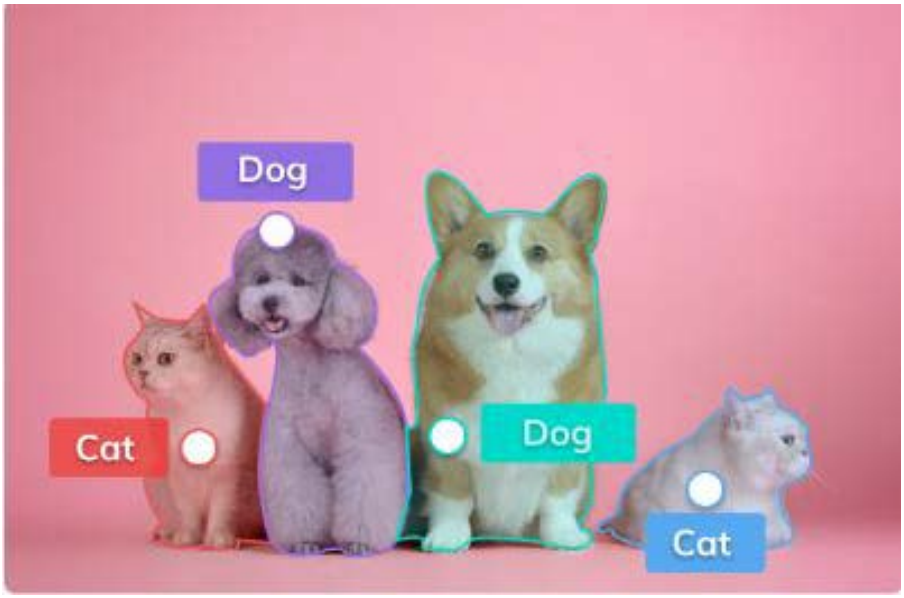


Рисунок 1.13 – Приклад того, як працює сегментація екземпляра.

Порівняно із семантичною сегментацією та сегментацією екземплярів, сегментація екземплярів створює більш детальний вихідний формат. Дивіться рисунок (1.14) для порівняння.

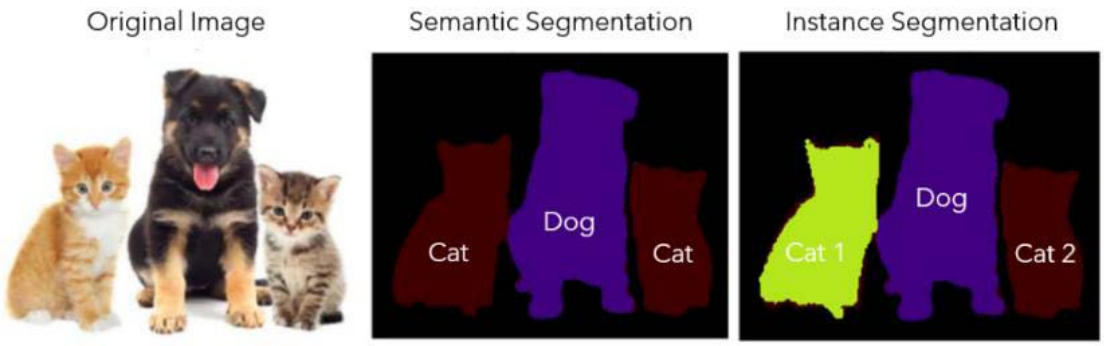


Рисунок 1.14 – Порівняння семантичної сегментації та сегментації екземплярів.

• Маска R-CNN.

Це двоетапний підхід до аналітичного методу виявлення, описаного вище. Mask R-CNN — це вдосконалена модель сегментації екземплярів, яка обробляє три перетворені виходи. Він містить мітки класів і зміщення обмежувальної рамки, подібно до методу Faster R-CNN, але також має третю гілку, яка генерує маски об’єктів. Ця маска забезпечує більш точне просторове розташування об'єктів, які необхідно розрізнити [18].

‘

1.3 Формалізація проблеми та постановка деяких питань дослідження.

Метою даної роботи є розробка програмної системи, яка виявлятиме заданий об’єкт на основі даних ДЗЗ із відкритих джерел інформації.

Дії, які необхідно виконати перед початком розробки системи програмного забезпечення, повинні включати наступне:

1. Виберіть службу, яка має доступ до запущеного образу DZZ (наприклад, MAXAR або Sentinel Hub).

2. Оберіть поле дослідження чи об’єкт дослідження для подальшої обробки та ідентифікації.

3. Зберігайте дані DZZ у відкритому коді для зручного використання зображення.

4. Поліпште якість зображення, щоб відфільтрувати шум і усунути розмитість, щоб покращити чіткість і деталізацію об’єкта.

5. Вам потрібно використовувати векторизацію, щоб перетворити растрове зображення у векторне представлення, що складається з геометричних об’єктів, таких як лінії та криві. Це спростить зображення та виділить геометричні особливості об’єкта, що допоможе в подальшому аналізі та ідентифікації об’єкта.

6. Виконайте операції сегментації або морфологічні перетворення. Це може допомогти диференціювати цільові об’єкти.

7. Порівняйте рисунки.

8. Виявляти об'єкти за допомогою деяких\певних методів, які дозволяють виявляти об'єкти.

9. Створити інтерфейс програмної системи для полегшення керування її функціями.

Усі ці дії та операції із зображеннями відображені в основних положеннях першої частини «Аналіз відомих методів виявлення заданих об’єктів за даними ДЗЗ з відкритих джерел інформації»:

1. Проаналізувати відомі технічні рішення виявлення даного об’єкта за даними ДЗЗ з відкритих джерел інформації.

2. Проаналізувати відомі методи розпізнавання об’єктів у задачах комп’ютерного зору.

‘

Процес вивчення та аналізу різноманітних методичних рішень виявлення заданих об’єктів на основі відкритих інформаційних джерел даних ДЗЗ, а також методичних прийомів ідентифікації об’єктів у задачах комп’ютерного зору сприяє розумінню роботи різних методів, які допомагають досягти цілі та здібності.

Бажаний спосіб розробки:

1. Виберіть бізнес-послуги дистанційного зондування Землі, щоб надавати швидкі та найновіші зображення.

2. Використовуйте мову програмування, яка надає інструменти для вирішення проблем комп’ютерного зору.

3. Застосовуйте методи обробки зображень, щоб покращити якість і зберегти важливу інформацію

4. Використовуйте методи виявлення об'єктів, засновані на особливих характеристиках об'єктів, які допоможуть точно ідентифікувати об'єкти на зображеннях.

Небажані шляхи розвитку:

1. Використовуйте методи, які негативно впливають на швидкість вашої програмної системи. Важливо уникати ресурсомістких або недостатньо оптимізованих алгоритмів.

Дотримання цих рекомендацій дозволить вам ефективно реалізувати свої проєкти та досягти високої якості обробки зображень та розпізнавання об’єктів.

‘

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ ДЛЯ ВИЯВЛЕННЯ КОНКРЕТНИХ ОБ'ЄКТІВ ЗА ДАНИМИ ВІДКРИТИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

2.1 Вибір методів і прийомів практичної реалізації процесу ідентифікації даного об'єкта за даними ДЗЗ з відкритих джерел інформації.

Після аналізу визначено основні методи та прийоми, які сприятимуть розробці конкретних програмних систем виявлення об’єктів на основі даних ДЗЗ із відкритих джерел інформації.

Мова програмування Python добре підходить для завдання ідентифікації об’єктів на зображеннях. Ця мова програмування має велику кількість бібліотек, призначених для обробки зображень і методів CV.

Якщо говорити про переваги Python перед його основним конкурентом C++ в задачах комп'ютерного зору, то потрібно звернути увагу на наступні деталі:

Python має розширені бібліотеки, такі як NumPy, Matplotlib, Pandas і SciPy, які надають інструменти для обробки та аналізу зображень. Вони також тісно інтегровані з OpenCV, спрощуючи складні завдання розпізнавання окремих об’єктів.

Недоліком мови програмування Python порівняно з C++ є продуктивність. Але продуктивність залежатиме від конкретного завдання, наприклад, якщо ви використовуєте Python з оптимізованими бібліотеками для векторизації та операцій з масивами, різниця в продуктивності може бути незначною.

Зображення певних цікавих місць будуть знаходитися за допомогою сервісів і можливостей MAXAR і Sentinel Hub, які надаватимуть оперативні знімки, а також Google Maps (еталонні знімки із супутників). Більш детально про ці сервіси було описано в першому розділі вище, описуючи їх функціональні можливості та переваги порівняно з аналогічними програмами цього типу. Фотографії з цих служб зберігаються в каталозі, який містить назву фотографії та інтервал часу, протягом якого вона була зроблена.

Python і його набір інструментів і бібліотек допоможуть читати зображення з каталогу і зберігати зміни під час цифрової обробки.

Інформація про бібліотеки, які будуть використовуватися при розробці програмної системи:

1. OpenCV — це бібліотека, яка забезпечує обробку зображень, завантаження, збереження та функції виведення для файлів JPEG і PNG.

2. NumPy - Допоможе в роботі з масивами даних разом з математичними функціями для виконання операцій над масивами.

3.Matplotlib — це бібліотека візуалізації даних. Надають можливість відображати графіки та порівнювати рисунки між собою.

4. Подушка - також дозволяє завантажувати, зберігати та експортувати знімки. Крім того, це дозволяє виконувати операції над зображеннями, наприклад покращувати їх якість.

5. CustomTKinter – бібліотека для створення графічних інтерфейсів програм. Це дозволяє створювати інтерфейси користувача за допомогою бібліотеки Tkinter і ваших власних налаштувань. Це також розширює функціональні можливості Tkinter і дозволяє створювати більш привабливі та індивідуальні інтерфейси.

Метод, заснований на розпізнаванні образів за допомогою векторизації піксельних зображень, буде використовуватися для виявлення заданого об'єкта через DZZ. Це буде досягнуто за допомогою розпізнавання контурів і знаходження контурів на зображенні шляхом підрахунку заданого об’єкта.

На додаток до цього методу розпізнавання буде застосовано метод використання шаблонів для пошуку та визначення місцезнаходження об’єктів на зображеннях. Цей метод (відповідність шаблону) порівнює зображення шаблону з вхідним зображенням, де кожен піксель відображає, наскільки добре околиці цього пікселя відповідають шаблону.

2.2 Розробка програмної системи для виявлення конкретних об'єктів на основі даних ДЗЗ з відкритих джерел інформації.

У цьому розділі розробка вимог, архітектурний дизайн програмного забезпечення та сама його реалізація описані більш детально за допомогою фрагментів коду для

кращого розуміння.

2.2.1 Розробка вимог.

Щоб почати впровадження програмної системи, спочатку потрібно сформулювати вимоги, щоб уникнути проблем і непорозумінь під час розробки. Рішення цієї проблеми полягає у створенні пунктів, які описують кожну конкретну вимогу.

Для перевірки були обрані полігони та сонячні електростанції. На рисунку (2.1) можна побачити етапи обробки зображення, які призведуть до виявлення об’єктів на зображенні.

Рисунок 2.1 – Етапи обробки.

Проєкт, що забезпечує програмну систему об’єкта (полігону) такими функціями:

‘

1. Зчитування даних. Ця опція дозволяє вибрати зображення, а потім обробити його і виявити на ньому заданий об'єкт.

1) Вхідне зображення має зберігатися в якомусь каталозі, який матиме назву служби відкритого джерела інформації - Sentinel Hub,

2) Можливість вибрати потрібну картинку з одного з каталогів з назвою послуги.

3) Якщо знімок не завантажено, наведені нижче операції неможливо виконати.

4) Після завантаження зображення на екрані повинні з'явитися кнопки для операцій обробки.

2. Підвищення якості. Після цієї операції зображення повинно мати більший контраст і кращу візуальну якість.

1) При цьому вхідні зображення нижчої якості мають виглядати детальнішими та чіткішими.

3. Відтінки сірого. Спрощена інформація про колір, яка зберігає лише яскравість пікселя, незалежно від його кольору.

1) Перехід від кольорових зображень до монохромних.

2) Значення яскравості кожного пікселя на зображенні коливається від 0 (чорний) до 255 (білий).

4. Розпізнавання об'єктів. Об'єкт, який потрібно виявити, буде знайдено.

1) Створіть шаблон зображення, що містить вказаний об’єкт.

2) Порівняння зображення шаблону та вхідного зображення.

3) Виявлення об'єктів.

5. Зберігання даних, які щоразу обробляються та ідентифікуються.

1) Після кожної операції над зображенням зображення має бути збережено в попередньо створеному каталозі для зручного перегляду.

6. Іміджеве порівняння різних сервісів.

1) Порівняння слід проводити через службу MAXAR із Sentinel Hub.

2) Для цієї функції необхідно створити окреме вікно.

Проєкт, який забезпечує наступні функції для програмної системи об’єкта – сонячної станції (операції, які повторюються з попереднім проєктом, нижче описані не будуть, оскільки вони схожі):

‘

1. Зчитування даних.

2. Підвищення якості.

3. Відтінки сірого.

4. Виявлення краю. Необхідно повернути зображення, яке правильно визначає краї.

1) Виберіть нижній поріг для виявлення країв. Тобто це значення має керувати початком інтенсивності пікселів, починаючи з якого пікселі вважатимуться відображенням країв.

2) Виберіть верхній поріг для виявлення краю.

5. Морфологічні перетворення. Порожні місця між об'єктами необхідно заповнити.

1) Використовуйте операцію закриття, щоб заповнити отвори всередині об’єкта та закрити проміжки на межах об’єкта.

6. Розпізнавання об'єктів. Об'єкт, який потрібно виявити, буде знайдено.

1) Знайдіть контур за морфологічним перетворенням.

2) Контур потрібного контуру буде налаштований відповідно до параметрів (кількість вершин і площа контуру).

7. Зберігання даних, які щоразу обробляються та ідентифікуються.

Озброївшись цими моментами, ви зможете зрозуміти, як виглядає дизайн архітектури програмного забезпечення для кожної області виявлення.

2.2.2 Проєктування архітектури програмного забезпечення.

На цьому етапі буде описана архітектура, в якій будуть методологічні та технічні рішення для завдання, які були розглянуті в процесі дослідження та аналізу.

В архітектурному проєктуванні та будівництві необхідно враховувати формалізацію завдань та розробку вимог.

На наступному рисунку (2.2) показана структурна схема програмної системи.

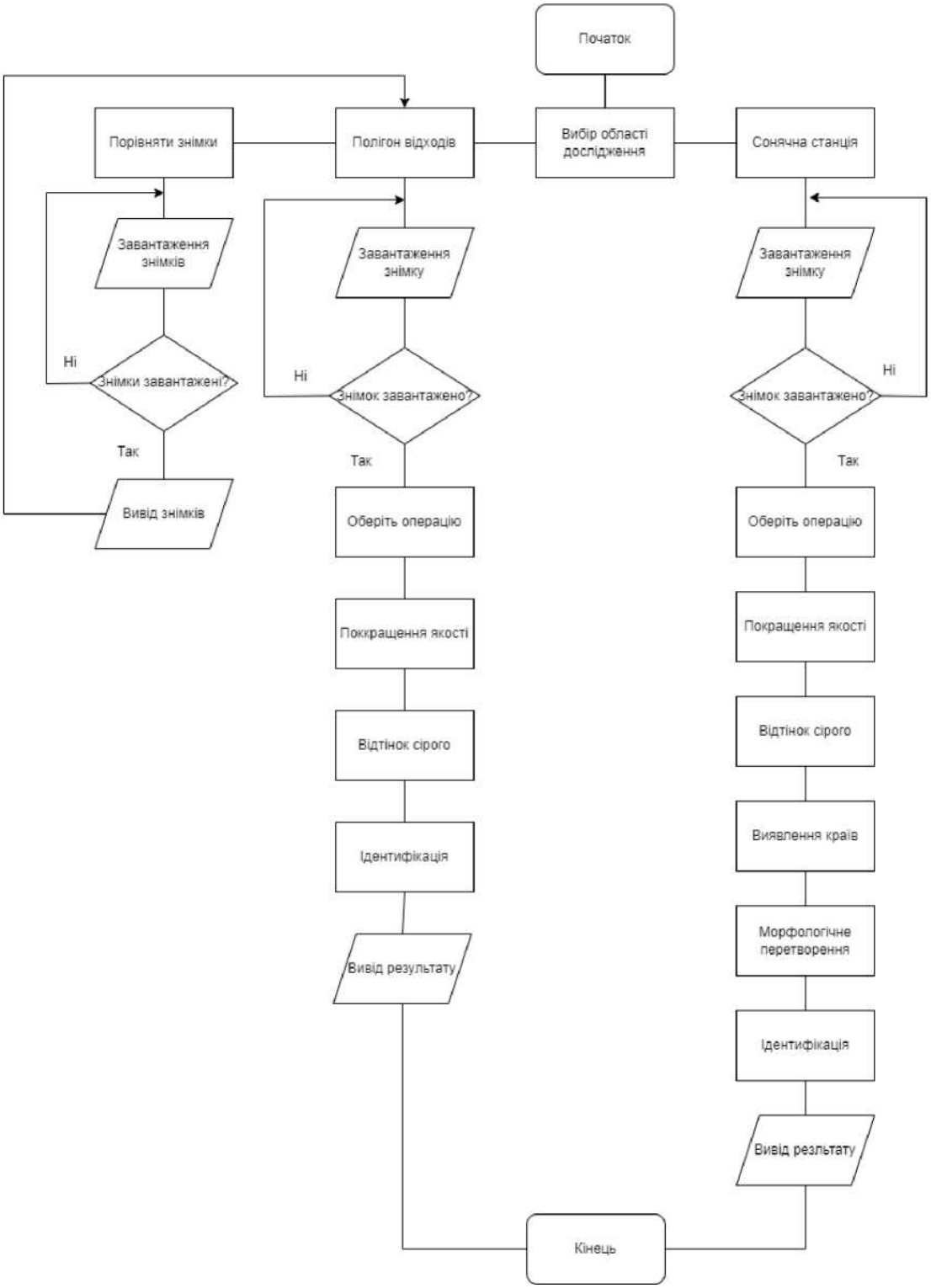


Рисунок 2.2 – Алгоритм роботи програмної системи.

Пояснення рішення (2.2):

Перший етап – вибір досліджуваної території. Після вибору, наприклад, полігону, наступний етап – завантаження зображення.

Для подальших операцій дистанційного зондування Землі функціональність програми буде недоступна без завантаження зображень. Ми попередньо зберігаємо це

зображення в створеному нами каталозі, де будуть розташовані всі зображення з сервісів MAXAR і Sentinel Hub. Далі настав час підрахувати цифри. Почати варто з покращення якості картинки, яка досить низька. Коли зображення стане кращим, слід переходити до основної операції, яка в подальшому вплине на результати виявлення об’єкта, в даному випадку полігону.

У міру поліпшення зображення слід переходити до отримання монохромних зображень або зображень з відтінками сірого. За допомогою цієї операції зображення можна розділити на пікселі в діапазоні від 0 до 255.

На підставі монохромного зображення будуть виконуватися операції виявлення об’єктів. Виконується порівняння зображення шаблону з вхідним зображенням, тобто пікселі порівнюються за схожістю.

Для полегшення доступу до зображень, отриманих після попередніх операцій, буде створено каталог, до якого ми будемо звертатися для перегляду результатів.

Після завершення всіх операцій обробки зображення відобразяться результати детектування, тобто позиціонування об’єкта дослідження.

Крім того, є операція, яка дозволяє порівнювати зображення, натиснувши на неї, відкриється нове вікно, де можна завантажити зображення. Вам потрібно завантажити зображення, щоб відобразити його на екрані. Якщо операція завершена, ви можете закрити вікно та повернутися до вікна «Полігон».

Якщо всі операції у вікні «Полігон» виконано, ви можете повернутися до головного вікна для вибору наступної області дослідження – Сонячна станція.

Щоб уникнути повторення пояснень операцій під час попереднього вивченого вікна, вони лише згадуються без детального огляду.

Виберіть вікно Сонячна станція. Спочатку зображення зчитується з каталогу, і якщо зображення вибрано, функції вікна дозволяють виконувати операції над зображенням, а потім подібні операції обробки зображення - покращення якості та відтінки сірого.

На основі монохромних зображень їх виявлення використовується для проведення меж. Після побудови кордонів застосуйте морфологічну трансформацію, яка допоможе заповнити порожні простори між невеликими об'єктами.

‘

Нарешті, на основі вихідних даних і результатів попередніх операцій обробки об’єкти на зображенні виявляються шляхом пошуку контурів і малювання їх відповідно до певних налаштувань.

2.2.3 Програмна реалізація програмного забезпечення.

Проведено аналіз, обрано техніко-методичні рішення в задачі CV, сформовано розробку вимог та побудовано архітектурне рішення з різними напрямками дослідження, Відповідно різні й підходи до вирішення проблем.

На основі попередніх досліджень у цьому розділі буде розглянуто конкретну реалізацію програмної системи, яка дозволить виявляти певні об’єкти на знімках дистанційного зондування Землі з відкритих джерел інформації.

Перед початком реалізації рішень місії необхідно зібрати дані у вигляді супутникових знімків, які будуть об'єктом дослідження. Збір зображень відбуватиметься з джерел еталонних зображень та оперативних джерел (тобто джерел, які надають супутникові зображення в різні проміжки часу), таким чином зберігаючи їх актуальність.

За допомогою Google Maps ми знайдемо об’єкт дослідження як наш майбутній напрямок, ці зображення будемо називати референсами.

Першим об’єктом вивчення та обстеження стане полігон твердих побутових відходів № 5, який є найбільшим в Україні (площею 65,2 га за даними на 12 березня 2021 року), розташований на півдні Києва, поблизу села Орба Підігірці, Обухівський район.

Рисунок 2.3 – Полігон твердих побутових відходів №5.

Зараз ми шукаємо цей об'єкт на сервісі МАКСАР. Необхідно виділити квадратиком область, де знаходиться об'єкт, що дозволить переглядати і завантажувати всі картинки різних часових інтервалів. Однак необхідно звернути увагу на деякі показники, які відіграють важливу роль у подальшому виявленні об'єктів. Наприклад, якщо виділена область знімка має високу каламутність, тобто більше 25-30%, то знайти об'єкти на знімку навіть візуально буде практично неможливо. Приклад зображення, яке не вдалося обробити, показано на рисунку (2.4).

‘

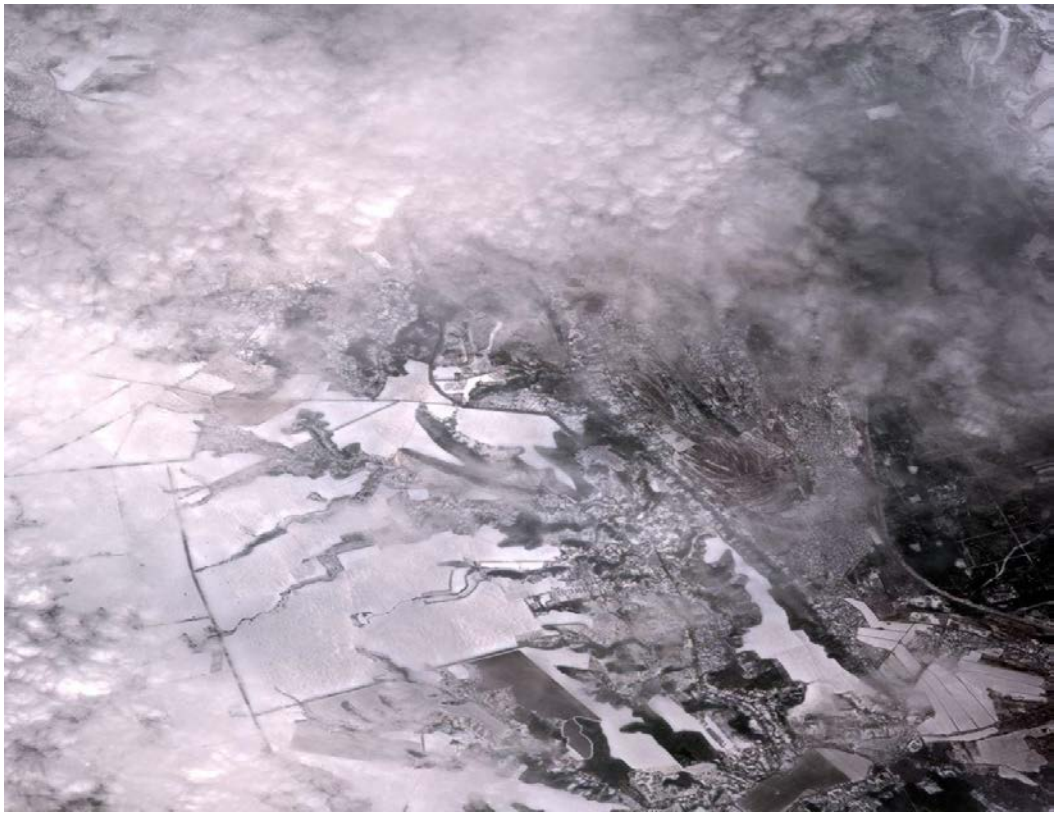


Рисунок 2.4 – Приклад зображення високої хмарності (33%) MAXAR 2 лютого 2022 року.

Після налаштування параметра хмарності (тобто зображення з найменшою хмарністю) мені вдалося знайти досить гарне зображення, де добре видно звалище. Приклад зображення MAXAR на рисунку (2.5) з мінімальною хмарністю дає вам відчуття різниці між рисунком (2.4).

‘

Рисунок 2.5 – Приклад MAXAR зображення низької хмарності (0%) 28 травня 2005 року. 2024 рік.

Схожий алгоритм роботи для збору зображень із Sentinel Hub. Ми також налаштовуємо параметр хмарності та вибираємо доступні зображення за датою з природними кольорами, знайомими людям, як показано на рисунку (2.6).

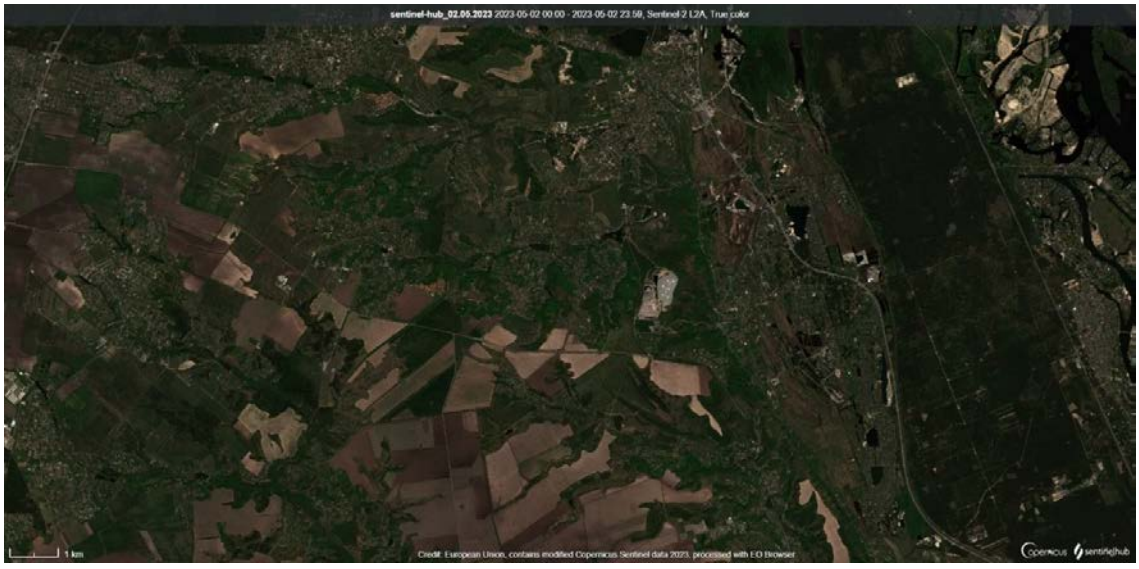


Рисунок 2.6 - Приклад супутникового знімка Sentinel-2 від 02.05. 2024 рік.

‘

Другий напрямок досліджень – сонячні електростанції. Сонячна станція менша за розміром, ніж звалище, але є ідеальною територією для застосування методів виявлення позамежних об’єктів, оскільки її форма має чітко визначену структуру.

Рисунок 2.7 – Приклад зображення сонячної станції на Google Maps.

Варто зазначити, що сонячна станція є невеликим об’єктом і її неможливо побачити з сервісу оперативних зображень через низьку якість зображення та недостатнє масштабування, тому достатньо скористатися лише сервісом Google Maps, який надає зображення із супутника планети.

Перш ніж будувати діаграми класів, описувати класи та методи, я хочу зосередитися на бібліотеках, які допомагають досягти мети, яка полягає в обробці зображень, відображенні, читанні файлів, розробці інтерфейсів і, що найважливіше, виявленні об’єктів.

Програмна система складається з трьох класів Main(), Landfill_Window() і Solar_Station_Window(). Ви можете дізнатися більше про класи та їхні методи за допомогою діаграми класів системи програмування, як показано на рисунку (2.8).

‘

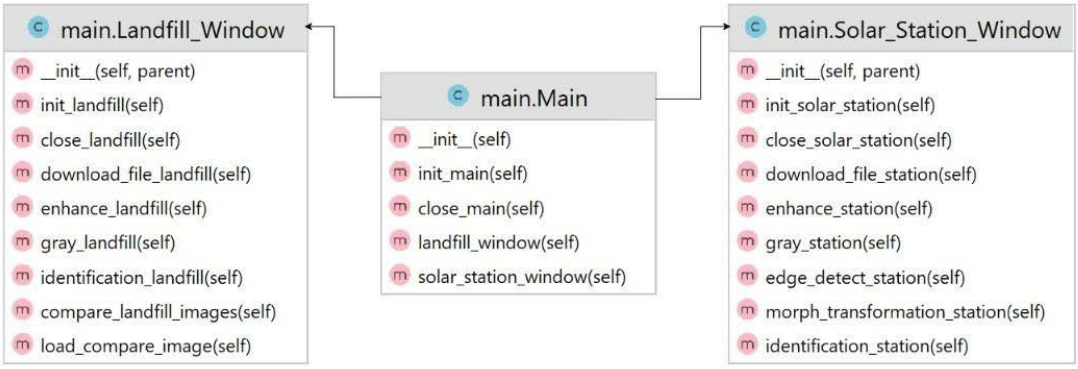


Рисунок 2.8 – Діаграма класів програмної системи.

Клас Main() є основним класом програми (він діє як головне вікно в інтерфейсі) і успадковує клас ctk.CTk. У конструкторі __init__(self) викликається конструктор батьківського класу, а потім викликається метод init_main(), який встановлює головне вікно програми. Метод close_main(self) викликається, коли закривається головне вікно програми. Відобразиться діалогове вікно підтвердження із запитом на підтвердження виходу з програми. Якщо користувач підтверджує, що хоче вийти, використовуйте self.destroy(), щоб знищити вікно програми. Далі методи landfill_window(self) і solar_station_window(self) містять налаштування для відкриття додаткових вікон - коли ви натискаєте Waste Landfill, створюється нове Landfill_Window і встановлюється режим захоплення grab_set(). Подібна дія відбувається, якщо натиснути Сонячна станція. Також створіть нове вікно Solar_Station_Window і встановіть для нього режим grab_set().

Клас Landfill_Window є підкласом ctk.CTkToplevel і відповідає за вікно дослідження полігону. Це вікно відкривається поверх головного вікна, доступ до якого неможливий, доки вікно не закрито.

Метод класу __init__(self,parent) є конструктором класу Landfill_Window. Він викликає конструктор батьківського класу ctk.CTkToplevel, а потім викликає метод init_landfill(self), щоб встановити вікно звалища. Метод close_landfill(self) викликається, коли вікно дослідження полігону закрито. Відобразиться діалогове вікно підтвердження з проханням підтвердити закриття вікна. Якщо користувач підтверджує, що хоче закрити вікно, використовуйте self.destroy(), щоб знищити вікно. download_file_landfill(self) — це метод, який викликається, коли натискається кнопка

«Завантажити знімок» (self.load_landfill_button). Відкриється діалогове вікно для вибору файлу зображення. Після вибору файлу зображення він створює об’єкт зображення self.input_image за допомогою ctk.CTkImage та відображає зображення на екрані. Також створіть і розмістіть кнопки для обробки зображень (enhance_button, gray_button, ідентифікаційна_кнопка). Варто зазначити, що якщо зображення не завантажиться, подальша обробка та операції розпізнавання будуть неможливі.

Далі викликайте методи для обробки та ідентифікації знімка:

1) enhance_landfill(self) - викликається при натисканні кнопки «Покращення якості» (enhance_button).

Програмний код цього методу наведено на рисунку (2.9):

```
def enhance_landfill(self):
    if hasattr(self, 'input_image'):
        img = Image.open(self.filepath)
        enc_img = img.filter(ImageFilter.DETAIL)
        img_con = ImageEnhance.Contrast(enc_img)
        self.img_enh = img_con.enhance(1.3)
        self.img_enh.save('results\\landfill\\enhanced_image.jpg')
        self.img_enh.show()
    else:
        messagebox.showerror("Помилка!", "Спочатку завантажте знімок!")
```

Рисунок 2.9 – Демонстрація методу enhance_landfill(self).

2) Gray_landfill(self) - цей метод викликається при натисканні кнопки

«Тіні сірого» (gray_button). Він перетворює попередньо покращене зображення на відтінки сірого та відображає його.

Програмний код цього методу наведено на рисунку (2.10):

```
def gray_landfill(self):
    if hasattr(self, 'input_image') and hasattr(self, 'img_enh'):
        img = 'results\\landfill\\enhanced_image.jpg'
        self.image = cv2.imread(img)
        self.gray_img = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
        plt.figure(figsize=(17, 8))
        plt.imshow(self.gray_img, cmap='gray')
        plt.title('Gray Image')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\landfill\\gray_image_landfill.jpg", self.gray_img)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Покращення якості'!")
```

Рисунок 2.10 – Демонстрація методу Gray_landfill(self).

3) identifikation_landfill(self) - Цей метод викликається при натисканні кнопки «Ідентифікація» (identification_button). Він виконує розпізнавання об’єктів на зображеннях, які були вдосконалені та перетворені на градації сірого. Виявлені об'єкти представлені прямокутниками на вихідному зображенні.

Програмний код цього методу наведено на рисунку (2.11):

defidentification_l.andfitt(self):

```
def identification_landfill(self):
    if hasattr(self, 'input_image') and hasattr(self, 'img_enh') and hasattr(self, 'gray_img'):
        template = None
        if self.filepath == 'C:/Program System/load_images/sentinel-hub/sentinel-hub_02-05-2023.jpg':
            template = cv2.imread("data\\landfill.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/sentinel-hub/sentinel-hub_16-05-2022.jpg':
            template = cv2.imread("data\\landfill_3.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/maxar/maxar_28-07-2022.jpg':
            template = cv2.imread("data\\landfill_1.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/maxar/maxar_12-10-2022.jpg':
            template = cv2.imread("data\\landfill_2.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/google maps/landfill.jpg':
            template = cv2.imread("data\\landfill_4.jpg", cv2.IMREAD_GRAYSCALE)

        if template is not None:
            w, h = template.shape[::-1]
            result = cv2.matchTemplate(self.gray_img, template, cv2.TM_CCOEFF_NORMED)
            loction = np.where(result >= 0.8)
            for pt in zip(*loction[::-1]):
                cv2.rectangle(self.image, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 2)
            plt.figure(figsize=(17, 8))
            plt.imshow(cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB))
            plt.title('Object detection')
            plt.axis('off')
            plt.show()
            cv2.imwrite("results\\landfill\\identify_image_landfill.jpg", self.image)
        else:
            messagebox.showerror("Помилка!", "Неможливо знайти шаблон для даного зображення!")
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
```

Рисунок 2.11 – Демонстрація методу ідентифікації полігону.

‘

Методи обробки та розпізнавання зображень переглянуто, тепер переходимо до інших методів, які будуть використовуватися з вікном, яке буде порівнювати два зображення з різних сервісів.

4) compare_landfill_images(self) - викликається при натисканні кнопки «Порівняти зображення» (self.compare_landfill_button). Він створює нове вікно (self.compare_window), де можна порівняти два зображення.

Програмний код цього методу наведено на рисунку (2.12):

```
def compare_landfill_images(self):
    self.compare_window = ctk.CTkToplevel(self)
    self.compare_window.title("Порівняння знімків")
    self.compare_window.geometry(f"{1200}x{630}")
    self.compare_window.resizable(False, False)

    self.compare_window.down_compare = ctk.CTkLabel(self.compare_window, text="Завантажити знімки:", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.down_compare.place(x=30, y=30)
    self.compare_window.info_maxar = ctk.CTkLabel(self.compare_window, text="Знімок MAXAR", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.info_maxar.place(x=210, y=150)
    self.compare_window.info_sentinel_hub = ctk.CTkLabel(self.compare_window, text="Знімок Sentinel Hub", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.info_sentinel_hub.place(x=790, y=150)
    self.compare_window.add_sign = ctk.CTkImage(dark_image=Image.open("data\\sign.png"), size=(30, 30))
    self.compare_window.compare_landfill_button = ctk.CTkButton(self.compare_window, text='', command=self.load_compare_image,
                                                                compound=ctk.TOP, image=self.add_sign)
    self.compare_window.compare_landfill_button.place(x=50, y=70)

    self.compare_window.mainloop()
```

Рисунок 2.12 – Демонстрація методу Compare_landfill_images(self).

Рисунок (2.12) пояснює:

Цей метод створює та налаштовує вікно, необхідне для порівняння зображень MAXAR і Sentinel Hub.

5) load_compare_image(self) – цей метод викликається, коли у вікні порівняння зображень натискається кнопка «Завантажити зображення» (self.compare_landfill_button).

Програмний код цього методу наведено на рисунку (2.13):

```
def load_compare_image(self):
    self.filepath_compare_1 = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
    self.input_img_1 = ctk.CTkImage(dark_image=Image.open(self.filepath_compare_1), size=(500, 375))
    self.filepath_compare_2 = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
    self.input_img_2 = ctk.CTkImage(dark_image=Image.open(self.filepath_compare_2), size=(650, 375))
    input_label_1 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_1)
    input_label_1.place(x=20, y=200)
    input_label_2 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_2)
    input_label_2.place(x=540, y=200)
```

Рисунок 2.13 – Демонстрація методу load_compare_image(self).

‘

Останнім класом у цій програмній системі є клас Solar_Station_Window. Він також є підкласом ctk.CTkToplevel і відповідає за вікно дослідження сонячної станції.

Методи, що відповідають за виклик конструктора - __init__(self,parent), налаштування вікна - init_solar_station(), закриття вікна -

close_solar_station(self), download snapshot - download_file_station(self) схожі на попередній клас Landfill_Window, тому ми не будемо на них зосереджуватися, а краще продовжимо розгляд методів обробки знімків і виявлення в них об’єктів.

Методи обробки та розпізнавання зображень:

1) enhance_station(self) – цей метод працює так само, як і попередня категорія. Виконує функцію поліпшення картинки.

Програмний код цього методу наведено на рисунку (2.14):

```
def enhance_station(self):
    if hasattr(self, 'input_image'):
        img = Image.open(self.filepath)
        enc_img = img.filter(ImageFilter.DETAIL)
        img_con = ImageEnhance.Contrast(enc_img)
        self.image_enh = img_con.enhance(1.3)
        self.image_enh.save('results\\solar_station\\enhanced_image_station.jpg')
        self.image_enh.show()
    else:
        messagebox.showerror("Помилка!", "Спочатку завантажте знімок!")
```

Рисунок 2.14 – Демонстрація методу станції доповнення.

2)gray_station(self) - метод перетворення зображення на відтінок сірого. Він відрізняється від методу gray_landfill(self) тим, що він застосовує розмиття за Гаусом для зменшення шуму та згладжування деталей.

Програмний код цього методу наведено на рисунку (2.15):

```
def gray_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh'):
        img_path = 'results\\solar_station\\enhanced_image_station.jpg'
        self.img = cv2.imread(img_path)
        b, g, r = cv2.split(self.img)
        rgb_img = cv2.merge([r, g, b])
        self.bgr_to_gray = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
        self.gray_image = cv2.GaussianBlur(self.bgr_to_gray, (7, 7), 1)
        plt.figure(figsize=(15, 7.6))
        plt.imshow(self.gray_image, cmap='gray')
        plt.title('Gray Image')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\gray_image_station.jpg", self.gray_image)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Покращення якості'!")
```

Рисунок 2.15 – Демонстрація методу Gray_station(self).

3) edge_detect_station(self) - Визначає краї на зображеннях.

Програмний код цього методу наведено на рисунку (2.16):

```
def edge_detect_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image'):
        self.edged_image = cv2.Canny(self.gray_image, 150, 220)
        plt.figure(figsize=(15, 7.6))
        plt.imshow(self.edged_image, cmap='gray')
        plt.title('Edge detection Image')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\edge_detection_image_station.jpg", self.edged_image)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
```

Рисунок 2.16 – Демонстрація методу Edge_Detect_station(self).

4) morph_transformation_station(self) – Цей метод виконує морфологічну трансформацію зображення перед розпізнаванням.

Програмний код цього методу наведено на рисунку (2.17):

```
def morph_transformation_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image') and hasattr(self, 'edged_image'):
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
        self.morphology = cv2.morphologyEx(self.edged_image, cv2.MORPH_CLOSE, kernel, iterations=2)
        plt.figure(figsize=(15, 7.6))
        plt.imshow(self.morphology, cmap='gray')
        plt.title('Morphology transformation Image')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\morphology_transformation_image.jpg", self.morphology)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Виявлення країв'!")
```

Рисунок 2.17 – Демонстрація методу morph_transformation_station(self).

‘

5)identification_station(self) - останній метод класу Solar_Station_Window, відповідальний за ідентифікацію об'єктів шляхом малювання контурів з певними вимогами.

Програмний код цього методу наведено на рисунку (2.18):

```
def identification_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image') and \
            hasattr(self, 'edged_image') and hasattr(self, 'morphology'):
        contours = cv2.findContours(self.morphology.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
        for contour in contours:
            peri = cv2.arcLength(contour, True)
            approx = cv2.approxPolyDP(contour, 0.02 * peri, True)

            if len(approx) == 4 and cv2.contourArea(contour) > 2000:
                cv2.drawContours(self.img, [approx], -1, (0, 255, 0), 4)

        plt.figure(figsize=(15, 7.6))
        plt.imshow(cv2.cvtColor(self.img, cv2.COLOR_BGR2RGB))
        plt.title('Object detection')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\identification_image.jpg", self.img)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Морфологічне перетворення'!")
```

Рисунок 2.18 – Демонстрація методу identify_station(self).

Це останній метод класу Solar_Station_Window, який відповідає за виявлення об’єктів на зображенні шляхом пошуку та малювання контурів.

У цьому розділі підібрано методи та прийоми для ефективної реалізації процесу ідентифікації заданого об’єкта. При виборі враховуються вимоги проєкту, його обсяг і складність, а також потенційні переваги й обмеження кожного підходу й технології.

Одним із варіантів є використання сервісів MAXAR і Sentinel Hub для доступу та завантаження оперативних зображень, отриманих за допомогою дистанційного зондування Землі. Ці послуги пропонують широкий спектр можливостей, включаючи можливість отримувати найновіші зображення з різних супутників і джерел даних. Крім того, використовувалися еталонні зображення з Google Maps, які надали високоякісні зображення для порівняння.

Для виявлення об'єктів обрані ефективні методи, такі як методи, засновані на розпізнаванні контурів і пошуку контурів для виявлення сонячних систем на зображеннях. Крім того, метод виявлення об'єктів за шаблонами також

використовується для виявлення місць звалищ. Цей метод дозволяє порівняти візерунок об’єкта з різними частинами зображення, дозволяючи точно визначити його присутність.

Також створюються діаграми класів, які описують класи, що відповідають за функціональність і конфігурацію інтерфейсу програмної системи. Показано основні методи в класах Main, Landfill_window та Solar_Station_Window, які допоможуть краще зрозуміти алгоритми дій та операцій, розроблені під час виконання завдання.

‘

3 ТЕСТУВАННЯ ТА АНАЛІЗ СТВОРЕНОЇ СИСТЕМИ ВИЯВЛЕННЯ КОНКРЕТНИХ ОБ'ЄКТІВ ЗА ДАНИМИ ВІДКРИТИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

У цьому розділі розглядається інтерфейс, практичний функціонал і документація розробленої програмної системи.

3.1 Демонстрація практичної функції розробленої програмної системи.

Основна ідея при створенні інтерфейсу — зробити його максимально простим і зрозумілим, забезпечуючи при цьому естетично привабливий дизайн. Тому користувачі зможуть швидко зорієнтуватися та використовувати його без особливих зусиль.

Під час запуску програми з’явиться головне вікно програми (рисунок (3.1)), з якого можна розпочати дослідження.

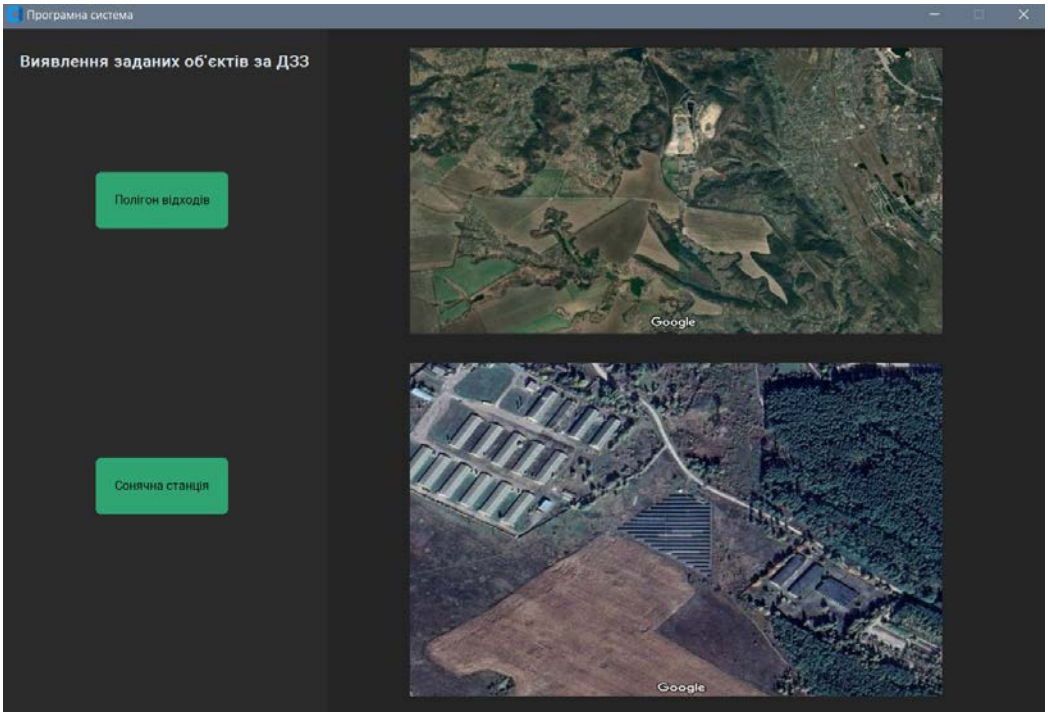


Рисунок 3.1 – Ілюстрація головного вікна програми.

У головному вікні можна вибрати об’єкт дослідження, натиснувши на кнопку «Полігон», щоб відкрити нове вікно або «Сонячна станція». Крім того, навпроти кожної кнопки знаходяться самі об’єкти, як ми побачимо в майбутньому.

‘

Спочатку ми натискаємо «Полігон», і ми побачимо нове вікно, що відкриється лише з двома кнопками - «Завантажити зображення» (у вигляді зеленої стрілки) і «Порівняти зображення» (Рис. (3.2)).

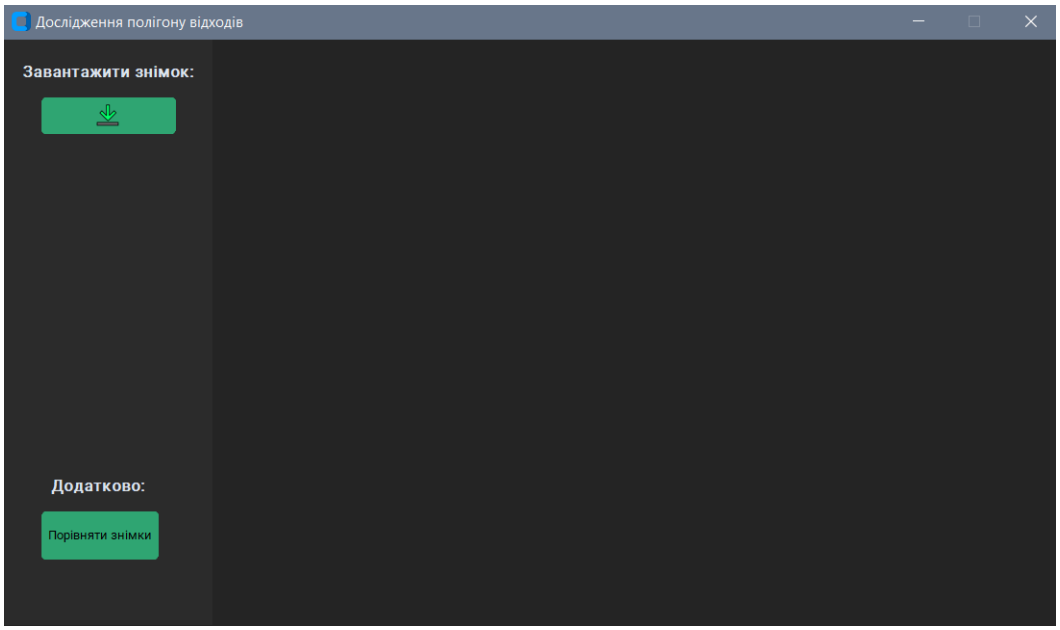


Рисунок 3.2 – Ілюстрація вікна при натисканні на «Полігон».

Ви можете натиснути кнопку «Порівняти зображення» зараз, що відкриє нове вікно для порівняння зображень із супутників MAXAR і Sentinel Hub, або після завантаження зображень, у цьому випадку різниці немає. Але припустимо, що користувач негайно хоче порівняти картинки, тоді, коли він натисне кнопку «Порівняти картинки», відкриється інше вікно, у якому вже є кнопка для завантаження картинок і залишає порожнє місце для вибраного зображення (Рисунок (3.3)).

‘

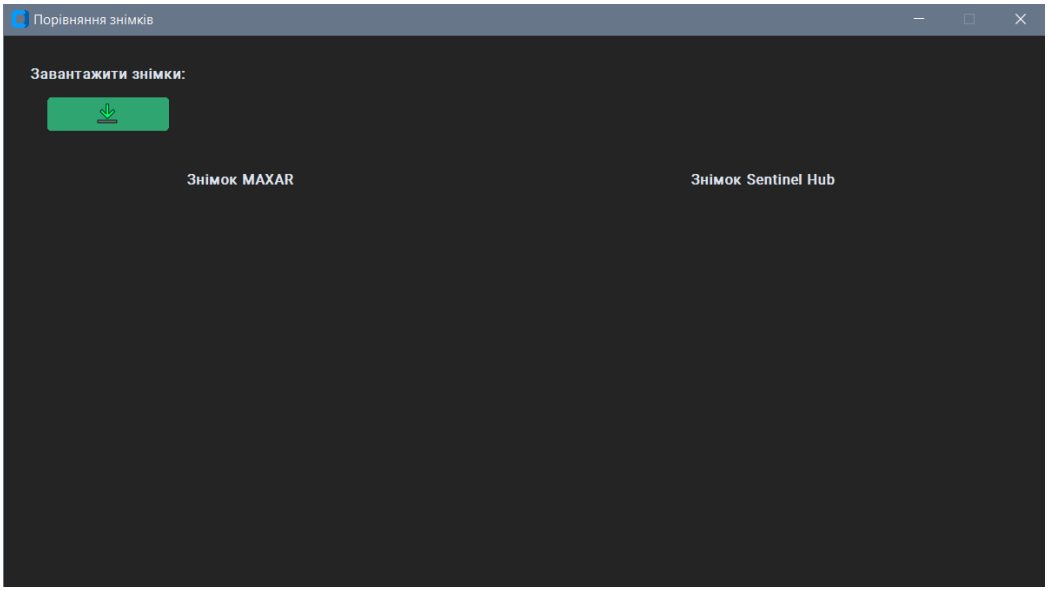


Рисунок 3.3 – Ілюстрація вікна після натискання «Порівняти знімки».

Вам потрібно лише один раз натиснути кнопку. На екрані негайно з’явиться діалогове вікно файлу, і вам потрібно буде вибрати знімок спочатку для MAXAR, а потім для Sentinel Hub (Рис. (3.4) і Рис. (3.5)).

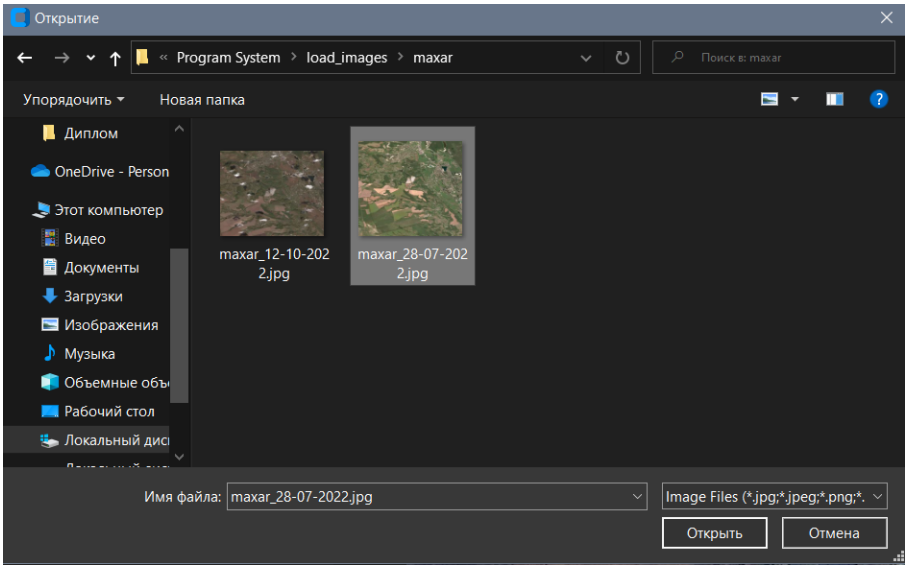


Рисунок 3.4 – Завантаження знімку з MAXAR.

‘

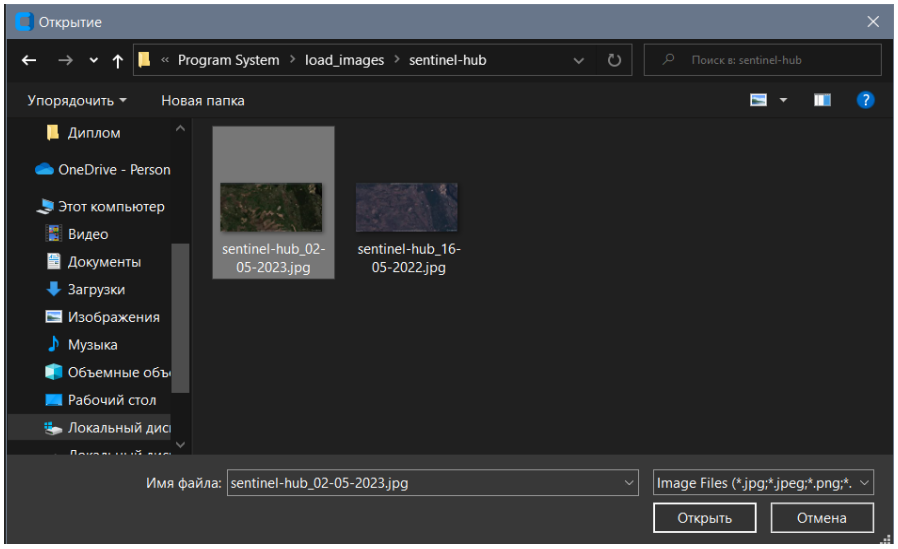


Рисунок 3.5. Завантаження знімка з Sentinel Hub.

Коли друге зображення буде завантажено, воно відобразиться у вікні, як ви бачите на зображенні (3.6), раніше було порожнє місце.

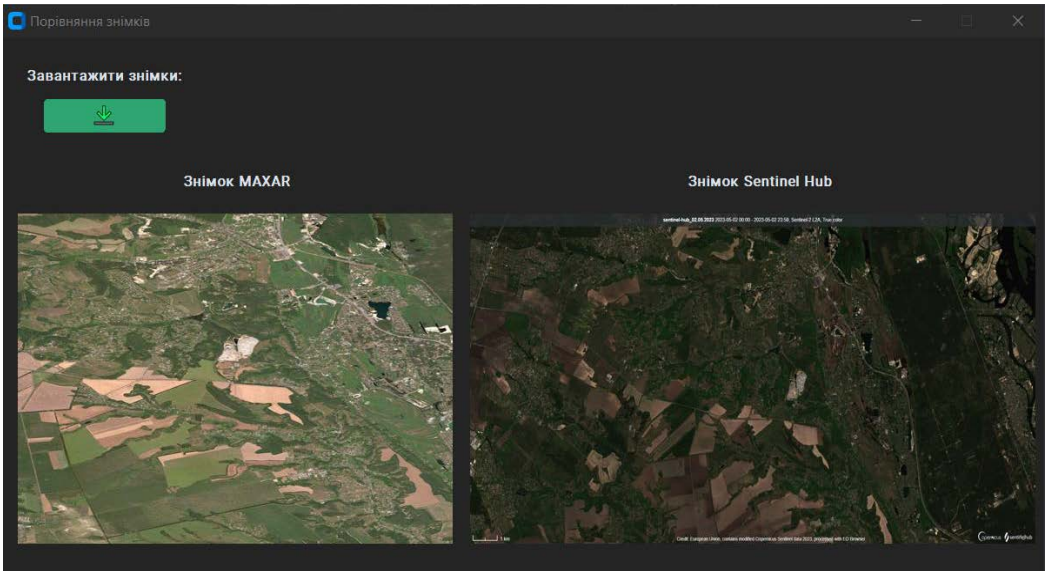


Рисунок 3.6 – Експорт знімка у вікно для порівняння.

Після порівняння зображень закрийте вікно та поверніться до вікна полігону, як показано на рисунку (3.2). Туди потрібно завантажити зображення, інакше повний функціонал обробки зображень і виявлення об’єктів на них буде недоступний. Отже, ми дотримуємося того самого принципу, щоб завантажити зображення з діалогового вікна файлу, показаного на рисунку (3.4) або (3.5), залежно від того, яка служба вам потрібна для перевірки зображення.

‘

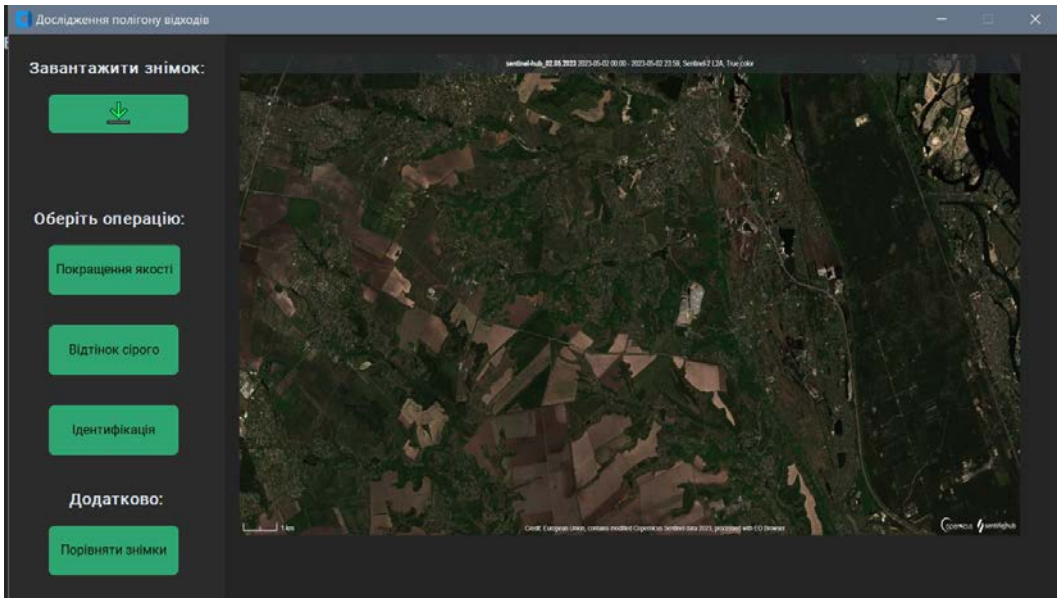


Рисунок 3.7 – Вікно полігону після завантаження зображення

На рисунку (3.7) ми бачимо, що після успішного завантаження зображення можна виконати наступні операції: «Покращення якості зображення», «Відтінки сірого» та «Розпізнавання». Варто зауважити, що застосування операцій повинно здійснюватися суворо за планом, наведеним на рисунку (3.7). Якщо ви спробуєте «пропустити» цю операцію, на екрані з’явиться повідомлення про те, що спочатку потрібно виконати попередню обробку.

Почнемо з першої кнопки, яка покращить якість картинки (рис. (3.8)).

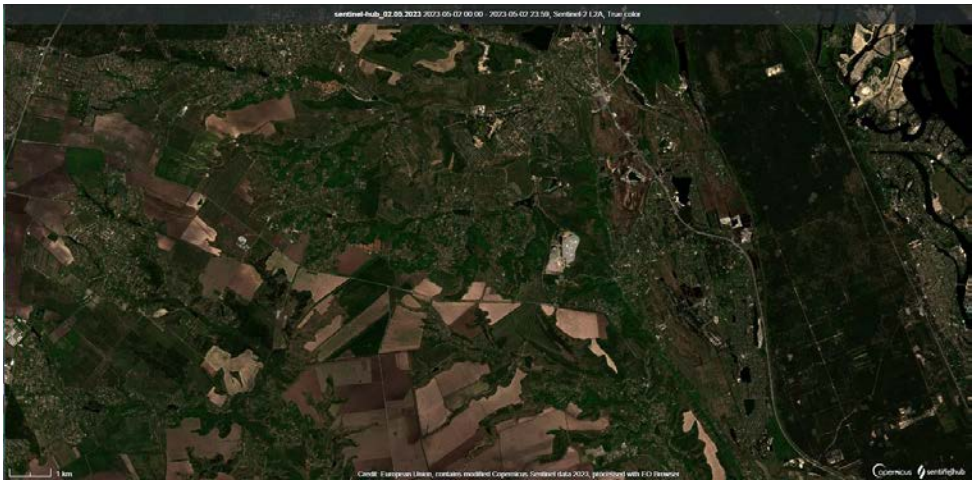


Рисунок 3.8 – Покращення якості після натискання першої дії.

‘

Щоб виконати наступний крок, закрийте зображення, яке з’явиться, і натисніть «Відтінки сірого».

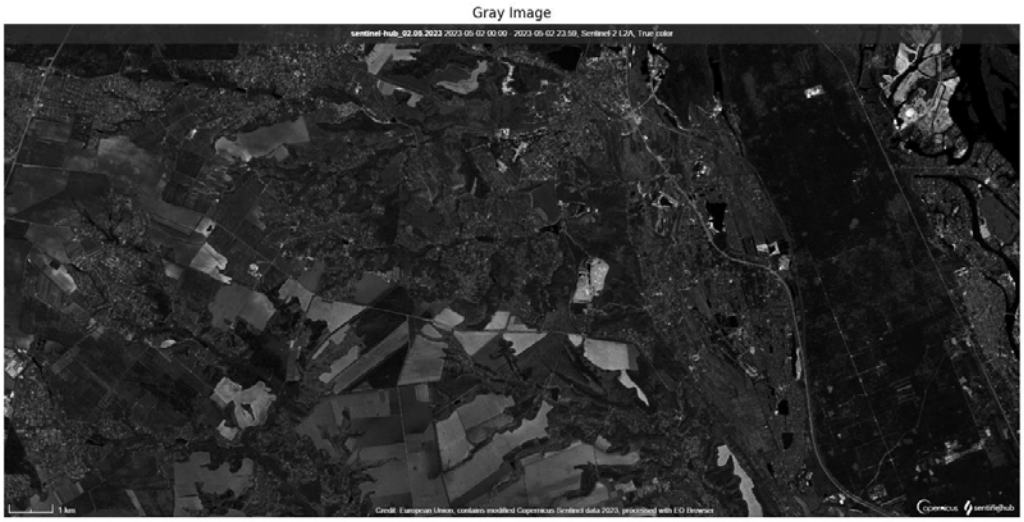


Рисунок 3.9 – Відтінки сірого після натискання другої дії.

Після цього, як показано на рисунку (3.9), можна буде виявити об’єкти на зображенні.

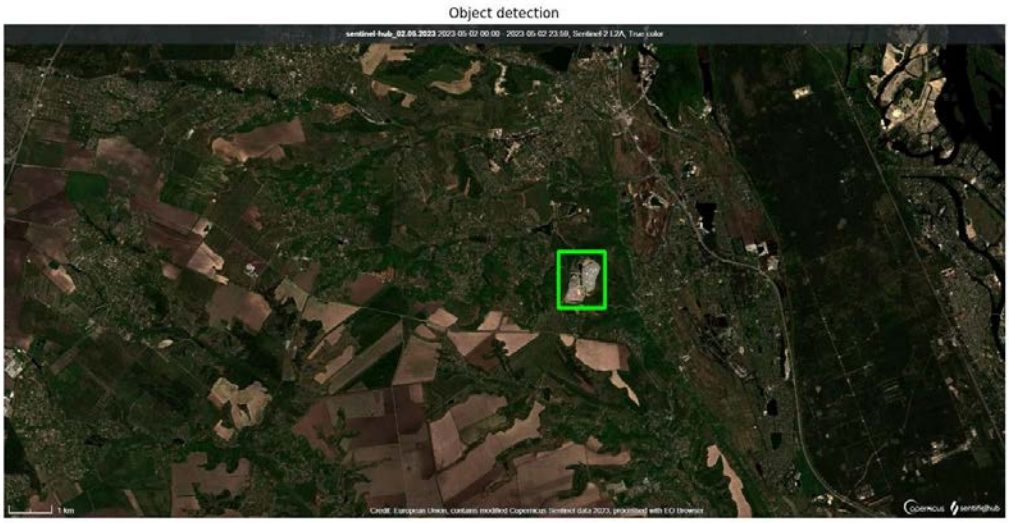


Рисунок 3.10 – Розпізнавання об’єкта після натискання третьої дії.

Якщо все виконувати послідовно, то з’явиться зображення з очікуваним результатом, як показано на рисунку (3.10). Місце виявлення буде оточене зеленим прямокутником.

‘

Далі закриваємо вікно дослідження полігону та повертаємось до головного вікна – рисунок (3.1).

Обираємо наступну дослідницьку локацію – «Сонячну станцію».

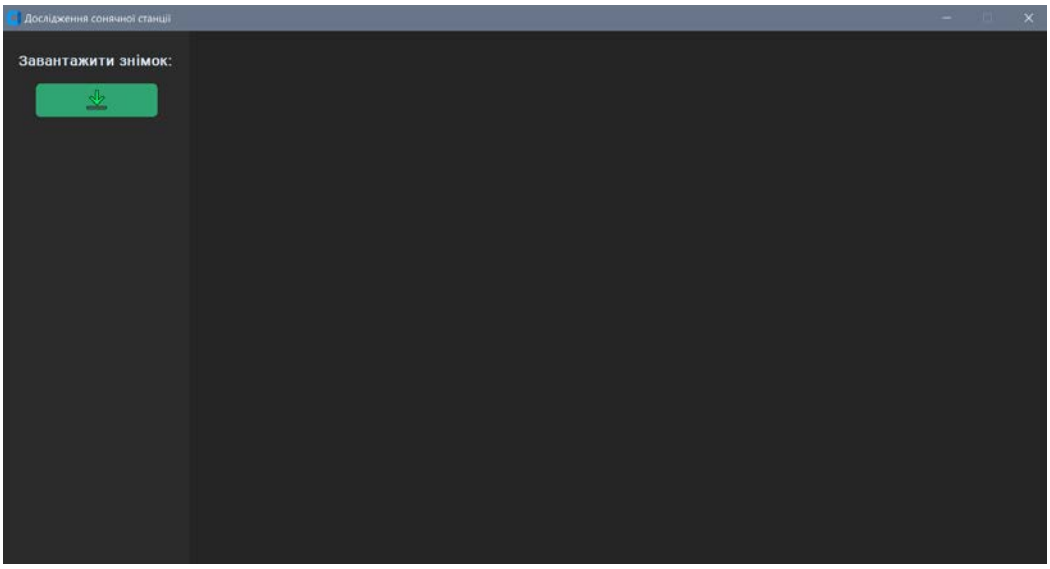


Рисунок 3.11 – Вікно вивчення сонячної станції.

Так само, як і для обстеження полігону, вам потрібно завантажити зображення, перш ніж ви зможете використовувати цю функцію, як показано на рисунку (3.12).

Рисунок 3.12 - Вікно сонячної станції після завантаження зображення

Вибираємо по черзі кожну операцію обробки зображення та отримуємо наступні

‘

результати обробки, як показано на рисунку (3.13-3.16):

Рисунок 3.13 – Операція покращення якості.

Gray image

Рисунок 3.14 – Операція відтінку сірого.

Edge detection Image

‘

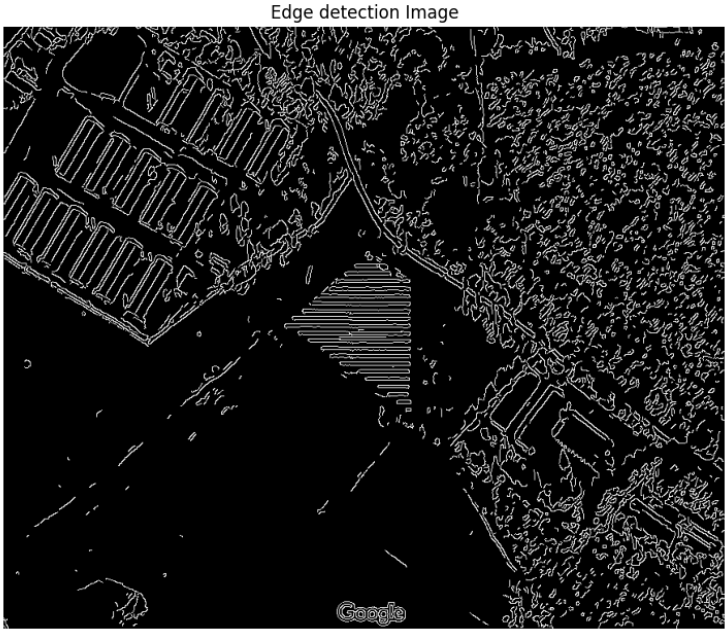


Рисунок 3.15 – Операція виявлення країв.

Morphology transformation Image

Рисунок 3.16 – Операція перетворення форми.

Після успішної обробки зображення ми отримуємо очікуваний результат, тобто виявлення цілі на досліджуваному зображенні, як показано на рисунку (3.17).

Object detection

‘

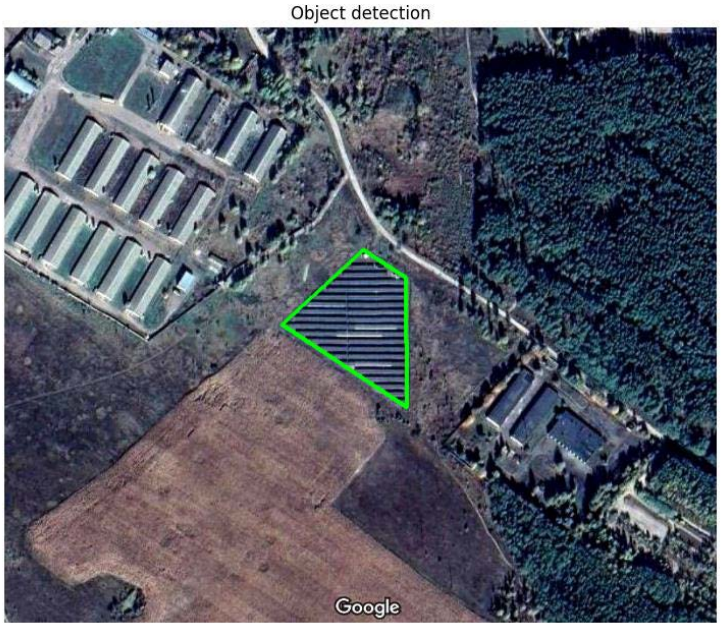


Рисунок 3.17 – Операція розпізнавання об’єкта.

У процесі ознайомлення з інтерфейсом і доступними функціями програмної системи ви матимете можливість на практиці працювати з об’єктами на картинках і виявляти їх. Достовірність розробленої статті було доведено за допомогою скріншотів програмної системи, і мета була досягнута.

3.2 Програмна документація до програмного забезпечення.

У цьому розділі міститься опис програмної документації, основних методів класу та залежностей.

Щоб почати роботу з цим проєктом, вам потрібно написати серію дій, які допоможуть користувачам на комп’ютері успішно запустити програмну систему.

1. Увійдіть до репозиторію GitHub за посиланням: https://github.com/Program-System

2. Завантажте всі файли у форматі ZIP зі сховища, як показано на рисунку (3.18).

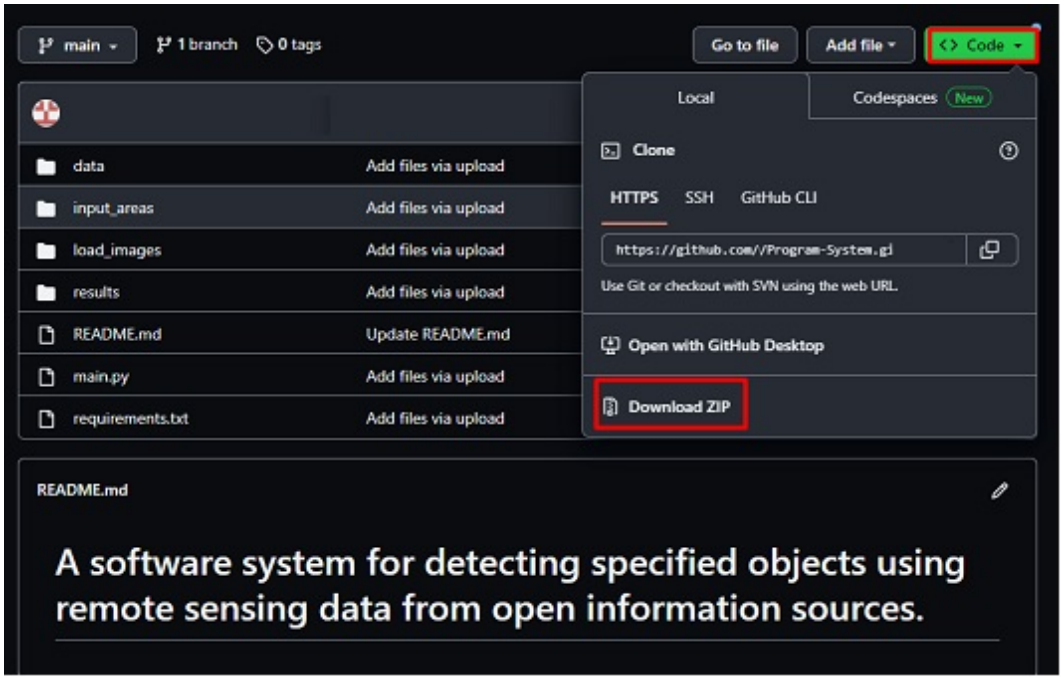


Рисунок 3.18 – Завантаження файлів проєкту з GitHub.

3. Створіть папку «Program System» на диску «C» і перенесіть усі файли з архіву в новостворену папку «Program System».

4. Після цього заходимо в будь-яку IDE, де будемо налаштовувати віртуальне середовище проєкту.

5. Увійдіть у термінал і за допомогою команди python -m venv venv створіть віртуальне середовище (у проєкті буде створено папку з назвою venv).

6. Якщо у вас операційна система Windows і джерело Linux venv\bin\activate, скористайтеся командою venv\Scripts\active, щоб активувати віртуальне середовище.

7. Використовуйте команду pip install -rrequirements.txt для встановлення залежностей проєкту.

*requirements.txt — це файл, у якому зберігаються всі пакети проєкту.

8. Після завершення всіх налаштувань ми можемо запустити виконуваний файл main.py.

Далі, програмним системам потрібні описи залежностей і класів, щоб краще зрозуміти роботу.

Ця програма має такі залежності:

‘

Таблиця 3.1 – Опис бібліотек, залучених у програмі.

Залежність	Опис
numpy	(NumPy) - Бібліотека для числових обчислень.
cv2	(OpenCV) - Бібліотека для комп'ютерного зору та обробки зображень.
tkinter	Стандартна бібліотека для розробки GUI.
customtkinter	Бібліотека на основі Tkinter, що надає додатковий функціонал для GUI.
PIL	(Pillow) - Бібліотека для обробки та маніпулювання зображеннями.)
matplotlib.pyplot	Бібліотека для візуалізації даних.

Клас Main представляє головне вікно програмної системи. Він успадковує клас ctk.CTk, який забезпечує базову функціональність для створення ваших власних програм Tkinter. Нижче наведено основний метод класу Main.

Таблиця 3.2 – Характеристика методів класа Main.

Метод	Опис
init_main()	Ініціалізує головне вікно програмної системи. Налаштовує елементи GUI.
close_main()	Відображає діалогове вікно підтвердження про зак риття та закриває програму, якщо користувач підтверджує.
landfill_window()	Відкриває нове вікно для функціональності, пов'язаної з полігоном відходів.
solar_station_window()	Відкриває нове вікно для функціональності, пов'язаної з сонячною станцією.

Клас Landfill_Window представляє вікно для функцій, пов’язаних зі звалищем. Це дочірнє вікно головного вікна програми.

Таблиця 3.3 - Характеристика методів класа Landfill_Window.

‘

Метод	Опис
init_landfill()	Ініціалізує вікно полігону відходів програмної системи. Налаштовує елементи GUI.
close_landfill()	Відображає діалогове вікно підтвердження про зак риття та закриває програму, якщо користувач підтверджує.
download_file_landfill()	Відкриває діалогове вікно для вибору файлу зображення. Після вибору файлу, відображає зображення на вікні та створює додаткові кнопки для різних операцій.
enhance_landfill()	Застосовує покращення якості до завантаженого зображення. Відкриває нове вікно з покращеним зображенням.
identification_landfill()	Виконує ідентифікацію об'єктів на зображенні полігону відходів. Він використовує певні шаблони зображень для порівняння з вхідним зображенням і позначає знайдений об'єкт на знімку.
compare_landfill_images()	Відкриває нове вікно для порівняння двохзображень.
load_compare_image()	Відкриває діалогове вікно для вибору двох зображень для порівняння.

Клас Solar_Station_Window представляє вікно для функцій, пов’язаних зі звалищем. Це дочірнє вікно головного вікна програми.

Таблиця 3.4 – Характеристика методів класа Solar_Station_Window.

‘

Метод	Опис
init_solar_station()	Ініціалізує вікно сонячної станції програмної систе ми. Налаштовує елементи GUI.
close_solar_station()	Відображає діалогове вікно підтвердження про закриття та закриває програму, якщо користувач підтверджує.
download_file_station()	Відкриває діалогове вікно для вибору файлу зображення. Після вибору файлу, відображає знімок на вікні та створює додаткові кнопки для різних операцій.
enhance_station()	Застосовує покращення якості до завантаженого знімку. Відкриває нове вікно з покращеним зображенням.
gray_station()	Перетворює знімок в відтінок сірого та відображає його на вікні.
edge_detect_station()	Виконує виявлення країв на знімку сонячної станції. Відображає знімок з виявленими краями на вікні.
identification_station()	Виконує ідентифікацію об'єкта на знімку сонячної станції. Відображає знімок з позначеним ідентифікованим об'єктом на вікні.

У цій частині дипломної роботи досягнуто важливих результатів у розробці програмного забезпечення для виявлення заданих об’єктів за допомогою дистанційного зондування Землі. Ефективність програми підтверджена практичним впровадженням системи та наданими скріншотами, що демонструють успішні результати.

Одним із головних досягнень є створення програмної системи, яка може точно виявляти об’єкти за допомогою аналізу даних дистанційного зондування Землі. Це включає вибір найкращих методів обробки зображень і методів виявлення об’єктів на основі конкретних характеристик.

Крім того, приділяється увага розробці документації, що дуже важливо для

‘

майбутніх користувачів програмної системи. Цей документ містить детальний опис, конфігурацію та інструкції з використання функціональності системи.

Загальна мета цього розділу — продемонструвати, що програмне забезпечення ефективно виконує свої функції та має практичну цінність. Результати, отримані під час впровадження програмної системи, підтверджують успішність проєкту та його придатність у реальних ситуаціях.

ВИСНОВКИ

У дипломній роботі було розроблено програмну систему, яка може виявляти задані об’єкти, а саме сміттєзвалища та сонячні електростанції.

Аналізуючи відомі технічні рішення та способи ідентифікації об’єктів у задачах комп’ютерного зору, було обрано певні засоби, які дозволять досягти поставленої мети. Також описані методи обробки зображень і способи полегшення виявлення об'єктів.

В рамках дослідження детально вивчена проблема ідентифікації даного об'єкта та обробки ДЗЗ Землі. Методи впровадження програмної системи були обрані з урахуванням вимог і функціональності кожного методу в попередньому аналізі.

В процесі реалізації була проведена обробка зображення, а саме: покращення якості, відтінки сірого, виділення країв об’єкта, морфологічна трансформація. Використовуються два способи виявлення заданого об’єкта на зображенні, один – метод пошуку та знаходження об’єкта на зображенні за шаблоном (збіг шаблону) – використовується для виявлення звалищ ТПВ, інший – метод пошуку та малювання. Контурний метод - для виявлення сонячних станцій. Отримані результати демонструють практичні можливості розробленої системи у виявленні об’єктів на DZD-зображеннях. Крім того, створено документацію, яка містить інструкції для майбутніх користувачів, які допоможуть їм максимально просто та зрозуміло вивчити та використовувати систему.

Загалом результати цього проєкту свідчать про те, що ДЗЗ за допомогою відкритих джерел інформації успішно розробила програмну систему виявлення зазначених об’єктів, яка має сучасний та зрозумілий інтерфейс та надає користувачам широкі практичні можливості.

ПЕРЕЛІК ПОСИЛАНЬ

1. J. R. Jensen, "Introductory Digital Image Processing: A Remote Sensing Perspective," 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2015.

2. P. M. Mather and M. Koch, "Computer Processing of Remotely-Sensed Images: An Introduction," 4th ed. Chichester, UK: Wiley-Blackwell, 2011.

3. R. C. Gonzalez and R. E. Woods, "Digital Image Processing," 4th ed. New York, NY, USA: Pearson, 2018.

4. T. Blaschke, S. Lang, and G. J. Hay, "Object-Based Image Analysis: Spatial Concepts for Knowledge-Driven Remote Sensing Applications," Berlin, Germany: Springer, 2008.

5. S. Prasad, L. M. Bruce, and J. Chanussot, "Optical Remote Sensing: Advances in Signal Processing and Exploitation Techniques," New York, NY, USA: Springer, 2011.

6. D. Lu and Q. Weng, "A Survey of Image Classification Methods and Techniques for Improving Classification Performance," International Journal of Remote Sensing, vol. 28, no. 5, pp. 823-870, Mar. 2007.

7. M. Ehlers, "Multisensor Image Fusion Techniques in Remote Sensing," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 46, no. 1, pp. 19-30, Aug. 2001.

8. X. Jia, B. Zhang, and C. K. Chan, "Spectral-Spatial Classification for Hyperspectral Remote Sensing Images with Spatial Pixel Pair Features," IEEE Transactions on Geoscience and Remote Sensing, vol. 50, no. 10, pp. 4198-4211, Oct. 2012.

9. L. Bruzzone and D. F. Prieto, "A Neural-Statistical Approach to Multitemporal and Multisource Remote-Sensing Image Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 37, no. 3, pp. 1350-1359, May 1999.

10. J. A. Benediktsson, J. Chanussot, and M. Fauvel, "Multiple Classifier Systems in Remote Sensing: From Basics to Recent Developments," in Multiple Classifier Systems, MCS 2007, F. Roli, J. Kittler, and T. Windeatt, Eds. Berlin, Germany: Springer, 2007, pp. 501-512.

11. D. Tuia, F. Ratle, F. Pacifici, M. Kanevski, and W. Emery, "Active Learning Methods for Remote Sensing Image Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 47, no. 7, pp. 2218-2232, Jul. 2009.

12. P. Ghamisi, M. Dalla Mura, and J. A. Benediktsson, "A Survey on Spectral-Spatial

‘

Classification Techniques Based on Attribute Profiles," IEEE Transactions on Geoscience and Remote Sensing, vol. 53, no. 5, pp. 2335-2353, May 2015.

13. J. Li, P. Zhang, X. Zhang, and R. Hu, "An Integrated Framework of Remote Sensing Image Fusion for Urban Object Detection," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 6, pp. 2393-2404, Jun. 2014.

14. L. Zhang, L. Zhang, and B. Du, "Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art," IEEE Geoscience and Remote Sensing Magazine, vol. 4, no. 2, pp. 22-40, Jun. 2016.

15. Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Segmentation and Classification of Hyperspectral Images Using Watershed Transformation," Pattern Recognition, vol. 43, no. 7, pp. 2367-2379, Jul. 2010.

16. W. L. Chiang, W. H. Hsu, and Y. P. Hung, "An Incremental Learning Method for Remote Sensing Image Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 41, no. 4, pp. 1075-1081, Apr. 2003.

17. C. Chen, L. Wang, and J. Wang, "A Survey on Machine Learning for Data Fusion in Remote Sensing," International Journal of Image and Data Fusion, vol. 7, no. 1, pp. 1-31, Jan. 2016.

18. S. Li, X. Kang, L. Fang, J. Hu, and H. Yin, "Pixel-Level Image Fusion: A Survey of the State of the Art," Information Fusion, vol. 33, pp. 100-112, Jan. 2017.

‘

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: « Автоматизована система дистанційного зондування земної поверхні з можливістю класифікації об'єктів»

Тип роботи: Бакалаврська дипломна робота
(БДР, МКР)

Підрозділ КСУ, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99,4% Схожість 0,6%

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Дмитро ПАРХОМЕНКО
(підпис) (прізвище, ініціали)

Керівник роботи _____ Олена НИКИТЕНКО
(підпис) (прізвище, ініціали)

Додаток Б

(обов'язковий)

ЗАТВЕРДЖЕНО

Зав. кафедри КСУ ВНТУ,

д.т.н., проф.

_____

" 25 " 09 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

Інформаційна система дистанційного тестування знань студентів з можливістю класифікації об'єктів

Студент групи 1МОТ-23м

_____

Керівник: доцент кафедри КСУ

_____

2024

1. Назва та галузь застосування

1.1. Назва – Автоматизована система дистанційного зондування земної поверхні з можливістю класифікації об’єктів

1.2. Галузь застосування – автоматизовані системи класифікації та розпізнавання об’єктів.

2. Підстава для проведення розробки.

Тема бакалаврської дипломної роботи затверджена наказом по ВНТУ № 80 від 11 березня.2024 р.

3. Мета та призначення розробки.

Метою даного дослідження є підвищення ефективності та надійності ідентифікації даного об’єкта за даними дистанційного зондування землі на основі відкритих джерел інформації..

4. Джерела розробки.

Бакалаврська дипломна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. J. R. Jensen, "Introductory Digital Image Processing: A Remote Sensing Perspective," 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2015.

2. P. M. Mather and M. Koch, "Computer Processing of Remotely-Sensed Images: An Introduction," 4th ed. Chichester, UK: Wiley-Blackwell, 2011.

3. R. C. Gonzalez and R. E. Woods, "Digital Image Processing," 4th ed. New York, NY, USA: Pearson, 2018..

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- Передача даних із до хмарних сервісів
- Аналіз даних та розпізнавання об’єктів
- Інтеграція з хмарними сервісами
- Безпечне зберігання та управління даними

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10,Linux;

5.2.2. Умови експлуатації системи:

- можливість цілодобового функціонування системи;

‘

- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми дипломної роботи. Постановка задач дослідження «04» травня 204 р.

2. Визначення технічних характеристик системи «15» травня 2024р.

3. Розробка програмного забезпечення системи «20» травня 2024 р.

6.2 Графічні матеріали:

1. Розробка моделі системи «24» травня 2024 р.

2. Тестування програмного забезпечення «26» травня 2024р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «10» червня 2024 р.

7.2. Атестація проєкту здійснюється на попередньому захисті. Попередній захист бакалаврської дипломної роботи провести до «10» червня 2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист бакалаврської дипломної роботи провести до «20»червня 2024 р.

Додаток В

Лістинг

Файл: main.py

```
import numpy as np
import cv2
import customtkinter as ctk
from tkinter import messagebox, filedialog
from PIL import Image, ImageEnhance, ImageFilter
import matplotlib.pyplot as plt
ctk.set_appearance_mode("Dark")
ctk.set_default_color_theme("green")
class Main(ctk.CTk):
def __init__(self):
super().__init__()
self.init_main()
def init_main(self):
self.title("Програмна система") self.geometry(f"{1100}x{730}") self.resizable(False, False) self.protocol("WM_DELETE_WINDOW", self.close_main)
self.grid_columnconfigure(1, weight=1) self.grid_columnconfigure((2, 3), weight=0) self.grid_rowconfigure((0, 1, 2), weight=1)
self.sidebar_frame = ctk.CTkFrame(self, width=140, corner_radius=0)
self.sidebar_frame.grid(row=0, column=0, rowspan=4, sticky="nsew")
self.sidebar_frame.grid_rowconfigure(4, weight=1)
self.logo_label = ctk.CTkLabel(self.sidebar_frame,
text="Виявлення заданих об'єктів за ДЗЗ", font=ctk.CTkFont(size=17, weight="bold"))
self.logo_label.grid(row=0, column=0, padx=20, pady=(20, 10))
self.sidebar_button_1 = ctk.CTkButton(self.sidebar_frame, text='Полігон відходів', command=self.landfill_window, width=140, height=60, text_color='black')
self.sidebar_button_1.place(x=100, y=150)
self.sidebar_button_2 = ctk.CTkButton(self.sidebar_frame, text='Сонячна станція', command=self.solar_station_window, width=140, height=60, text_color='black')
self.sidebar_button_2.place(x=100, y=460)
self.landfill_image =
ctk.CTkImage(dark_image=Image.open("C:\\Program
System\\input_areas\\landfill.jpg"), size=(560, 300))
self.landfill_label = ctk.CTkLabel(self, text='', image=self.landfill_image)
```

‘

```
self.landfill_label.place(x=430, y=20)
self.solar_station_image =
ctk.CTkImage(dark_image=Image.open("C:\\Program
System\\input_areas\\solar_station.jpg"), size=(560, 350)
self.solar_station_label = ctk.CTkLabel(self, text='', image=self.solar_station_image)
self.solar_station_label.place(x=430, y=350)
def close_main(self):
if messagebox.askokcancel("Вихід із системи", "Впевнені, що хочете вийти з системи?"):
self.destroy()
def landfill_window(self):
landfill = Landfill_Window(self) landfill.grab_set()
def solar_station_window(self):
solar_station = Solar_Station_Window(self) solar_station.grab_set()
class Landfill_Window(ctk.CTkToplevel):
def __init__(self, parent):
super().__init__(parent)
self.init_landfill()
def init_landfill(self):
self.title("Дослідження полігону відходів") self.geometry(f"{1050}x{590}")
self.resizable(False, False)
self.protocol("WM_DELETE_WINDOW", self.close_landfill)
self.grid_columnconfigure(1, weight=1) self.grid_columnconfigure((2, 3), weight=0)
self.grid_rowconfigure((0, 1, 2), weight=1)
self.sidebar_frame = ctk.CTkFrame(self, width=140,
corner_radius=0)
self.sidebar_frame.grid(row=0, column=0, rowspan=4,
sticky="nsew")
self.sidebar_frame.grid_rowconfigure(4, weight=1)
self.down_label = ctk.CTkLabel(self.sidebar_frame,
text="Завантажити знімок:", font=ctk.CTkFont(size=16, weight="bold"))
self.down_label.grid(row=0, column=0, padx=20, pady=(20,
10))
self.add_sign =
ctk.CTkImage(dark_image=Image.open("C:\\Program
System\\data\\sign.png"), size=(30, 30))
self.load_landfill_button = ctk.CTkButton(self.sidebar_frame, text='', command=self.download_file_landfill,
compound=ctk.TOP, image=self.add_sign, text_color='black')
self.load_landfill_button.place(x=40, y=60)
self.compare_label = ctk.CTkLabel(self.sidebar_frame, text="Додатково:", font=ctk.CTkFont(size=16,
weight="bold"))
```

‘

```
self.compare_label.place(x=60, y=450) self.compare_landfill_button =
ctk.CTkButton(self.sidebar_frame, text='Порівняти знімки', command=self.compare_landfill_images,
width=130, height=50, text_color='black')
self.compare_landfill_button.place(x=40, y=490)
def close_landfill(self):
if messagebox.askokcancel("Закрити вікно", "Впевнені, що хочете закрити вікно?"):
self.destroy()
def download_file_landfill(self):
self.filepath =
filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
if self.filepath:
self.input_image =
ctk.CTkImage(dark_image=Image.open(self.filepath), size=(780, 480))
input_label = ctk.CTkLabel(self, text='',
image=self.input_image)
input_label.place(x=230, y=20)
operation_label = ctk.CTkLabel(self.sidebar_frame,
text='Оберіть операцію:', font=ctk.CTkFont(size=16, weight="bold")) operation_label.place(x=25, y=170)
enhance_button = ctk.CTkButton(self.sidebar_frame, text='Покращення якості',
command=self.enhance_landfill, width=70, height=50, text_color='black')
enhance_button.place(x=40, y=210)
gray_button = ctk.CTkButton(self.sidebar_frame, text='Відтінок сірого',
command=self.gray_landfill, width=130, height=50, text_color='black')
gray_button.place(x=40, y=290)
identification_button =
ctk.CTkButton(self.sidebar_frame, text='Ідентифікація',
command=self.identification_landfill, width=130, height=50, text_color='black')
identification_button.place(x=40, y=370)
else:
messagebox.showerror("Помилка!", "Оберіть знімок!")
def enhance_landfill(self):
if hasattr(self, 'input_image'):
img = Image.open(self.filepath)
enc_img = img.filter(ImageFilter.DETAIL)
img_con = ImageEnhance.Contrast(enc_img)
self.img_enh = img_con.enhance(1.3)
self.img_enh.save('C:\\Program
System\\results\\landfill\\enhanced_image.jpg')
self.img_enh.show()
else:
```

‘

```
messagebox.showerror("Помилка!", "Спочатку завантажте знімок!")
def gray_landfill(self):
if hasattr(self, 'input_image') and hasattr(self, 'img_enh'):
img = 'C:\\Program
System\\results\\landfill\\enhanced_image.jpg'
self.image = cv2.imread(img)
self.gray_img = cv2.cvtColor(self.image,
cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(17, 8))
plt.imshow(self.gray_img, cmap='gray')
plt.title('Gray Image')
plt.axis('off')
plt.show()
cv2.imwrite("C:\\Program
System\\results\\landfill\\gray_image_landfill.jpg", self.gray_img) else:
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Покращення якості'!")
def identification_landfill(self):
if hasattr(self, 'input_image') and hasattr(self, 'img_enh') and hasattr(self, 'gray_img'):
template = None
if self.filepath == 'C:/Program
System/load_images/sentinel-hub/sentinel-hub_02-05- 2024.jpg':
template = cv2.imread("C:\\Program
System\\data\\landfill.jpg", cv2.IMREAD_GRAYSCALE)
elif self.filepath == 'C:/Program
System/load_images/sentinel-hub/sentinel-hub_16-05-2022.jpg':
template = cv2.imread("C:\\Program
System\\data\\landfill_3.jpg", cv2.IMREAD_GRAYSCALE)
elif self.filepath == 'C:/Program
System/load_images/maxar/maxar_28-07-2022.jpg':
template = cv2.imread("C:\\Program
System\\data\\landfill_1.jpg", cv2.IMREAD_GRAYSCALE)
elif self.filepath == 'C:/Program
System/load_images/maxar/maxar_12-10-2022.jpg':
template = cv2.imread("C:\\Program
System\\data\\landfill_2.jpg", cv2.IMREAD_GRAYSCALE)
elif self.filepath == 'C:/Program
System/load_images/google maps/landfill.jpg':
template = cv2.imread("C:\\Program
System\\data\\landfill_4.jpg", cv2.IMREAD_GRAYSCALE)
if template is not None:
```

‘

```
w, h = template.shape[::-1]
result = cv2.matchTemplate(self.gray_img, template, cv2.TM_CCOEFF_NORMED)
loction = np.where(result >= 0.8)
for pt in zip(*loction[::-1]):
cv2.rectangle(self.image, pt, (pt[0] + w, pt[1]
+ h), (0, 255, 0), 2)
plt.figure(figsize=(17, 8))
plt.imshow(cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB))
plt.title('Object detection') plt.axis('off') plt.show()
cv2.imwrite("C:\\Program
System\\results\\landfill\\identify_image_landfill.jpg", self.image) else:
messagebox.showerror("Помилка!", "Неможливо знайти шаблон для даного зображення!") else:
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
def compare_landfill_images(self):
self.compare_window = ctk.CTkToplevel(self) self.compare_window.title("Порівняння знімків")
self.compare_window.geometry(f"{1200}x{630}") self.compare_window.resizable(False, False)
self.compare_window.down_compare =
ctk.CTkLabel(self.compare_window, text="Завантажити знімки:", font=ctk.CTkFont(size=16, weight="bold"))
self.compare_window.down_compare.place(x=30, y=30) self.compare_window.info_maxar =
ctk.CTkLabel(self.compare_window, text="Знімок MAXAR", font=ctk.CTkFont(size=16, weight="bold"))
self.compare_window.info_maxar.place(x=210, y=150) self.compare_window.info_sentinel_hub =
ctk.CTkLabel(self.compare_window, text="Знімок Sentinel Hub", font=ctk.CTkFont(size=16, weight="bold"))
self.compare_window.info_sentinel_hub.place(x=790, y=150) self.compare_window.add_sign =
ctk.CTkImage(dark_image=Image.open("C:\\Program System\\data\\sign.png"), size=(30, 30))
self.compare_window.compare_landfill_button = ctk.CTkButton(self.compare_window, text='',
command=self.load_compare_image,
compound=ctk.TOP, image=self.add_sign, text_color='black')
self.compare_window.compare_landfill_button.place(x=50, y=70)
self.compare_window.mainloop()
def load_compare_image(self):
self.filepath_compare_1 =
filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
self.input_img_1 =
ctk.CTkImage(dark_image=Image.open(self.filepath_compare_1), size=(500, 375))
self.filepath_compare_2 =
filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
self.input_img_2 =
ctk.CTkImage(dark_image=Image.open(self.filepath_compare_2), size=(650, 375))
input_label_1 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_1)
input_label_1.place(x=20, y=200)
```

‘

```
input_label_2 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_2)
input_label_2.place(x=540, y=200)
class Solar_Station_Window(ctk.CTkToplevel):
def __init__(self, parent):
super().__init__(parent)
self. init_solar_station()
def init_solar_station(self):
self.title("Дослідження сонячної станції") self.geometry(f"{1200}x{630}") self.resizable(False, False)
self.protocol("WM_DELETE_WINDOW", self.close_solar_station)
self.grid_columnconfigure(1, weight=1)
self.grid_columnconfigure((2, 3), weight=0) self.grid_rowconfigure((0, 1, 2), weight=1)
self.sidebar_frame = ctk.CTkFrame(self, width=140, corner_radius=0)
self.sidebar_frame.grid(row=0, column=0, rowspan=4, sticky="nsew")
self.sidebar_frame.grid_rowconfigure(4, weight=1)
self.down_label = ctk.CTkLabel(self.sidebar_frame, text="Завантажити знімок:", font=ctk.CTkFont(size=16, weight="bold"))
self.down_label.grid(row=0, column=0, padx=20, pady=(20, 10))
self.add_sign =
ctk.CTkImage(dark_image=Image.open("C:\\Program System\\data\\sign.png"), size=(30, 30))
self.load_landfill_button =
ctk.CTkButton(self.sidebar_frame, text='',
command=self.download_file_station, compound=ctk.TOP, image=self.add_sign, text_color='black')
self.load_landfill_button.place(x=40, y=60)
def close_solar_station(self):
if messagebox.askokcancel("Закрити вікно", "Впевнені, що
хочете закрити вікно?"):
self.destroy()
def download_file_station(self):
self.filepath =
filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
if self.filepath:
self.input_image =
ctk.CTkImage(dark_image=Image.open(self.filepath), size=(950, 580))
input_label = ctk.CTkLabel(self, text='',
image=self.input_image)
input_label.place(x=230, y=20)
operation_label = ctk.CTkLabel(self.sidebar_frame,
text='Оберіть операцію:', font=ctk.CTkFont(size=16, weight="bold")) operation_label.place(x=25, y=170)
enhance_button = ctk.CTkButton(self.sidebar_frame, text='Покращення якості',
command=self.enhance_station, width=70, height=50, text_color='black')
```

‘

```
enhance_button.place(x=40, y=210)
gray_button = ctk.CTkButton(self.sidebar_frame, text='Відтінок сірого',
command=self.gray_station, width=130, height=50, text_color='black') gray_button.place(x=40, y=290)
identification_button =
ctk.CTkButton(self.sidebar_frame, text='Виявлення країв',
command=self.edge_detect_station, width=130, height=50, text_color='black')
identification_button.place(x=40, y=370)
identification_button =
ctk.CTkButton(self.sidebar_frame, text='Морфологічне\nперетворення',
command=self.morph_transformation_station, width=130, height=50, text_color='black')
identification_button.place(x=40, y=450)
identification_button =
ctk.CTkButton(self.sidebar_frame, text='Ідентифікація',
command=self.identification_station, width=130, height=50, text_color='black')
identification_button.place(x=40, y=530)
else:
messagebox.showerror("Помилка!", "Оберіть знімок!")
def enhance_station(self):
if hasattr(self, 'input_image'):
img = Image.open(self.filepath)
enc_img = img.filter(ImageFilter.DETAIL)
img_con = ImageEnhance.Contrast(enc_img)
self.image_enh = img_con.enhance(1.3) self.image_enh.save('C:\\Program
System\\results\\solar_station\\enhanced_image_station.jpg') self.image_enh.show()
else:
messagebox.showerror("Помилка!", "Спочатку завантажте знімок!")
def gray_station(self):
if hasattr(self, 'input_image') and hasattr(self, 'image_enh'):
img_path = 'C:\\Program
System\\results\\solar_station\\enhanced_image_station.jpg' self.img = cv2.imread(img_path) b, g, r =
cv2.split(self.img)
rgb_img = cv2.merge([r, g, b])
self.bgr_to_gray = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)
self.gray_image = cv2.GaussianBlur(self.bgr_to_gray, (7, 7), 1)
plt.figure(figsize=(15, 7.6))
plt.imshow(self.gray_image, cmap='gray') plt.title('Gray Image')
plt.axis('off')
plt.show()
cv2.imwrite("C:\\Program
System\\results\\solar_station\\gray_image_station.jpg", self.gray_image)
```

‘

```
else:
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Покращення якості'!")
def edge_detect_station(self):
if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image'):
self.edged_image = cv2.Canny(self.gray_image, 150, 220)
plt.figure(figsize=(15, 7.6)) plt.imshow(self.edged_image, cmap='gray') plt.title('Edge detection Image') plt.axis('off') plt.show()
cv2.imwrite("C:\\Program
System\\results\\solar_station\\edge_detection_image_station.jpg", self.edged_image)
else:
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
def morph_transformation_station(self):
if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image') and hasattr(self, 'edged_image'):
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
self.morphology = cv2.morphologyEx(self.edged_image,
cv2.MORPH_CLOSE, kernel, iterations=2) plt.figure(figsize=(15, 7.6)) plt.imshow(self.morphology, cmap='gray') plt.title('Morphology transformation Image') plt.axis('off') plt.show()
cv2.imwrite("C:\\Program
System\\results\\solar_station\\morphology_transformation_image.jpg" , self.morphology)
else:
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Виявлення країв'!")
def identification_station(self):
if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image') and \
hasattr(self, 'edged_image') and hasattr(self,
'morphology'):
contours = cv2.findContours(self.morphology.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0] for contour in contours:
peri = cv2.arcLength(contour, True)
approx = cv2.approxPolyDP(contour, 0.02 * peri,
True)
if len(approx) == 4 and cv2.contourArea(contour) > 2000:
cv2.drawContours(self.img, [approx], -1, (0,
255, 0), 4)
plt.figure(figsize=(15, 7.6))
plt.imshow(cv2.cvtColor(self.img, cv2.COLOR_BGR2RGB)) plt.title('Object detection')
plt.axis('off')
plt.show()
cv2.imwrite("C:\\Program
System\\results\\solar_station\\identification_image.jpg", self.img) else:
```

‘

```
messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Морфологічне перетворення'!")
if __name__ == "__main__":
app = Main()
app.mainloop()
```

‘

ДОДАТОК Г

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«Автоматизована система дистанційного зондування земної поверхні з можливістю класифікації об'єктів»

Виконав: студент 4-го курсу,

групи 2АКІТ-20б

спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

_______ Дмитро ПАРХОМЕНКО

Керівник: к.т.н., доцент каф. КСУ

_______ Олена НІКІТЕНКО

«___» _______________ 2024 р.

Рецензент: к.т.н., доцент каф. АІІТ

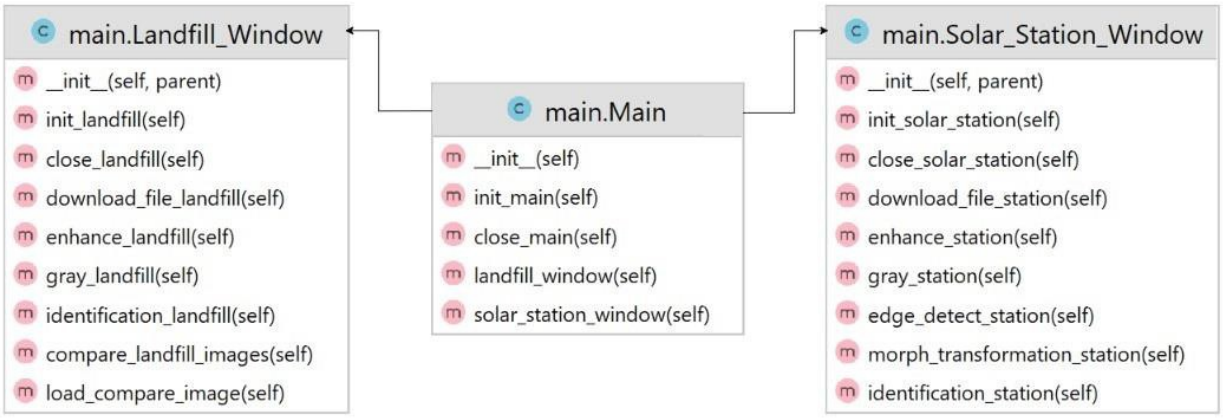
Юрій ІВАНОВ

(підпис)

«___» _______________ 2024 р.

Структурна схема системи

‘



Діаграма класів

```
def enhance_landfill(self):
    if hasattr(self, 'input_image'):
        img = Image.open(self.filepath)
        enc_img = img.filter(ImageFilter.DETAIL)
        img_con = ImageEnhance.Contrast(enc_img)
        self.img_enh = img_con.enhance(1.3)
        self.img_enh.save('results\\landfill\\enhanced_image.jpg')
        self.img_enh.show()
    else:
        messagebox.showerror("Помилка!", "Спочатку завантажте знімок!")
```

Демонстрація методу enhance_landfill(self)

‘

```
def identification_landfill(self):
    if hasattr(self, 'input_image') and hasattr(self, 'img_enh') and hasattr(self, 'gray_img'):
        template = None
        if self.filepath == 'C:/Program System/load_images/sentinel-hub/sentinel-hub_02-05-2023.jpg':
            template = cv2.imread("data\\landfill.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/sentinel-hub/sentinel-hub_16-05-2022.jpg':
            template = cv2.imread("data\\landfill_3.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/maxar/maxar_28-07-2022.jpg':
            template = cv2.imread("data\\landfill_1.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/maxar/maxar_12-10-2022.jpg':
            template = cv2.imread("data\\landfill_2.jpg", cv2.IMREAD_GRAYSCALE)
        elif self.filepath == 'C:/Program System/load_images/google maps/landfill.jpg':
            template = cv2.imread("data\\landfill_4.jpg", cv2.IMREAD_GRAYSCALE)

        if template is not None:
            w, h = template.shape[::-1]
            result = cv2.matchTemplate(self.gray_img, template, cv2.TM_CCOEFF_NORMED)
            loction = np.where(result >= 0.8)
            for pt in zip(*loction[::-1]):
                cv2.rectangle(self.image, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 2)
            plt.figure(figsize=(17, 8))
            plt.imshow(cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB))
            plt.title('Object detection')
            plt.axis('off')
            plt.show()
            cv2.imwrite("results\\landfill\\identify_image_landfill.jpg", self.image)
        else:
            messagebox.showerror("Помилка!", "Неможливо знайти шаблон для даного зображення!")
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
```

Демонстрація методу identification_landfill(self)

```
def compare_landfill_images(self):
    self.compare_window = ctk.CTkToplevel(self)
    self.compare_window.title("Порівняння знімків")
    self.compare_window.geometry(f"{1200}x{630}")
    self.compare_window.resizable(False, False)

    self.compare_window.down_compare = ctk.CTkLabel(self.compare_window, text="Завантажити знімки:", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.down_compare.place(x=30, y=30)
    self.compare_window.info_maxar = ctk.CTkLabel(self.compare_window, text="Знімок MAXAR", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.info_maxar.place(x=210, y=150)
    self.compare_window.info_sentinel_hub = ctk.CTkLabel(self.compare_window, text="Знімок Sentinel Hub", font=ctk.CTkFont(size=16, weight="bold"))
    self.compare_window.info_sentinel_hub.place(x=790, y=150)
    self.compare_window.add_sign = ctk.CTkImage(dark_image=Image.open("data\\sign.png"), size=(30, 30))
    self.compare_window.compare_landfill_button = ctk.CTkButton(self.compare_window, text='', command=self.load_compare_image,
                                                                compound=ctk.TOP, image=self.add_sign)
    self.compare_window.compare_landfill_button.place(x=50, y=70)

    self.compare_window.mainloop()
```

Демонстрація методу compare_landfill_images(self)

```
def load_compare_image(self):
    self.filepath_compare_1 = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
    self.input_img_1 = ctk.CTkImage(dark_image=Image.open(self.filepath_compare_1), size=(500, 375))
    self.filepath_compare_2 = filedialog.askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.gif")])
    self.input_img_2 = ctk.CTkImage(dark_image=Image.open(self.filepath_compare_2), size=(650, 375))
    input_label_1 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_1)
    input_label_1.place(x=20, y=200)
    input_label_2 = ctk.CTkLabel(self.compare_window, text='', image=self.input_img_2)
    input_label_2.place(x=540, y=200)
```

‘

Демонстрація методу load_compare_image(self)

```
def edge_detect_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image'):
        self.edged_image = cv2.Canny(self.gray_image, 150, 220)
        plt.figure(figsize=(15, 7.6))
        plt.imshow(self.edged_image, cmap='gray')
        plt.title('Edge detection Image')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\edge_detection_image_station.jpg", self.edged_image)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Відтінок сірого'!")
```

Демонстрація методу edge_detect_station(self)

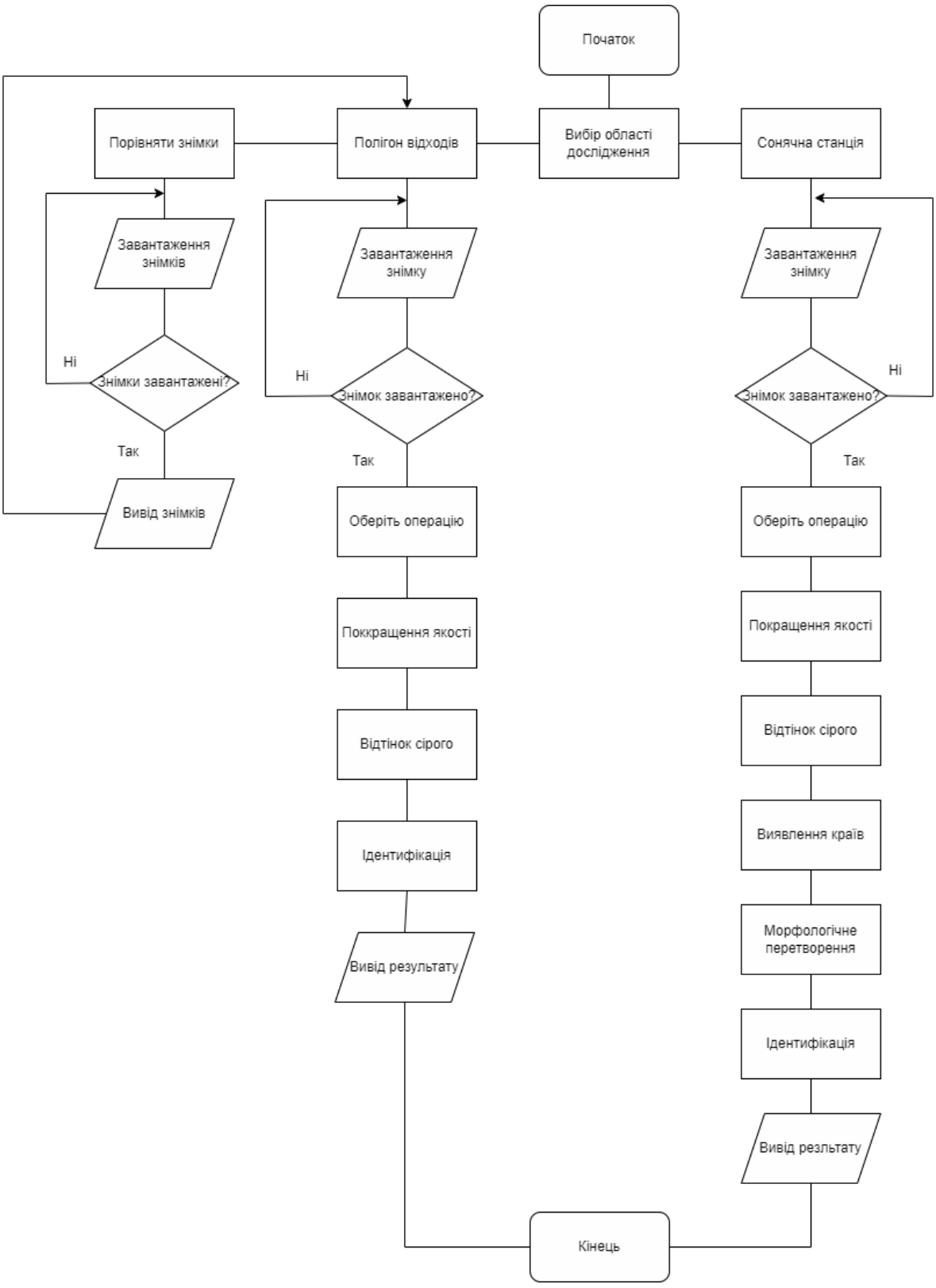
```
def identification_station(self):
    if hasattr(self, 'input_image') and hasattr(self, 'image_enh') and hasattr(self, 'gray_image') and \
            hasattr(self, 'edged_image') and hasattr(self, 'morphology'):
        contours = cv2.findContours(self.morphology.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
        for contour in contours:
            peri = cv2.arcLength(contour, True)
            approx = cv2.approxPolyDP(contour, 0.02 * peri, True)

            if len(approx) == 4 and cv2.contourArea(contour) > 2000:
                cv2.drawContours(self.img, [approx], -1, (0, 255, 0), 4)

        plt.figure(figsize=(15, 7.6))
        plt.imshow(cv2.cvtColor(self.img, cv2.COLOR_BGR2RGB))
        plt.title('Object detection')
        plt.axis('off')
        plt.show()
        cv2.imwrite("results\\solar_station\\identification_image.jpg", self.img)
    else:
        messagebox.showerror("Помилка!", "Спочатку оберіть операцію - 'Морфологічне перетворення'!")
```

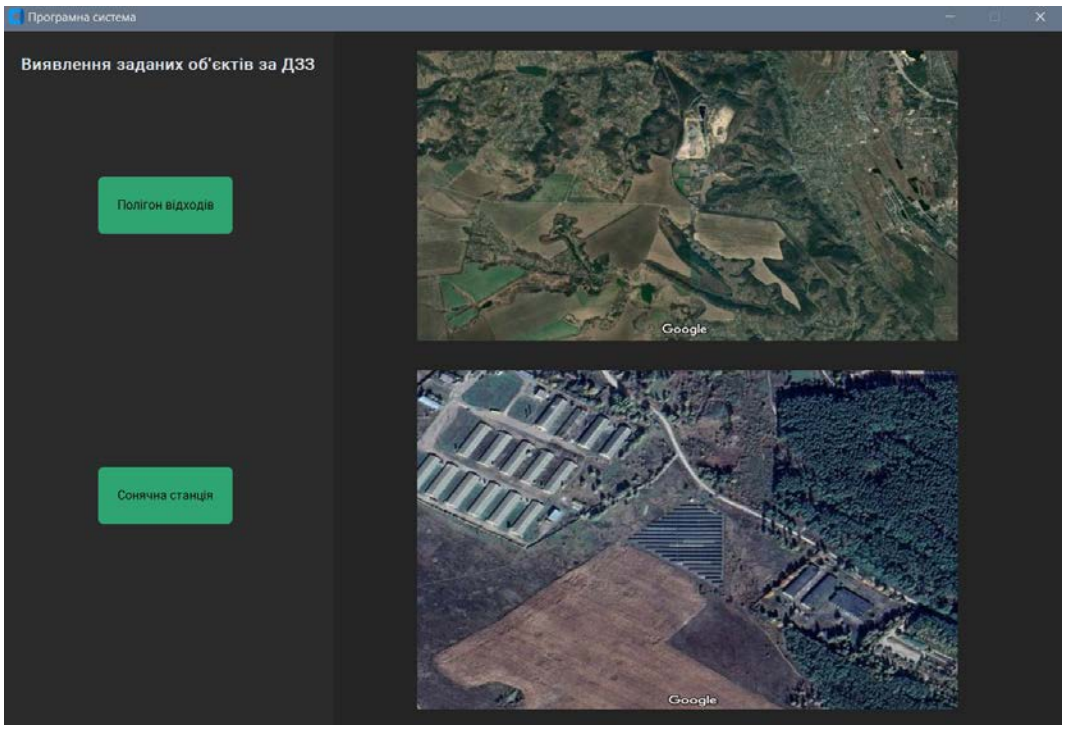
Демонстрація методу identification_station(self)

‘

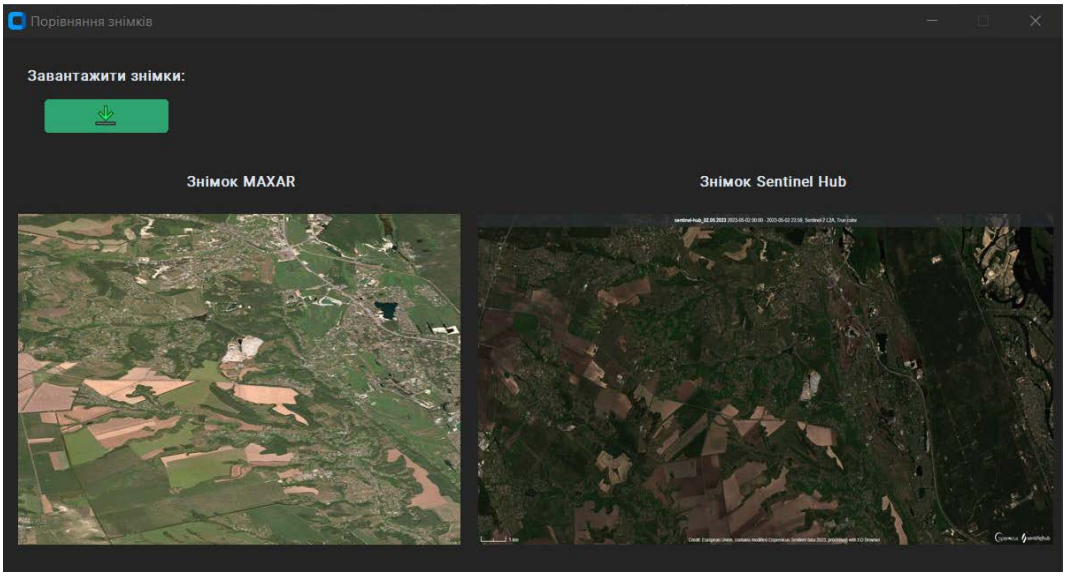


Базовий алгоритм роботи системи

‘

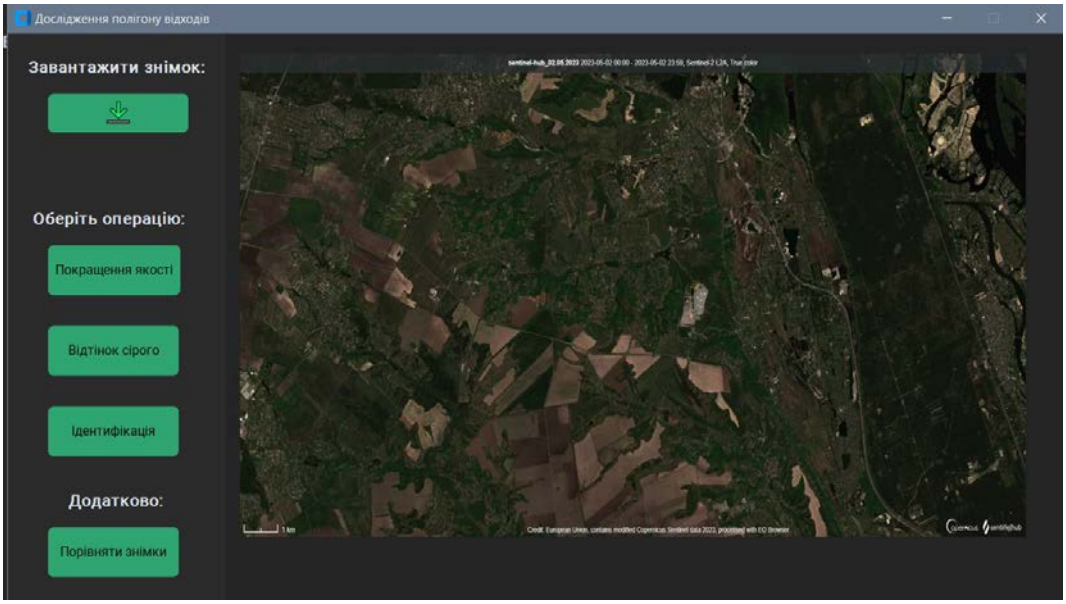


Ілюстрація головного вікна програми

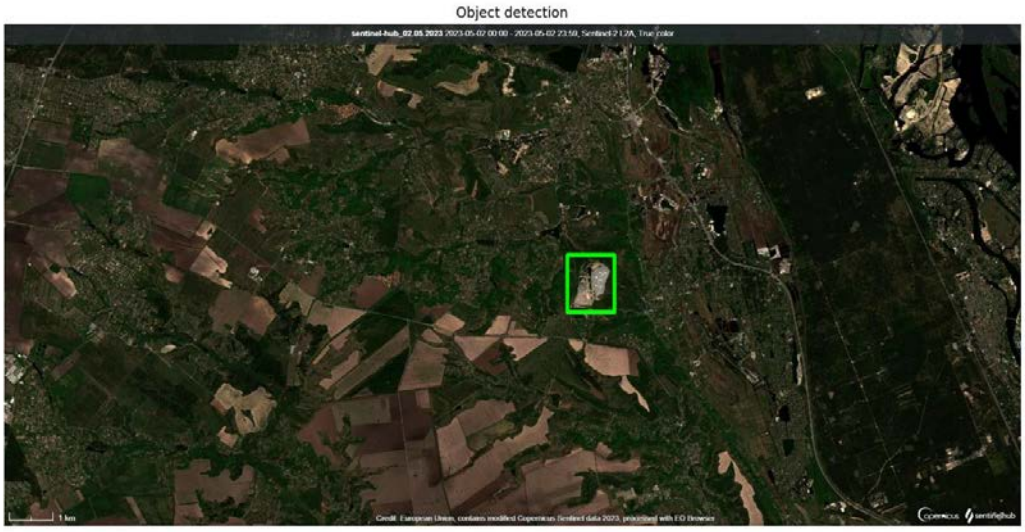


Вивід знімків на вікно для порівняння

‘



Вікно полігону відходів після завантаження знімку



Ідентифікація об'єкта після натискання на третю операцію

‘

Вікно сонячної станції після завантаження знімку

‘