Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження методів розпізнавання за допомогою нейронної мережі щодо аналізу та побудови карт з використанням GPS треку з прив'язками до візуальних даних
(тема)

Виконав: студент 2 курсу, групи ІПЗм-17-1
спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-професійної програми Інженерія програмного забезпечення

_____Черепахіна К.К._____
(прізвище, ініціали)
Керівник _____доц. Назаров О.С._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121– Інженерія програмного забезпечення
(код і повна назва)

Освітньо-професійна програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Черепахіної Катерини Кирилівни
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів розпізнавання за допомогою нейронної мережі щодо аналізу та побудови карт з використанням GPS треку з прив'язками до візуальних даних
затверджена наказом по університету від «18» квітня 2019 р. № 564 Ст
2. Термін подання студентом роботи до екзаменаційної комісії «20» червня 2019 р.
3. Вихідні дані до роботи *засоби покращення продуктивності веб-додатків, пояснювальна записка. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування*
4. Перелік питань, що потрібно опрацювати в роботі *позначка роботи, аналіз проблемної галузі і постановка задачі, опис проблем продуктивності, використовувані методи та алгоритми, опис розробленої програмної системи, аналіз можливих застосувань*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Мета завдання, обґрунтування доцільності розроблення, постановка задачі, об'єктна модель системи, базові моделі, інтерфейс програмної системи, результати тестування програмної системи, демонстраційні матеріали* _____

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Назаров О.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	19 травня 2019р.	Виконала
2.	Огляд існуючих методів	27 травня 2019р.	Виконала
3.	Методи підвищення ефективності ве додатків	25 травня 2019р.	Виконала
4.	Підготовка пояснювальної записки	26 травня 2019р.	Виконала
5.	Спецчастина	31 травня 2019р.	Виконала
6.	Підготовка презентації та доповіді	9 червня 2019р.	Виконала
7.	Попередній захист	10 червня 2019р.	Виконала
8.	Нормоконтроль, рецензування	11 червня 2019р.	Виконала
9.	Занесення диплома в електронний архів	14 червня 2019р.	Виконала
10	Допуск до захисту у зав. кафедри	17 червня 2019р.	Виконала

Дата видачі завдання _____ _____2019 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Назаров О.С.
(підпис)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 87 с., 48 рис., 3 таблиці, 3 додатки, 23 джерел.

МАПА, НЕЙРОНА МЕРЕЖА, .NET, РОЗПІЗНАВАННЯ ОБ'ЄКТІВ, СКЛАДАННЯ ЗОБРАЖЕНЬ, ДОСЛІДЖЕННЯ МІСЦЕВОСТІ, ГЕОРАХУВАННЯ, ЛОКАЦІЯ

Об’єктом дослідження є існуючі можливості використання дронів для дослідження місцевості та спостереження за об’єктами з висоти.

Метою роботи є створення ефективної системи для інтелектуального створення та доповнення мап та можливістю знаходження об’єктів на обраному участку місцевості та відмічання їх на мапі.

Методи розробки базуються на технології .Net Core, Micro .Net Framework, мові C# та C++.

У результаті роботи здійснена інтелектуальна система що складається з програми та апаратної частини, що використовується для створення мап та розпізнавання об’єктів на місцевості.

MAP, NEURON NETWORK, .NET, OBJECTS RECOGNITION, IMAGE COMPOSITION, LOCATION STUDY, GOREFLOWING, LOCATION

The object of the research is the existing possibilities of using drones for studying the area and observing objects from the altitude.

The aim of the work is to create an effective system for intelligent creation and updating of maps and the possibility of finding objects in the chosen area of the area and marking them on the map.

Development methods are based on .Net Core, Micro .Net Framework, C # and C ++.

As a result of the work, an intelligent system consisting of a program and a hardware part used to create a map and object recognition on the ground is carried out.

ЗМІСТ

ВСТУП

Інтереси сучасного суспільства змінюються вкрай швидко, але що залишається незмінним так це цікавість до подорожей і пізнання світу. Важливою частиною життя людини стало знайомство з цікавими куточками планети, тому для цього було придумано безліч різних способів починаючи від подорожей і закінчуючи фотографіями та відеозйомки. У свою чергу інновацією поточного сторіччя стала можливість дослідження планети за допомогою знімками із супутників або картографічної фіксації поверхні.

На превеликий жаль ці способи ще не досягли належного рівня розвитку який задовольнить потреби будь-якого користувача, в незалежно від складності його запиту.

Кожна людина хоча б один раз за час використання супутникових карт стикався з проблемою недостатньої точності знімків супутника або відсутності еге в цілому. Створення нових супутникових знімків є дуже дорогим процесом і не завжди існує можливість створити стабільне зображення.

У той час як додаток відсутніх фрагментів зображень за допомогою різних літальних апаратів вимагає трудомісткого процесу створення, а також подальшої обробки, при цьому не завжди вдається побудувати її з першої спроби.

На даний момент використання супутникових карт розширюється і потреба в максимально деталізованих картах зростає в багатьох областях. Найбільш поширеними серед всіх областей вважаються області: будівництва, археології, військової розвідки й звичайно ж у звичайних користувачів.

Також це застосовується час від часу для оновлення супутникових карт, тому що це засіб дешевше ніж оновлювати супутникові знімки.

Ще однією досить популярною функцією робіт з картами є знаходження певних елементів на ділянках, будь то розташування ворожого табору або відзначити все грядки з помідорами на бабусиному городі.

Не важливо які цілі переслідує користувач, адже попит на подібні функції зростає щодня. Метою даної роботи полягає створення кошти створення карт в реальному часі за допомогою літальних апаратів (переважно БПЛА), яка містити функцію розпізнавання заданих об'єктів на поверхні за допомогою нейронної мережі.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

На сьогодні існує безліч способів подивитися і ознайомитися з будь-яким куточком нашої планети за допомогою різних способів, будь то перегляд фото, знятих з супутника, або картографічна фіксація поверхні ділянок землі, але складність і вартість створення нових карт велика. Починаючи від процесу фіксації і склейки нових матеріалів, закінчуючи виявленням на них об'єктів і аномалій.

1.1 Аналіз предметної області

Аналіз даної роботи можна розділити на два етапи. Перший етап це так званий ортофотоплан. Другим етапом можна виділити розпізнавання об'єктів за допомогою нейронної мережі.

1.1.1 Ортофотоплан

Ортофотоплан - це прямокутний план на основі трансформованих аерознімків, прив'язаних до геодезичній основі і чітким орієнтуванням на місцевості. Ортотрансформування проводиться шляхом суміщення прилеглих знімків і визначення відповідності точок, в основі лежить побудова цифрової моделі і тріангуляція мережі. Ортофотоплан є основою топографічних планів, кадастрових карт, базисів інженерних вишукувань,

карт місцевості. За своєю суттю виконує візуальну роль для характеристики району. Широке застосування ортофотоплани отримали під час ведення військових дій і зараз використовуються в топографічних, геологічних, а також застроечних і меліоративних роботах. Крім того фотоплани використовуються при формуванні та оновленні цифрових карт і оперативної оцінки стану ґрунтів, зеленої рослинності, обробки і якості ріллі, і звичайно ж в будівництві, особливо в котеджної забудови.

Ортотрансформування усуває спотворення на знімку, обумовлені рельєфом місцевості і відхиленнями осі фотоапарата від вертикалі при зйомці, шляхом послідовного проектування трансформованого зображення можливо малими ділянками за допомогою ортофотопроекторів (див. рис. 1.1).

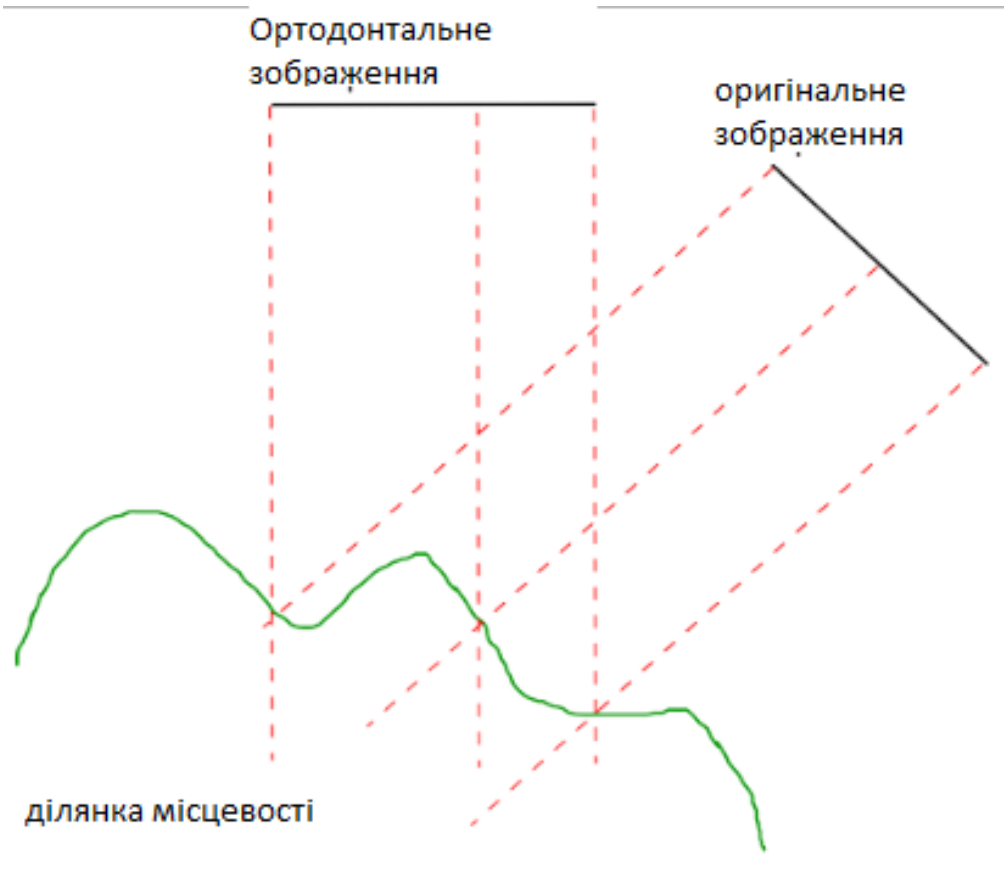


Рисунок 1.1 – Перерозподіл пікселів на зображенні

В результаті перетворення виходять ортофотознімків, які дозволяють скласти ортофотоплани на будь-які райони. Ортофотоплани особливо затребувані при геодезичних, топографічних, геологічних, гідрологічних, екологічних дослідницьких роботах, землеустрій, архітектурно-будівельному проектуванні і контролі будівельно-монтажних робіт [1].

Створення ортофотоплану проходить в три етапи:

— Редакційно-підготовчі роботи – з'ясовується наявність даних для трансформування знімка, готуються файли вихідних даних, виконується копіювання вихідного знімка (вихідного растра), створюється матриця рельєфу;

— Трансформування окремих знімків - на вихідному растрі орієнтуються і вимірюються опорні точки, а також створюється файл трансформованого зображення вихідного растру (трансформованого растра);

— Створення ортофотоплану – відбувається зведення зображення на стиках трансформованих растрів (зшивання фотопланів) і нарізка фрагментів трансформованого растра по специфікаціях.

1.1.2 Нейронні мережі і машинне навчання

Нейронна мережа (див. рис. 1.2) – обчислювальна модель, яка функціонує майже як людський мозок. Кожен нейрон отримує інформацію, обробляє і передає наступному.

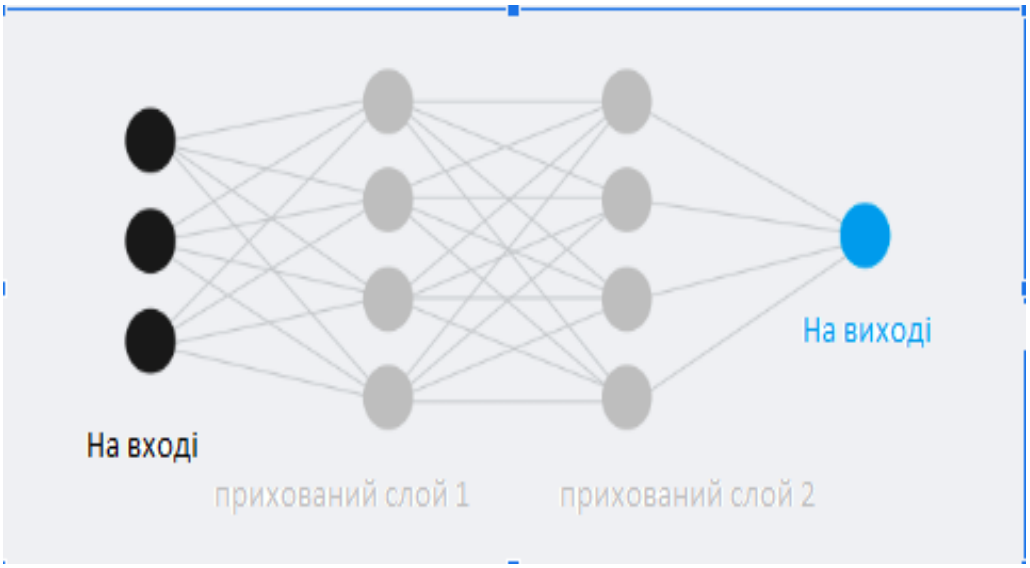


Рисунок 1.2 – Схематичне зображення 3-рівневої нейронної мережі

Штучна нейронна мережа зазвичай навчається з учителем. Це означає наявність навчального набору (датасета), який містить приклади з істинними значеннями: тегами, класами, показниками.

Нерозмічену набори також використовують для навчання нейронних мереж, але ми не будемо тут це розглядати.

Наприклад, якщо ви хочете створити нейросеть для оцінки тональності тексту, датасета буде список пропозицій з відповідними кожному емоційними оцінками. Тональність тексту визначають ознаки (слова, фрази, структура пропозиції), які дають негативну чи позитивну забарвлення. Ваги ознак в підсумковій оцінці тональності тексту (позитивний, негативний, нейтральний) залежать від математичної функції, яка обчислюється під час навчання нейронної мережі.

Раніше люди генерували ознаки вручну. Чим більше ознак і точніше підібрані ваги, тим точніше відповідь. Нейронна мережа автоматизировала цей процес [2].

Штучна нейронна мережа складається з трьох компонентів (див. рис.1.2):

— Вхідний шар.

— Приховані (обчислювальні) шари.

— Вихідний шар.

Навчання нейромереж відбувається в два етапи:

— Пряме поширення помилки.

— Зворотне поширення помилки.

Під час прямого поширення помилки робиться прогноз відповіді. При зворотному поширенні помилка між фактичним відповіддю і передбаченим мінімізується [2].

Глибоке навчання (deep learning) – це клас алгоритмів машинного навчання, які навчаються глибше (більш абстрактно) розуміти дані.

Більш формально в deep learning:

— Використовується каскад (пайплайн, як послідовно переданий потік) з безлічі обробних шарів (нелінійних) для вилучення і перетворення ознак;

— Ґрунтується на вивченні ознак (поданні інформації) в даних без навчання з учителем. Функції вищого рівня (які знаходяться в останніх шарах) виходять з функцій нижнього рівня (які знаходяться в шарах початкових шарах);

— Вивчає багаторівневі уявлення, які відповідають різним рівням абстракції; рівні утворюють ієрархію уявлення.

Найскладніше завдання в роботі з нейромережею – грамотно підібрати коефіцієнти до нейронам.

Для цього використовується навчання – процес знаходження коректних ваг для нейромережі. Від того, як саме навчать нейромережу, будуть залежати її вирішення.

1.2 Формування проблем і актуальних рішень

Карти і їх створення налічує багато століть. Люди використовували карти як засобу для орієнтування на місцевості. З розвитком людства розвивалися і карти. З'явилося безліч видів і застосування карт. За останні десятиліття карти отримали нове застосування. З розвитком електронних пристроїв з'явилася можливість зробити загальнодоступними супутникові знімки, згодом почали активно розроблятися різні системи дозволяють подивитися на нашу планету зверху вниз.

У теперішній час досить популярним ставати створення власних карт деяких ділянок землі. Маючи ринок споживачів від звичайних землевласників до військових, від археологів до пошукових служб. З розвитком БПЛА за останні 5 років стали розвиватися і таке поняття як ортофотоплан. На сьогоднішній день це найдешевший спосіб створення карт поверхні землі.

Ортофотопланування досить трудомісткий процес і не дозволяє створювати карти в реальному часі. Він відбувається в три етапи один з яких потребує втручання людини і у більшості випадків вимагає двох або більше разів проходження першого етапу (етап збору матеріалів) для більш точного результату. Також важливим моментом є те що абсолютно всі елементи на створеній карті потрібно відзначати вручну. І на жаль це не завжди ефективно, так як всі маніпуляції здійснюються з готовим продуктом, що не дає можливості оцінити об'єкт з різних сторін. Слідуючи з цього якщо ви побачите на готовому продукті лише частина будь-якого об'єкта ви не можете визначити його з максимальною точністю.

З цього можна виділити кілька проблем, які потребують вирішення в даній роботі.

Першою проблемою є створення стабільного зображення досить високої якості. Для вирішення даної проблеми потрібно знайти стабільне обладнання та встановити на нього розробляється продукт в якості ПО. Бажано використовувати БПЛА для більш стабільного і маневреного процесу отримання зображень.

Наступна важливіша проблем. Потрібно створити ПО для швидкої обробки одержаної інформації і склейки його в продукт готовий до використання. Основною проблемою в даному питанні знайти обладнання яке буде стабільно передавати інформацію без затримок.

Однією з найважливіших проблем є знаходження невідомих або вже заздалегідь відомих об'єктів на поверхні. Основна проблема даного рішення це навчання нейронної мережі для отримання очікуваного результату. Даний процес є найбільш трудомістким.

Останньою важливою частиною є розпаралелювання процесів створення карт і знаходження на них об'єктів. Дані процеси повинні відбуватися одночасно для поліпшення якості знаходження об'єктів. Так як повне зображення об'єкта на кінцевому результаті карти може бути неповним через можливості перешкоди іншим об'єктом або накладанням об'єкта на об'єкт.

Таким чином ми виявили основні проблеми даної роботи.

1.3 Аналіз існуючих рішень

Досліджуючи ринок було виявлено відсутність повного аналога даного продукту, але досить велика кількість систем які виконують всі функції окремо. У теперішній час існує чимало різних рішень побудови

карт, починаючи від супутників закінчуючи БПЛА. У той час як розпізнавання об'єктів все ще не досягло належного рівня. Досить велика кількість таких проектів перебувають на стадії розробки або ж недостатньо ефективні.

Так як повного аналога не існує розглянемо кожну функцію продукту окремо.

1.3.1 Google maps

Коли мова заходить про веб-картографії першим на думку спадає Google Maps (див. рис. 1.3). Даний набір додатків являє собою карту та супутникові знімки планети Земля. Так само для деяких регіонів доступні аерофотознімки дозволяють збільшувати поверхню до 10 метрів над рівнем землі. Цей сервіс має велику кількість функцій, але лише один розглянемо докладно.

Рисунок 1.3 – Логотип «Google Maps»

Аерофотознімки доступні лише для 19 регіону. У нього входить Сан Франциско, США. Розглянемо на прикладі даного міста специфіку даної функції. Гугл виробляв аерофотозйомку з висоти 250-500 м над рівнем землі і як ми можемо помітити на рисунку 1.4 цього не достатньо, щоб зображення при максимальному наближенні виглядало не як стара комп'ютерна гра з поганою графікою.

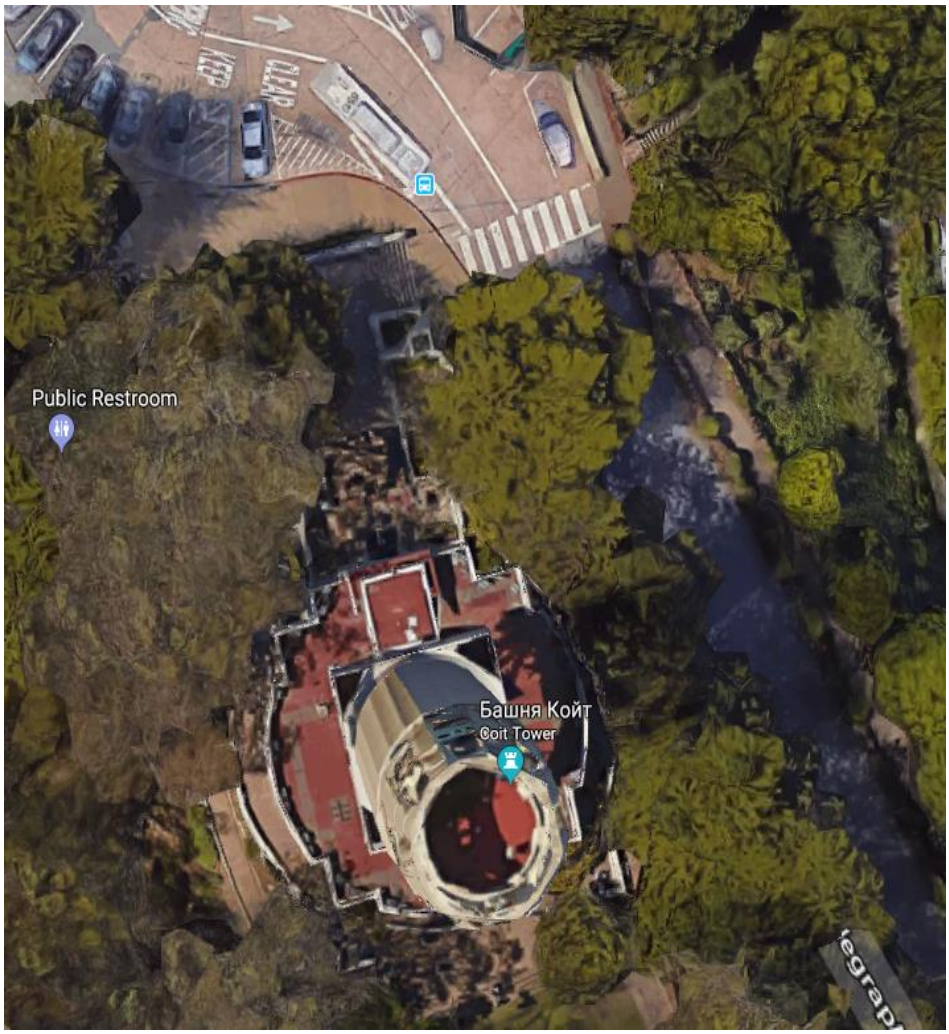


Рисунок 1.4 – Приклад аерофотознімки

Можна відзначити функцію яку варто взяти на замітку при створенні програми. При пересуванні ділянки нахил більше високих будівель змінюється нібито ви в даний момент пролітає над ним.

Підводячи підсумок по даному додатку можна зробити висновок, що для хорошої якості зйомки потрібно перебувати максимально можливо до земної поверхні.

1.3.2 Agisoft PhotoScan

У програмі Agisoft PhotoScan (див. рис. 1.5) реалізована сучасна технологія створення тривимірних моделей високої якості на основі цифрових фотографій.

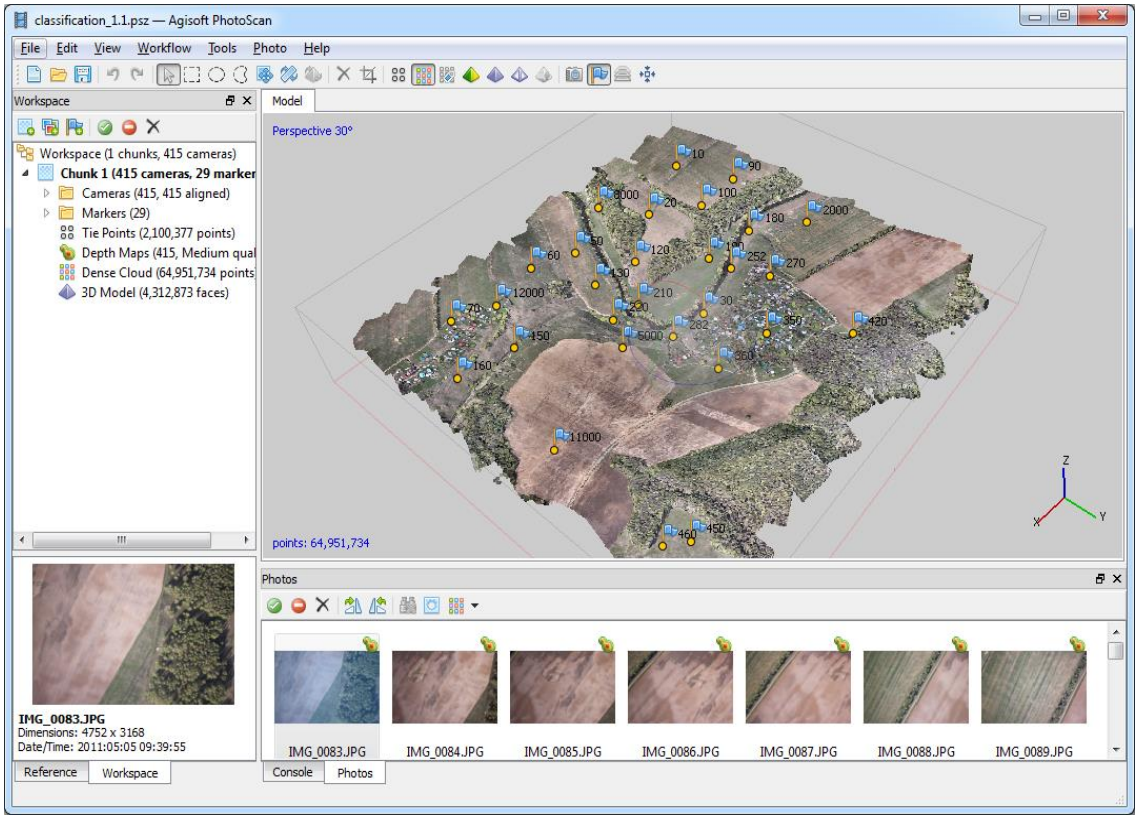


Рисунок 1.5 – Приклад робочого вікна

Для реконструкції 3D моделі об'єкта Agisoft PhotoScan дозволяє використовувати фотографії, зняті будь-якими цифровими фотокамерами з будь-яких ракурсів (за умови, що кожен елемент реконструйованої сцени видно принаймні з двох позицій зйомки). Процес створення тривимірної моделі повністю автоматизований.

Для моделей з заданим масштабом Agisoft PhotoScan також дозволяє вимірювати відстані і розраховувати площа поверхні і об'єм. Масштабування моделі проводиться на підставі попередніх вимірювань в межах реконструюється сцени.

Важливо відзначити що всі процеси моделювання здійснюються вручну, що займає велику кількість часу. В обробці можуть потребувати від 10 до нескінченності знімків в залежності від обсягу роботи.

Слід зазначити, що сам процес склеювання відбувається дуже простим і ефективним способом, що є плюсом і на це варто звернути увагу при створенні даного продукту. Також є дуже значимий мінус який суперечить поставленої мети роботи. Всі зображення обробляються вручну без будь-якої автоматизації. Це питання буде вирішене в даній роботі.

1.3.3 DroneDeploy

Додаток для iOS і Android замінює собою стандартний додаток DJI Go 4 для управління дроном.

У мобільному додатку можна спланувати маршрут і запустити дрон в політ по розрахованому маршруту. Він автоматично зробить фото в потрібних точках і під потрібним ракурсом. Після польоту відзняті фото з SD-карти пам'яті необхідно завантажити на сервер DroneDeploy (див.

рис.1.6) для обробки. Після обробки фотоплани і 3D моделі можна подивитися як через мобільний додаток так і через браузер, а також надіслати поштою з можливістю перегляду в браузері без спеціального ПЗ.

Рисунок 1.6 – Логотип «DroneDeploy»

Мобільний додаток може працювати в оффлайн режимі з попередньо збереженим маршрутом і кешувати картами Google Maps. Залежно від необхідного дозволу фотоплана в додатку потрібно встановити висоту польоту дрона. Чим вище дозвіл, тим більше фотографій потрібно зробити, і може вийде так, що для великої території не вистачить одного акумулятора. Додаток дозволяє продовжити політ за маршрутом з перерваного місця після заміни акумулятора або виключення дрона.

2 ПОСТАНОВА ЗАДАЧІ

2.1 Постанова задачі

Мета полягає в створенні гнучкої і розширюється системи, здатної працювати на невідомих ділянках створюючи нові карти з високим розширенням. Створення карт повинне відбуватись в режимі онлайн, або рішучості офлайн, але обробляти зображення і створювати карти відразу після надходження інформації. Знаходити на вибраному ділянці потрібні об'єкти і позначати їх, ведучи облік.

На сьогодні існує безліч способів подивитися і ознайомитися з будь-яким куточком нашої планети за допомогою різних способів, будь то перегляд фото, знятих з супутника, або картографічна фіксація поверхні ділянок землі, але складність і вартість створення нових карт велика. Починаючи від процесу фіксації і склейки нових матеріалів, закінчуючи виявленням на них об'єктів і аномалій.

Перша проблема панорамування полягає в побудові одного складеного зображення на основі набору вихідних зображень (див. рис. 2.1).

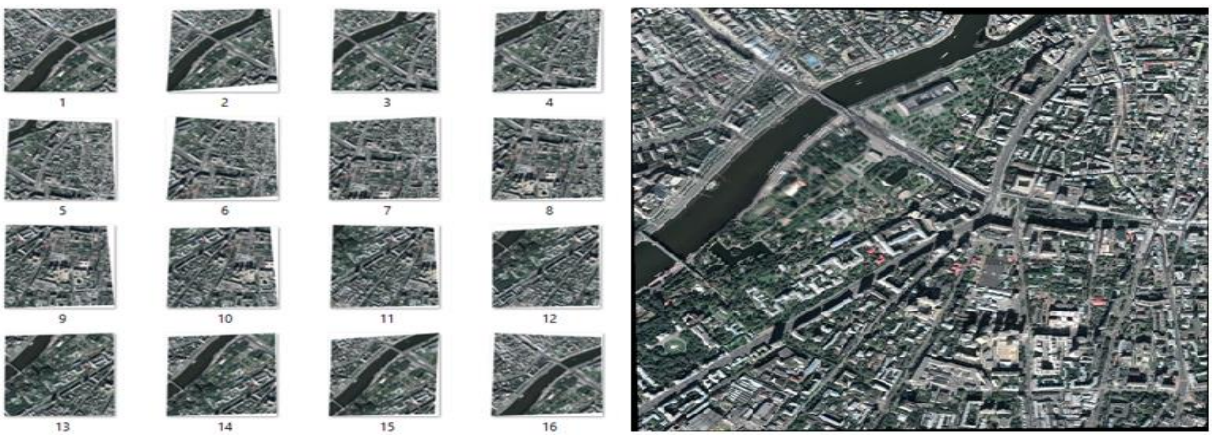


Рисунок 2.1 – Приклад набору вихідних зображень

Вона знаходить застосування у вирішенні таких практичних задач:

— зондування поверхні Землі з супутника або з дрона;

— склейка зображень, отриманих за допомогою мікроскопа;

— склейка відео;

— отримання зображення в супер-дозволі.

У загальному вигляді алгоритм склеювання панорами можна сформулювати наступним чином [3]. На самому початку потрібно витягти з відеопотоку достатню кількість кадрів. Це можна робити в режимі "онлайн", у якому послідовно зчитуються всі кадри і вибираючи окремі з них з необхідною частотою.

Таким чином існуючий алгоритм, являє нам першу помилку, яка випливає з основних умов алгоритму:

— ведеться зйомка нерухомого псевдовірогідного об'єкта;

— ведеться зйомка об'єкта близького до плоского з досить великої відстані;

— для всіх положень камери під час зйомки виконано вимогу: для всіх точок зображення промені, що з'єднують ці точки з фокусом камери, не збігаються один з одним.

А саме склейка в процесі руху, що в задачі динамічного побудови карт (див. рис. 2.2), є неминучим, отже, з'являється потреба в запровадженні нового і вдосконаленого типу обробки такого роду зображень, а саме впровадження алгоритму узгодження графа проектних перетворень [4].

Введемо поняття єдиної системи координат. Під єдиною системою координат будемо розуміти таку систему координат, де одні і ті ж точки об'єкта з різних зображень матимуть однакові координати [4]. Ця вимога можна виразити наступною формулою:

$$f(x)=y, \quad (2.1)$$

де

f - це відображення певне на загальній частині кадрів і переводить точки першого кадру в точки другого кадру;

x - координати точки в системі координат першого кадру;

y - координати точки в системі координат другого кадру.

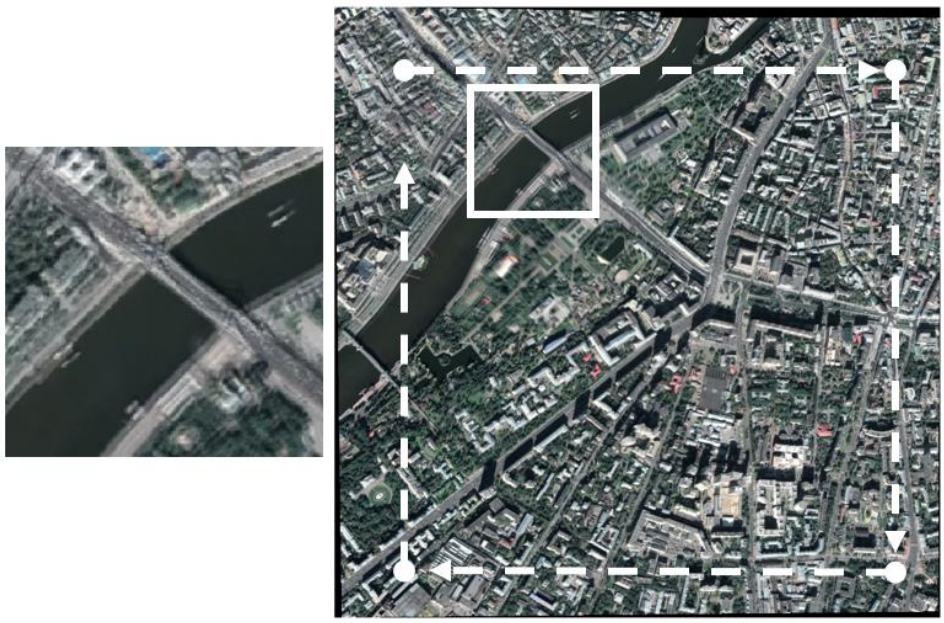


Рисунок 2.2 – Приклад динамічного зображення

У тому випадку, коли відображення f може бути коректно продовжено за область перетину кадрів, ми можемо доповнити другий кадр інформацією з першого. Таким чином буде отримана карта, склеєна як мозаїка, з двох або більше кадрів [5].

Після знаходження проектних перетворень між сусідніми кадрами є первісна склейка, що задає однозначне розташування кадрів в єдиній системі координат (див. рис. 2.3).

На сьогоднішній день не існує універсального методу оцінки якості склеювання зображень. Як правило якість склейки оцінюється експертами

органолептичним методом, але для наукових досліджень краще мати кількісну, автоматично обчислює оцінку якості [6].

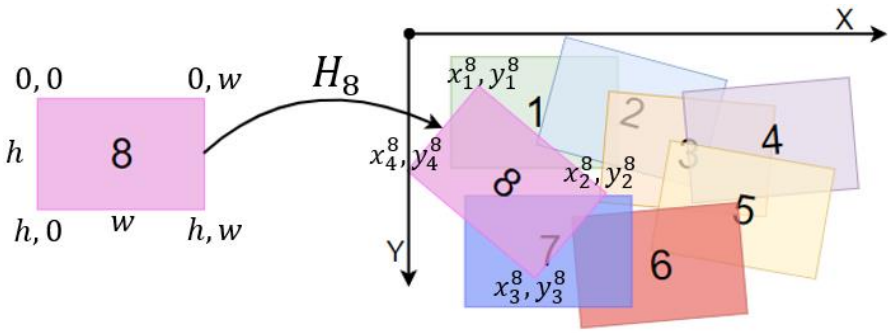


Рисунок 2.3 – Однозначне розташування кадру на мапі

У роботі з ново-впровадженим алгоритмом для кількісної оцінки якості склеювання панорами, маючи зображення з високою роздільною здатністю, створити штучне відео, кадрами якого є проективно спотворені області вихідного зображення проективної спотворюються всі кадри, за винятком першого, оскільки єдина система координат задається щодо першого кадру. Далі ці кадри штучного відео склеюються в панораму, яка в подальшому порівнюється з вихідним еталонним зображенням (див. рис. 2.4). При цьому підході вдається уникнути проблем різниці яскравості отриманої та еталонної склеювань, а також спотворень сцени.

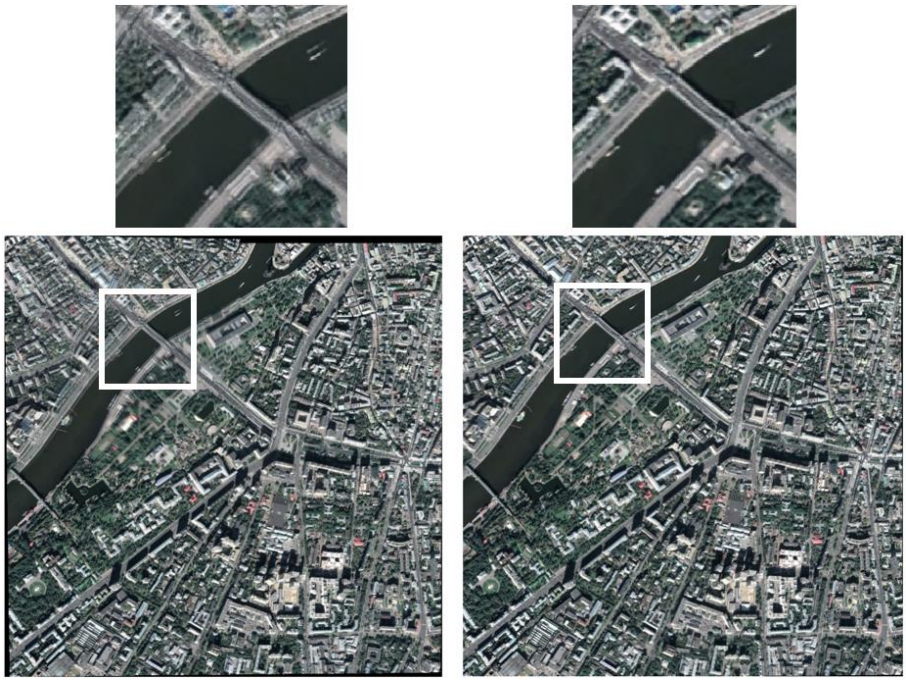


Рисунок 2.4 – Приклад порівняння до еталонної склеюванні

Наступною не вирішеною проблемою є узгодження карти, з результатами і матеріалами отриманими в ході дослідження (див. рис. 2.5 і рис. 2.6).

Рисунок 2.5 – Початкова карта

Рисунок 2.6 – Обновлена карта

Сам рух і коригування камери, в цьому завданні не має значення, а ось матеріали, які в подальшому потребують пост обробки на момент знаходження на ній об'єктів - є дуже складне завдання, яку розумним буде вирішити за допомогою впровадження нейронної мережі [10].

Найскладніше, що не видно, як можна вирішити це питання механічним способом. Комп'ютерний зір пов'язаний з відповідними проблемами дуже давно, з моменту своєї появи, і періодично знаходило ефективні приватні рішення - так, ми можемо пізнати зрушений в бік предмет, послідовно пересуваючи свій перевірки патерн по всьому зображенню (чим успішно користуються згорткові мережі), можемо справлятися з масштабуванням або поверненням картинок за допомогою ознак SIFT, SURF і ORB, але ефекти перспективи і поворот предмета в просторі сцени - схоже, речі якісно іншого рівня [11].

Тут нам потрібно знати, як предмет виглядає з усіх боків, отримати його справжню тривимірну форму, інакше нам нема з чим працювати.

Тому щоб розпізнавати картинки, не потрібно розпізнавати картинки. Вони брехливі, оманливі і свідомо неповноцінні. Вони – не наші друзі.

Важливе питання - як би нам отримувати тривимірну модель всього, що ми бачимо? Ще більш важливе питання – як при цьому обійтися без необхідності купувати лазерний просторовий сканер?

Навіть не стільки з тієї причини, що нам шкода, а тому, що тварини в процесі еволюції зорової системи явно якимось чином обійшлися без нього, одними лише очима, і було б цікаво дізнатися, як вони так, рішення:

— Взяти два кадри, зняті з відкаліброваною камери.

— Разом з параметрами калібрування (матрицею камери) покласти їх обидва в функцію stereoRectify, яка випрямить (ректифіков) ці два кадри – це перетворення, яке спотворює зображення так, щоб точка і її зміщення виявлялися на одній горизонтальній прямій.

— Ці ректіфіцірованні кадри ми кладемо в функцію stereoBM і отримуємо карту зсувів (disparity map) - таку картинку в відтінках сірого, де ніж піксель яскравіше, тим більший зсув він висловлює (за посиланням є приклад);

— Отриману карту зсувів кладемо в функцію з промовистою назвою reprojectImageTo3D (знадобиться ще і матриця Q, яку в числі інших ми отримаємо на кроці 2). Отримуємо наш тривимірний результат (див. рис. 2.7).

Таким чином, реалізувавши два відмінних рішення, для основних проблем в цій галузі, і з'єднати їх для парної і поетапної обробки зображень, з'являється можливість побудувати швидку і максимально точну систему, за допомогою якої можливо буде будувати карти включає в себе, метатеги (див. рис. 2.8) такі як об'єкти і прив'язки до координат, в реальному часі для невідомих або не деталізовані ділянках Землі.

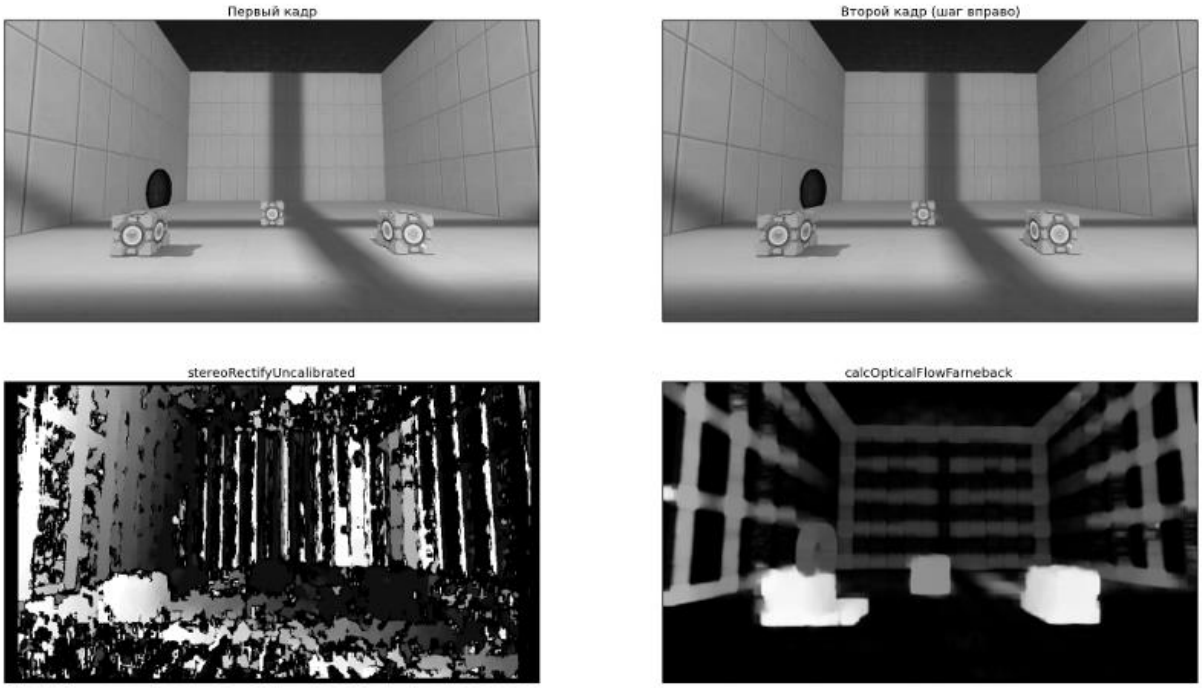


Рисунок 2.7 – Покрокове отримання результату

Рисунок 2.8 – Приклад метатегів на мапі

Таким чином, реалізувавши два відмінних рішення, для основних проблем в цій галузі, і з'єднати їх для парної і поетапної обробки зображень, з'являється можливість побудувати швидку і максимально точну систему, за допомогою якої можливо буде будувати карти включає в себе, мета теги такі як об'єкти і прив'язки до координат, в реальному часі для невідомих або не деталізовані ділянках Землі.

Таким чином ми маємо готовий проект матиме можливість створювати карти в реальному часі і знаходити запитувані об'єкти на поверхні і відзначати їх на карті. Також за допомогою використання нейронних продукт буде навчатись на опрацьованих об’єктах та з кожним разом вдосконалювати свої навички, чим самим забезпечувати біль стабільну та точнішу роботу.

3 РОЗРОБКА МАТЕМАТИЧНОЇ МОДЕЛІ ТА МЕТОДІВ РІШЕННЯ ЗАДАЧ

3.1 Математична модель формування зображень

Для створення оптимального алгоритму об'єднання вихідних фотографій карт місцевості, необхідно виділити, ключові місця, точки в кожному з поєднуваних зображень, для уточнення в процесі комбінації, точок на зображеннях, за якими буде проводитися зв'язка областей.

Для цих цілей був обраний алгоритм, виділення контурів зображень. Методи ґрунтуються на одному з базових властивостей сигналу яскравості - розривності. Найбільш загальним способом пошуку розривів є обробка зображення за допомогою ковзної маски, яку називають також фільтром, ядром, вікном або шаблоном, яка представляє собою якусь квадратну матрицю, відповідну зазначеній групі пікселів вихідного зображення. Елементи матриці прийнято називати коефіцієнтами. Оперування такою матрицею в будь-яких локальних перетвореннях називається фільтрацією або просторової фільтрацією (див. рис. 3.1).

Контроль здійснюється шляхом простому переміщенні маски фільтра від точки до точки зображення; в кожній точці (x, y) відгук фільтра обчислюється з використанням попередньо заданих зв'язків. У разі лінійної просторової фільтрації відгук задається сумою добутку коефіцієнтів фільтра на відповідні значення пікселів в області, покритої маскою фільтра.

Для маски 3x3 елемента, показаної на рисунку 3.1, результат (відгук) R лінійної фільтрації в точці (x, y) зображення складе:

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \quad (3.1)$$

що, як видно, є сума добутків коефіцієнтів маски на значення пікселів безпосередньо під маскою. Зокрема зауважимо, що коефіцієнт *w(0,0)* варто при значенні *f(x,y)*, вказуючи тим самим, що маска центрована в точці *(x,y)*.

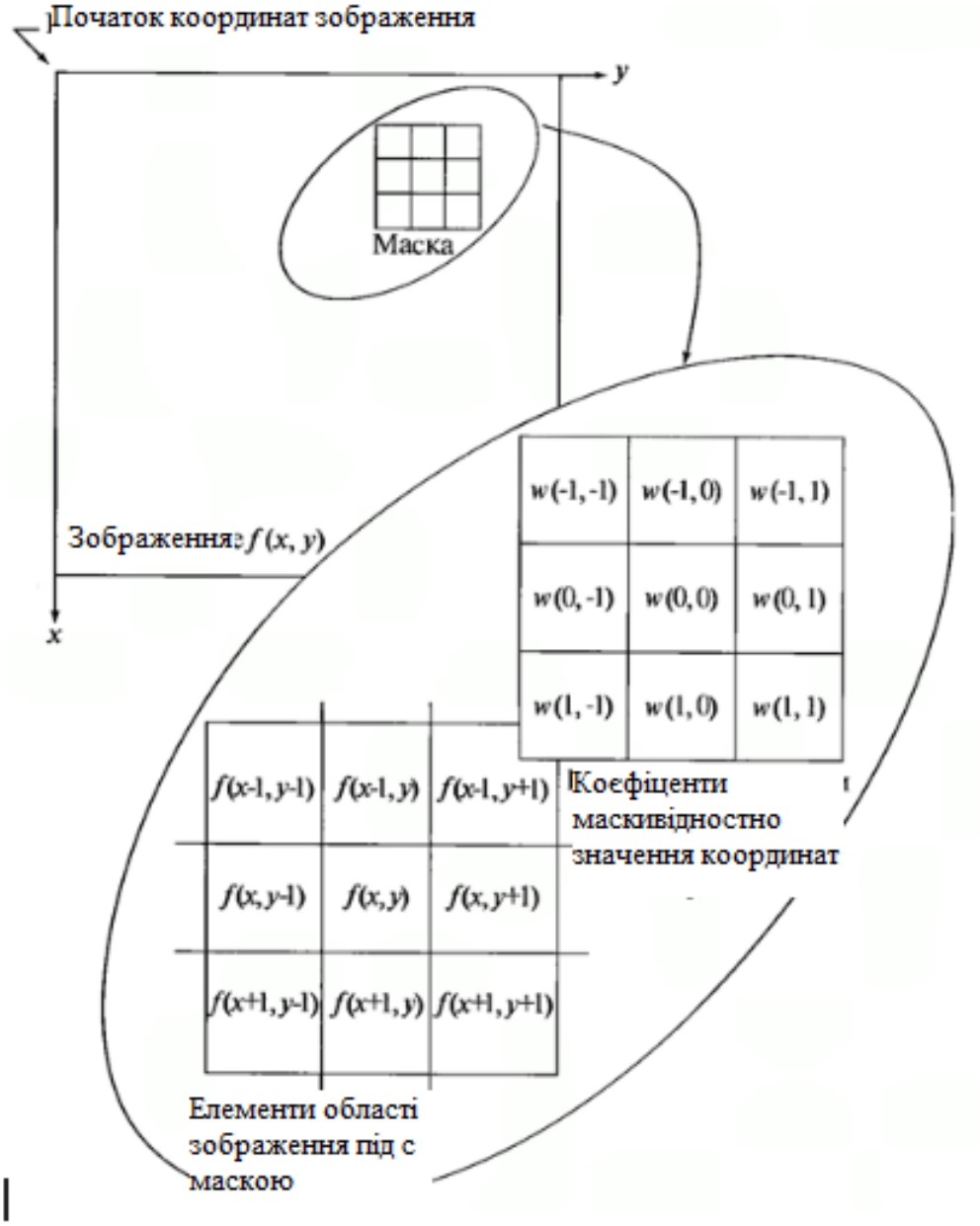


Рисунок 3.1 – Схема просторової фільтрації

При виявленні перепадів яскравості використовуються дискретні аналоги похідних першого і другого порядку. Для простоти викладу будуть розглянуті одновимірні похідні.

Перша похідна одновимірної функції *f (x)* визначається як різниця значень сусідніх елементів:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{3.2}$$

Тут використана запис у вигляді похідної для того, щоб зберегти ті ж позначення в разі двох змінних *f (x, y)*, де доведеться мати справу з приватними похідними по двом просторовим осях. Використання приватної похідною не змінює істоти розгляду.

Аналогічно, друга похідна визначається як різниця сусідніх значень першої похідної:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{3.3}$$

Обчислення першої похідної цифрового зображення засноване на різних дискретних наближеннях двовимірного градієнта. За визначенням, градієнт зображення *f (x, y)* в точці *(x, y)* - це вектор [14]:

$$\nabla f = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.4}$$

Як відомо з курсу математичного аналізу, напрямок вектора градієнта збігається з напрямком максимальної швидкості зміни функції *f* в точці *(x, y)* [15].

Важливу роль при виявленні контурів грає модуль цього вектора, який позначається ∇f і дорівнює:

$$\nabla f = |\nabla f| = \sqrt{G_x^2 + G_y^2} \tag{3.5}$$

Ця величина дорівнює значенню максимальної швидкості зміни функції *f* в точці *(x, y)*, причому максимум досягається в напрямку вектора ∇f. Величину ∇f також часто називають градієнтом.

Напрямок вектора градієнта також є важливою характеристикою. Позначимо *α (x, y)* кут між напрямком вектора ∇f в точці *(x, y)* і віссю x. Як відомо з математичного аналізу [15],

$$\alpha(x, y) = arctg\left(\frac{Gy}{Gx}\right) \tag{3.6}$$

Звідси легко знайти напрям контуру в точці *(x, y)*, яке перпендикулярно напрямку вектора градієнта в цій точці. А обчислити градієнт зображення можна, обчисливши величини приватних похідних $\partial f/\partial x$ і $\partial f/\partial y$ для кожної точки.

3.1.1 Оператор Робертса

Нехай область 3x3, показана на рисунку нижче (див. рис. 3.2), являє собою значення яскравості в околиці деякого елемента зображення.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Рисунок 3.2 – Матриця 3x3 всередині зображення

Один з найпростіших способів знаходження перших приватних похідних в точці складається в застосуванні наступного перехресного градиентного оператора Робертса [16]:

$$G_x = (z_9 - z_5) \tag{3.7}$$

та

$$G_y = (z_8 - z_6) \tag{3.8}$$

Ці похідні можуть бути реалізовані шляхом обробки всього зображення за допомогою оператора, описуваного масками на рисунку 3.3, використовуючи процедуру фільтрації, описану раніше.

-1	0
0	1

0	-1
1	0

Рисунок 3.3 – Маски оператора Робертса

Реалізація масок розмірами 2x2 не надто зручні, тому що у них немає чітко вираженого центрального елемента, що істотно відбивається на результаті виконання фільтрації. Але цей «мінус» породжує дуже корисна властивість даного алгоритму – високу швидкість обробки зображення.

3.1.2 Оператор Превітта

Оператор Превітта, так само як і оператор Робертса, оперує з областю зображення 3x3, представленої на рисунку 3.2, тільки використання такої маски задається іншими виразами:

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{3.9}$$

та

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{3.10}$$

У цих формулах різниця між сумами по верхній і нижній рядкам околиці 3x3 є наближеним значенням похідної по осі x, а різниця між сумами по першому і останньому стовпцям цієї матриці – похідній по осі y. Для реалізації цих формул використовується оператор, що описується масками на рисунку 3.4, який називається оператором Превітта.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Рисунок 3.4 – Маски оператора Превітта

3.1.3 Оператор Собеля

Оператор Собеля теж використовує область зображення 3x3, відображену на рисунку 3.2. Він досить схожий на оператор Превітта, а

видозміна полягає в використанні вагового коефіцієнта 2 для середніх елементів:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{3.11}$$

та

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{3.12}$$

Це збільшене значення використовується для зменшення ефекту згладжування за рахунок надання більшої ваги середніх точок.

Маски, використовувані оператором Собеля, відображені на малюнку нижче (див. рис. 3.5).

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Рисунок 3.5 – Маски оператора Собеля

Розглянуті вище маски застосовуються для отримання складових градієнта G_x и G_y. Для обчислення величини градієнта ці складові необхідно використовувати спільно:

$$\nabla f \approx |G_x| + |G_y| \tag{3.14}$$

чи

$$f = \sqrt{G_x^2 + G_y^2} \tag{3.15}$$

4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1 Схеми алгоритми розв'язання задач

Так як, основне ядро системи - це послідовність проведених операцій над вхідними зображеннями, враховуючи операції які виконуються на стороні нейронної мережі, дуже важким і обов’язковим є підготовка і опрацювання над зображенням до роботи з нейронною мережею.

Тож першою задачею є перетворення зображення зробленого у плоский виді, до зображення що не містить помилок виду та ракурсу, а саме перспективи погрішності, для чого був використаний алгоритм пошуку кутів. Для цього було прийняте рішення застосувати метод знаходження особливих точок Харріса та дескриптор Surf.

4.1.1 Метод знаходження особливих точок Харріса

За результатами досліджень, одним з найоптимальніших детекторів кутів є дуже популярний метод Харріса. Цей метод – поліпшення методу Моравеца, автори розглядають похідні зміни яскравості зображення для дослідження змін яскравості по усіх напрямах [17].

Для зображення I розглянемо деяке вікно W (як правило, розмір вікна 5x5, але може залежати від розміру зображення) з центром в точці (x,y), а також його зрушення на (u,v) (див. рис. 4.1).

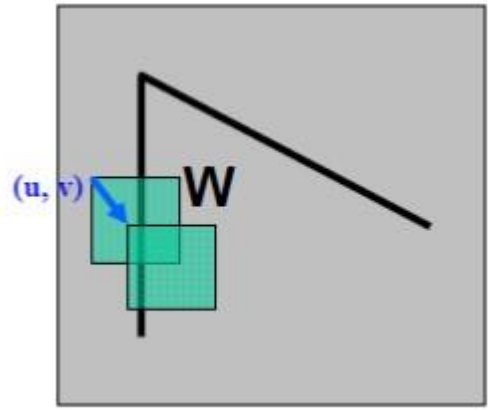
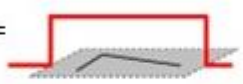
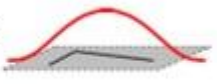


Рисунок 4.1 – Вікно з центром в (x, y) і його зсув на (u, v)

Сума квадратів різниць між вихідним і висунутим вікном буде дорівнює:

$$E(u,v) = \sum_{(x,y) \in W} w(x,y)(I(x+u,y+v) - I(x,y))^2 \approx \sum_{(x,y) \in W} w(x,y)(I_x(x,y)u + I_y(x,y)v)^2 \approx (x\ y)M\begin{pmatrix}x\\y\end{pmatrix}, \quad (4.1)$$

w(x,y) - вагова функція:

$w(x,y) =$  или 

1 – в окне, 0 – вне окна Функция Гаусса

M - автокореляційна матриця:

$$M = \sum_{(u,v) \in W} w(u,v)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \quad (4.2)$$

При великих змінах функції E (x, y) по всіляких напрямках (x, y) визначається кут, отже виходять великі по модулю власні значення матриці M. Нижче наведено розташування власних значень (див. рис. 4.2).

У зв'язку з трудомісткістю обчислення власних чисел матриці Харріс і Стефан запропонували ввести міру відгуку: R = detM - k (trM) 2> k, де k - емпірична константа, k [0,04; 0,06]. В такому випадку, значення R буде

позитивно для кутових особливих точок. Потім прибираються точки, які менше деякого порога R (ті, у яких значення R менше будь-якого порога). Далі обчислюються локальні максимуми функції R в околиці заданого радіусу і вибираються в якості кутових особливих точок [18].

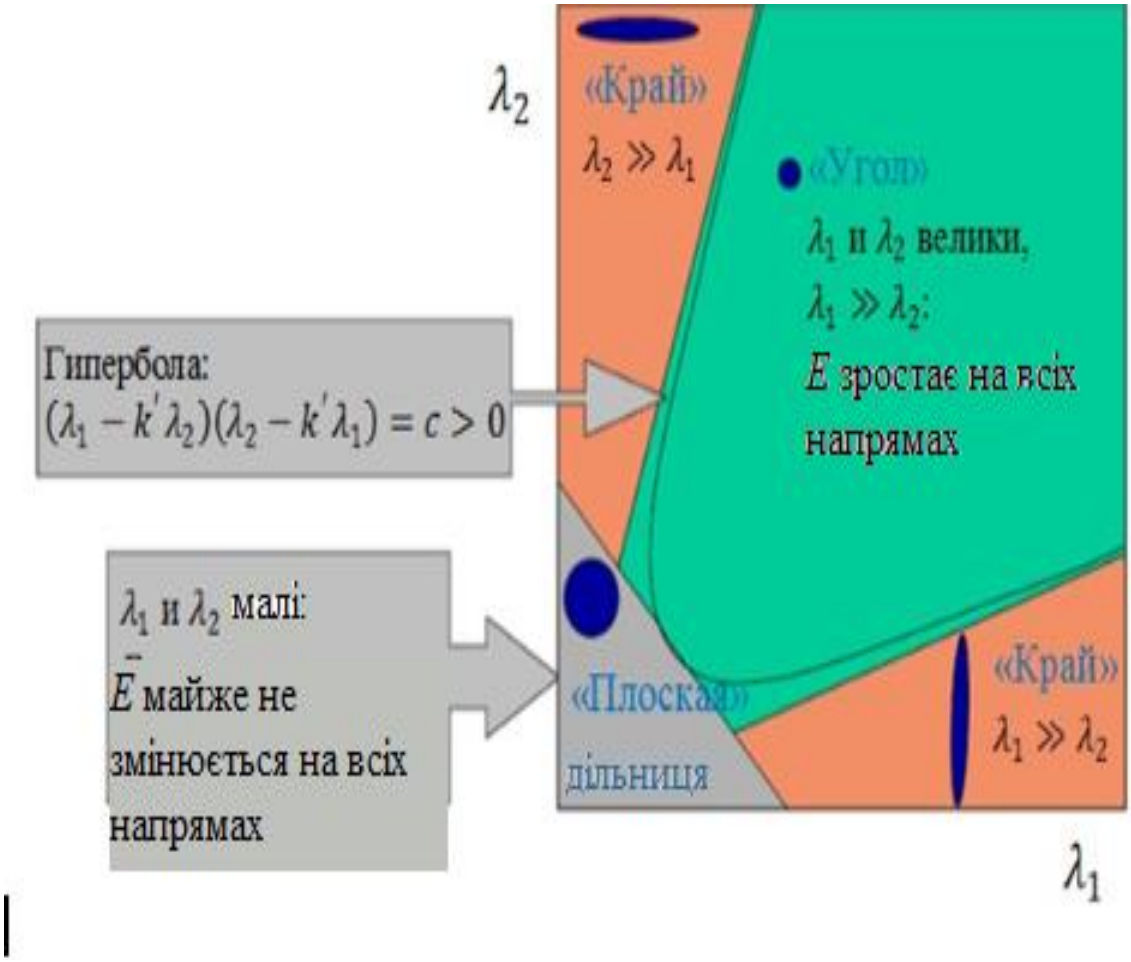


Рисунок 4.2 – Розташування власних значень

Метод Харріса інваріантний до обертання, а також до Афін змін яскравості. Однак він чутливий до шумів і залежить від масштабу зображення (в іншому випадку використовують багатомасштабного метод Харріса (multi-scale Harris detector)).

4.1.2 Дескриптор Surf

Алгоритм побудови дескриптора наступний. Вокруг точки будується квадратна околиця розміром, де - масштаб, на якому отримано максимальне значення детермінанта матриці Гессе [18].

Отримана квадратна область розбивається на блоки, в результаті область буде розбита на 4x4 регіони (див. рис. 4.3).

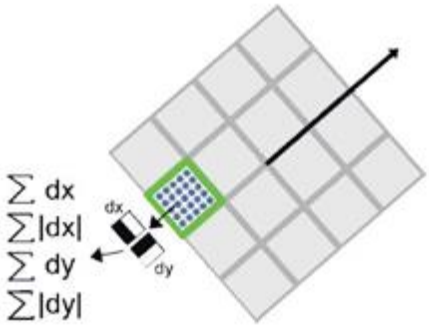


Рисунок 4.3 – Область, що розбита на 4x4 регіону

Для кожного блоку обчислюються більш прості ознаки. Як наслідок, виходить вектор, що містить 4 компоненти: 2 – це сумарний градієнт по квадранту, 2 – сума модулів точкових градієнтів (рис.4.4).

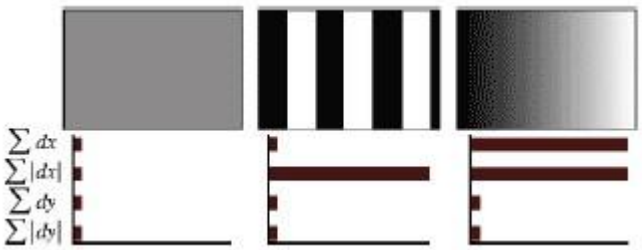


Рисунок 4.4 – Поведінка величин сум для різних ділянок зображень

Дескриптор формується в результаті склеювання зважених описів градієнта для 16 квадрантів навколо особливої точки. Елементи дескриптора зважуються на коефіцієнти Гауссова ядра. Ваги необхідні для більшої стійкості до шумів у віддалених точках.

Додатково до дескриптора заноситься слід матриці Гессе. Ці компоненти необхідні, щоб розрізняти темні і світлі плями. Для світлих точок на темному тлі слід негативний, для темних точок на світлому фоні - позитивний.

Метод SURF також використовується для пошуку об'єктів. Проте, дескриптор ніяк не використовує інформацію про об'єкти. SURF розглядає зображення як єдине ціле і виділяє особливості всього зображення, тому він погано працює з об'єктами простої форми.

4.2 Побудова моделі даних

Вибір типу СУБД. Документально-орієнтована СУБД - це СУБД, спеціально розроблена для зберігання ієрархічних структур даних (документів) і зазвичай реалізована з використанням підходу NoSQL. Основою документально-орієнтованої СУБД є репозиторії документів, що мають деревоподібну структуру (іноді ліси). Деревовидна структура починається з кореневого вузла і може зберігати кілька внутрішніх і листових вузлів.

Документи можуть бути організовані (згруповані) в колекції. Вони можуть розглядатися як віддалені аналогові таблиці реляційних СУБД, але колекції можуть містити інші колекції.

Прикладами СУБД цього типу є CouchDB, Couchbase, MarkLogic, MongoDB, eXist, Berkeley DB [19].

Реляційна СУБД являє собою набір взаємопов'язаних таблиць, кожна з яких містить інформацію про об'єкти певного типу. Запити на такі бази даних повертають таблицю, яка може знову брати участь у наступному запиті. Дані в деяких таблицях, як правило, пов'язані з даними з інших таблиць, звідки походить назва “relational”.

Прикладами СУБД цього типу є MsSQL, SQLite, MySQL, PostgreSQL.

Для вирішення цієї проблеми було обрано реляційну СУБД, адже її головною перевагою є простота її елементарних об'єктів – таблиці (відносини), а після цього – побудова математичного прозорого маніпуляційного методу – реляційної алгебри. Крім того, реляційна модель даних переживає нормалізацію, щоб запобігти дублюванню даних, на відміну від документально-орієнтованої СУБД [19].

4.2.1 Побудова схеми моделі даних

Модель Entity-Relationship (Entity-Relationship Model, ERM, ER-model) дозволяє описати концептуальну схему предметної області при проектуванні баз даних (див. рис. 4.5). Це допомагає побачити основну сутність предметної області, атрибути сутності та відносини між ними [20].

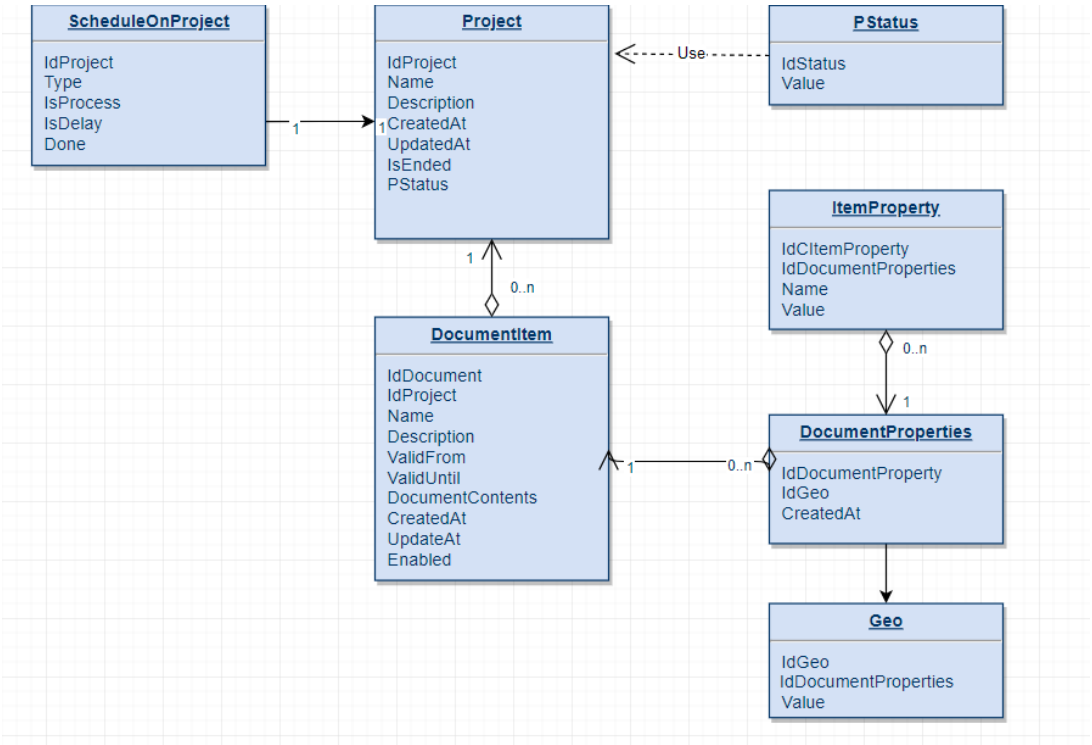


Рисунок 4.5 – Логічна схема даних

Опис типів сутностей. Усі сутності системи представлені у таблиці 4.1.

Таблиця 4.1 – Типи сутностей

Назва сутності	Опис
Project	Описує проект та його сорти
PStatus	Описує стан проекту
DocumentItem	Документи проекту – зображення
DocumentProperties	Мета інформація зображень
ItemProperty	Будь-які важливі дані кожного з зображень
Geo	Описує локацію та відносний периметр
ScheduleOnProject	Статуси та трекінг опрацюванням проектів

Опис атрибутів сутностей наведено в таблиці 4.2.

4.2.2 Огляд та обґрунтування вибору СКБД.

Сервер QL також відомий як Microsoft SQL Server, який з'явився раніше, ніж MySQL [21].

Microsoft розробила сервер SQL в 80-х роках з обіцянкою розробити надійну і масштабну реляційну базу даних. Вони залишаються невід'ємною частиною SQL Server, який закінчується всі ці роки, і надають незалежне рішення для великого корпоративного програмного забезпечення.

SQL-сервер більше підходить для розробників, які використовують .NET мовою розробки. Як був обраний основна мова розробників. SQL Server характеризується наступними особливостями:

– Продуктивність. SQL Server працює дуже швидко;

– Надійність і безпеку. SQL Server надає шифрування даних;

– Простота. З даної СУБД відносно легко працювати і вести адміністрування.

Розглянемо основні переваги обраного рішення:

Незалежність від конкретної СУБД. Незважаючи на наявність діалектів і відмінностей в синтаксисі, в більшості життів без текстових SQL-запитів, що містять DDL і DML, їх можна досить легко перенести з однієї СУБД в іншу.

Є системи, на які розробник спочатку орієнтувався! Застосовуючи хоча б кілька СУБД. Природно, при застосуванні деяких спецпропозицій для реалізації такої толерантності домогтися вже дуже складно;

Таблиця 4.2 – Опис атрибутів

Тип сутності	Атрибут	Тип даних,довжина	Обмеж.	Допустим. Null
Project	idProject	INT	Перв.ключ	Ні
	Name	VARCHAR(255)	–	Так
Project	Description	VARCHAR(255)	–	Так
	CreatedAt	DATETIME	–	Ні
	UpdatedAt	DATETIME	–	Так
	IsEnded	BYTE	–	Ні
	PStatus	VARCHAR(255)	–	Ні
PStatus	idStatus	INT	Перв.ключ	Ні
	Value	VARCHAR(255)	–	Ні
DocumentItem	IdDocument	INT	Перв.ключ	Ні
	IdProject	INT	Зовн.ключ	Ні
	Description	VARCHAR(255)	–	Так
	Name	VARCHAR(255)	–	Так
DocumentItem	DocumentContents	VARCHAR(255)	–	Так
	CreatedAt	DATETIME	–	Ні
	UpdateAt	DATETIME	–	Так
	Enabled	BYTE	–	Ні
DocumentProperties	IdDocumentProperty	INT	Перв.ключ	Ні
	IdGeo	INT	Зовн.ключ	Ні
	CreatedAt	DATETIME	–	Так
ItemProperty	IdCItemProperty	INT	Перв.ключ	Ні
	IdDocumentProperties	INT	Зовн.ключ	Ні
	Name	VARCHAR(255)	–	Ні
	Value	VARCHAR(255)	–	Так
Geo	IdGeo	INT	Перв.ключ	Ні
	IdDocumentProperties	INT	Зовн.ключ	Ні
	Value	VARCHAR(255)	–	Так
ScheduleOnProject	IdProject	INT	Зовн.ключ	Ні
	Type	VARCHAR(255)	–	Ні
	IsProcess	BYTE	–	Ні
	IsDelay	BYTE	–	Ні
	Done	BYTE	–	Ні

Наявність стандартів. Наявність стандартів і набір тестів для виявлення сумісності і відповідності конкретній реалізації SQL стандартному стандарту тільки просторової «стабілізації» мови. Правда, варто намалювати Рамуса, що еталон сам по собі є місцем занадто формалізованим і збільшеним за розміром [21].

Декларативна. За допомогою SQL програміст перевіряє тільки ті дані, які необхідно розтягнути або змінити. Ті, як це зробити, вирішує СУБД безпосередньо при обробці запиту SQL.

Однак не варто думати, що це абсолютно універсальний принцип - програміст запускає набір даних для вибірки або модифікації, проти нього корисно уявити, як СУБД буде сортувати текст свого запиту. Чим складніше складений запит, тим більше він допускає варіанти написання, різні по продуктивності, але однакові в кінцевому наборі даних.

Основні недоліки можуть бути відхилені:

– Невідповідність реляційної моделі даних. Творці реляційної моделі даних Едгар Кодд, Крістофер Смерть і їх прихильники вказують, що SQL НЕ є дійсно реляційним мовою. Зокрема, я можу вказати на наступні проблеми SQL:

а) Дублюються рядки;

б) Невизначені значення;

в) Очевидне вказівку порядку стовпців зліва направо;

г) Безіменні стовпці і повторювані імена стовпців;

ґ) Відсутність властивостей підтримки "=";

е) Використання покажчиків;

є) Висока надмірність;

– складність. Хоча SQL був об'єднаний як засіб роботи з кінцевим користувачем, врешті-решт він став настільки складним, що перетворився в інструмент для програмістів;

– Відхилення від стандартів. Незважаючи на наявність стандарту ANSI SQL-92, стандарт ANSI, багато компаній, які розробляють СУБД, вносять зміни в мову SQL, що використовується в розробляється СУБД, так само відходячи від стандарту. Таким чином, існують конкретні діалекти SQL для кожної конкретної СУБД;

– складність роботи з ієрархічними структурами. Раніше діалекти SQL більшості СУБД не пропонували способу маніпулювання деревоподібними структурами. Деякі постачальники СУБД запропонували своє рішення. На сьогоднішній день ANSI стандартизовано рекурсивную конструкцію CON з діалектом DB2 SQL. В MS SQL Server рекурсивні пропозиції з'явилися тільки у версії MS SQL Server 2005.

4.3 Вибір і опис середовища розробки

В якості основного середовища було обрано продукт MicrosoftVisualStudio, лінійку продуктів Microsoft, що включає інтегровану середу розробки програмного забезпечення та ряд інших інструментів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, включаючи підтримку технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як у власних, так і в керованих кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile , Windows CE ,. Чисті рамки. NETCompactFramework та Microsoft Silverlight [22].

VisualStudio включає в себе редактор вихідних кодів з підтримкою технології IntelliSense і можливість легко рефакторировать код. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і

відладчик рівня машини. Решта вбудованих інструментів включають редактор форм для спрощення створення графічного інтерфейсу користувача, веб-редактора, дизайнера класів і дизайнера схем бази даних. Visual Studio дозволяє створювати та підключати програми сторонніх виробників (плагінів) для розширення функціональних можливостей практично на всіх рівнях, включаючи додавання підтримки систем керування версіями вихідного коду (наприклад, Subversion та Visual SourceSafe), додавання нових інструментів (наприклад, , редагування та візуальний дизайн коду на мовах програмування, що стосуються домену, або інструментів для інших аспектів циклу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server). Visual Studio 2010 (код і назва Гаваї, для Ultimate - Rosario, внутрішня версія 10.0) - випущена 12 квітня 2010 року .NET Framework 4.0 Visual Studio включає підтримку C # 4.0 і Visual Basic .NET 10.0, а також F#, яка не була доступна в попередніх версіях.

4.4 Вибір мови програмування і шаблону проектування

Перш ніж приступити до розробки інтелектуального модуля управління дроном, вам потрібно вибрати мову програмування, на якому він буде написаний.

C ++ - це мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованого, узагальненого і процедурного. Мова використовується для системного програмування, розробки програмного забезпечення, написання драйверів, потужних

серверних і клієнтських додатків, а також для розробки розважальних програм, таких як відеоігри [23].

C # - це об'єктно-орієнтована мова програмування з безпечною системою введення для платформи .NET.

Перевага цієї мови в тому, що розмір одержуваних програмних продуктів недостатньо великий і не вимагає додаткових бібліотек.

Синтаксис C # близький до C ++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на члени класу функцій, атрибути, події, властивості, винятки, коментарі у форматі XML. Використання багатьох з його попередників - таких як C ++, Delphi, Module і Smalltalk - C #, заснованих на практиці їх використання, виключає деякі моделі, які виявилися проблематичними при розробці програмних систем, такі як класи множинного спадкоємства (на відміну від C + +).

Java – це об'єктно-орієнтована мова програмування, який спочатку розробляється для побутової електроніки, але пізніше став використовуватися для написання аплетів, додатків і серверного програмного забезпечення. розроблений в 1995 році компанією Sun Microsystems і спочатку розроблений як об'єктно-орієнтована мова. У цьому він вигідно відрізняється від C ++, який з міркувань сумісності зберігає багато елементів процедурного мови C. Мова запозичив синтаксис з C і C ++.

Зокрема, в якості основи використовувалася об'єктна модель C ++, але нього були внесені зміни. Виключена можливість виникнення деяких конфліктних ситуацій, які могли виникнути через помилки програміста, і полегшений процес розробки об'єктно-орієнтованих програм [23]. Ряд дій, які C / C ++ повинен реалізовувати програмістами, призначається віртуальній машині. Перш за все, Java була розроблена як незалежний від

платформи мова, тому вона має менше апаратних можливостей низького рівня (див. таблицю 4.3).

Таблиця 4.3 – Порівняння сучасних мов програмування

Можливості	C++	C#	Java
Об’єктно-орієнтована	+	+	+
Динамічні масиви	+	+	-
Багатопоточність	+	+	+
Узагальнене програмування	–	+	+
Багатоплатформеність	+	+	+
Статична типізація	+	+	+
Швидкість виконання	+	+	-
Сторонні бібліотеки	+	+	+
Структури даних	+	+	-

Для вибору використовуються такі критерії, як наявність об'єктно-орієнтованої парадигми, реалізація динамічних масивів, застосування багатопоточності, багатоплатформенность, статична типізація і швидкість виконання розробленого модуля, оскільки вони найбільш актуальні при виборі мови.

Для реалізації програми була вибрана мова програмування C #. Найкращим мовою для реалізації консультативної системи є C#, оскільки ця мова дозволить вам швидко і ефективно створити готовий продукт.

4.5 Опис експерименту

Так як розроблена система, є складним рішенням, яке безпосередньо працює з обладнанням наприклад - дрон, який може бути джерелом безперебійного постачання зображень, зроблених у невизначному проміжутку часу то логічним чином було зроблено інтерфейс який дозволяє інтегрувати до самого ядра системи, щоб додати можливість дивитися за процесом оброблення у режимі реального часу.

Саме можливість керувати усім з самописного джерела тестів, який вихідні зображення з різним пресетапом та емуляцією, тільки но надходження зображення та статусів з пов'язаним із ним зображенням модулів.

Тож уся система була покрита тестами, кожній з яких вміє видавати результат закінчення роботи, для всіх нод (етапів) (див. рис. 4.6).

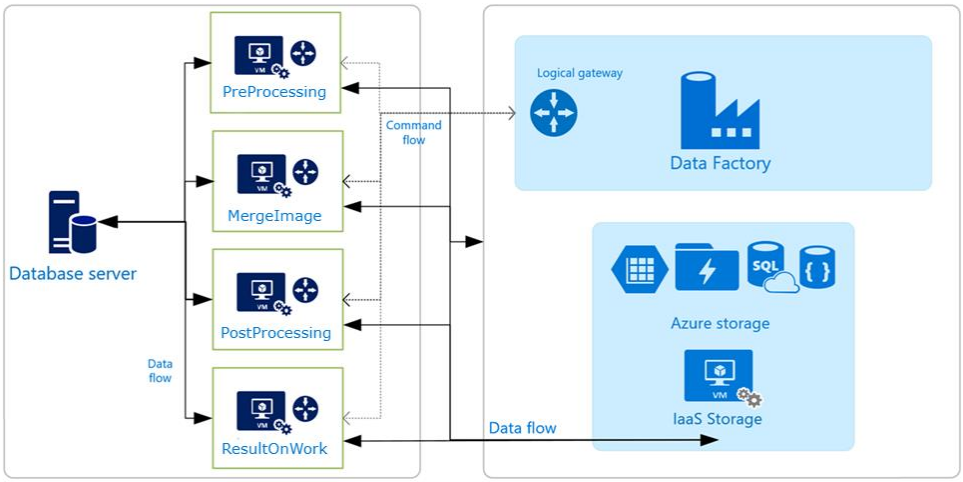


Рисунок 4.6 – Схема єтапів роботи

Наприклад частина тестів котра працює з вхідними зображеннями та виконую предпрацювання, с алгоритмами описаними у розділі 4.1. Кожен з етапів який знаходиться у проміжку працювання, з моменту отримання вхідних даних до остаточного результату також має тести.

Система та тести які дозволяють перевірити роботоздатність усієї системи у цілому (див. рис. 4.7).

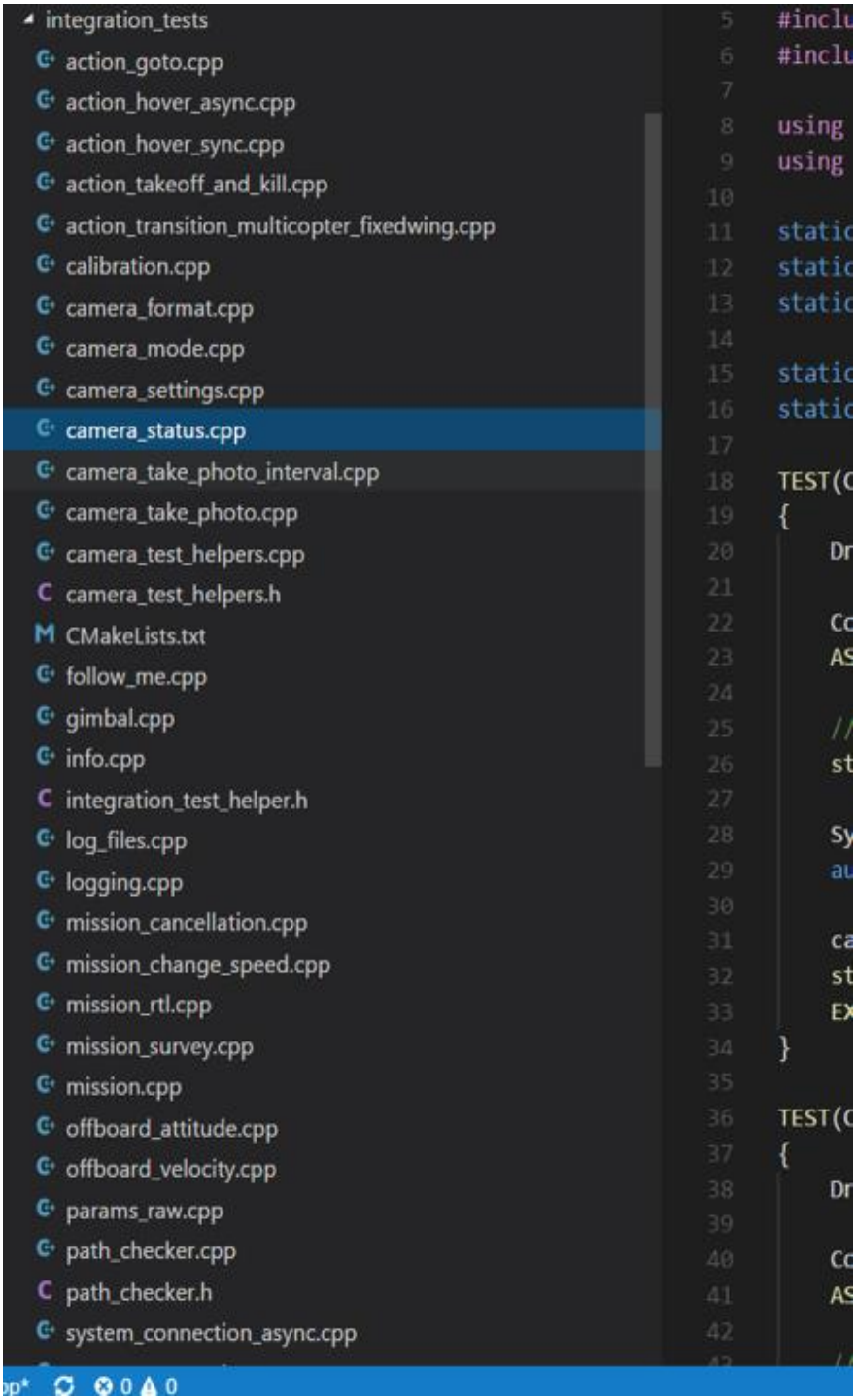


Рисунок 4.7 – Тести інтеграції з камерами та модулями

Для полного опрацювання сценаріев роботи та Перевірки усіх етапів Обробка даних - від Звичайно зображення з метаінформацією з додаткових модулів - ДЖПС, далекомір, та кутомір до готової карти, та усі проміжні стани, по-перше треба описати повну картину етапів опрацювання.

Для полного опрацювання сценаріев роботи та Перевірки усіх етапів Обробка даних - від Звичайно зображення з метаінформацією з додаткових модулів - ДЖПС, далекомір, та кутомір до готової карти, та усі проміжні стани, по-перше треба описати повну картину етапів опрацювання (див. рис. 4.8).

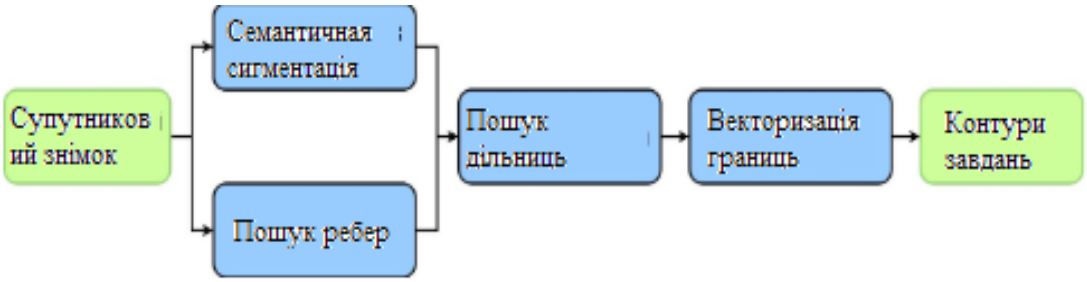


Рисунок 4.8 – Послідовність виконання етапів

Після етапі предопрацювання що досконало описанні у розділі 4.1 ми отримуємо результат у вигляді монохромного зображення котре є вже складним між собою та проте ми його бачимо не кольоровим, и прикладі ми також розглядати ситуацію коли в нас немає одного з зображень, на мапі - рожевий прямокутник (див. рис. 4.9).

Наступний крок - це отримання роздільних частин та повернення вхідних кольорових зображень, зклейкою по їх монохромним відзеркаленням. Результат закінчення другого етапу (без повернення кольору для наглядності) (див. рис. 10).

Наступний крок та самий найважливіший і складний – це опрацювання зображень за допомогою нейронної мережі.

Наша система завдяки написанні на .Net Core, та скомпільовано у нативний код, може виконуватися як на потужних машинах так і на мікроконтролерах у режимі реального часу наприклад на дроні, тож у системі є конфігурація яка дозволяє працювати без важкого та енерго неефективного етапу - а саме третього - опрацювання нейронною мережею. Тож тести також вміють тестувати оби дві умови.

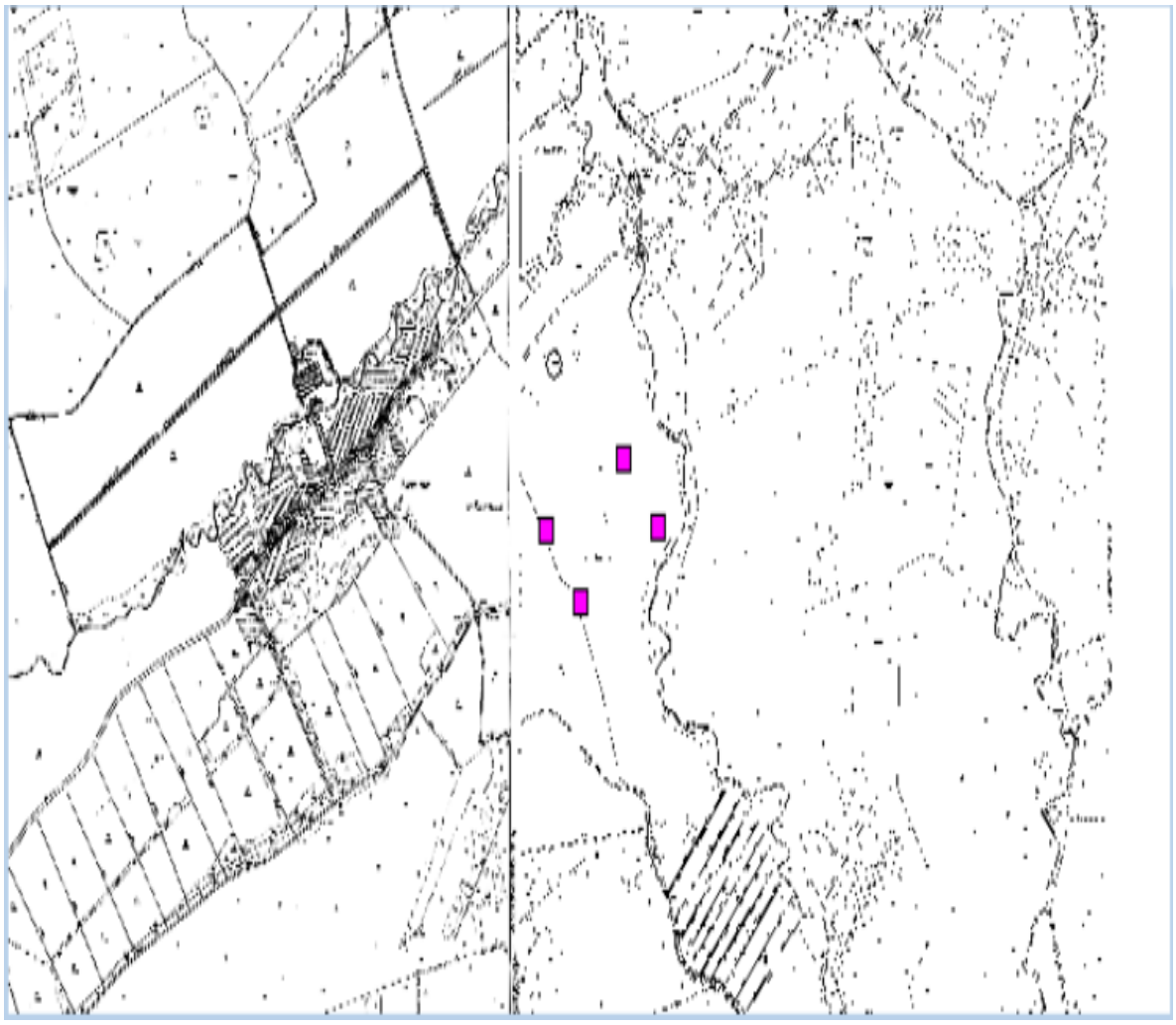


Рисунок 4.9 – Результат першого єтапу

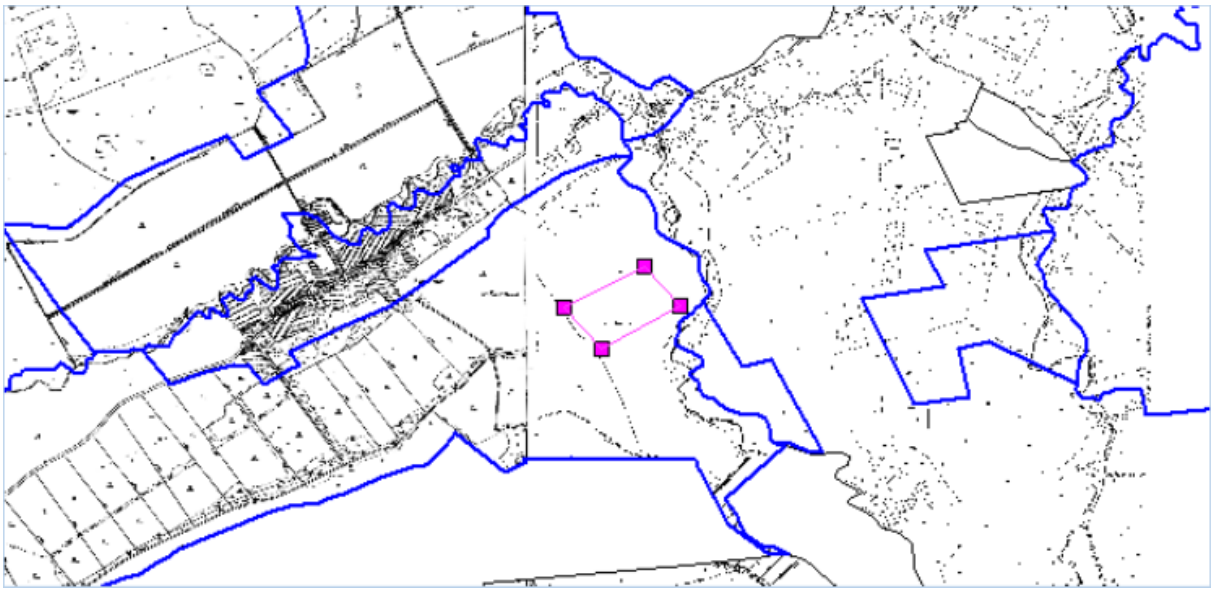


Рисунок 4.10 – Виділенні ділянки

Наша система завдяки написанні на .Net Core, та скомпільовано у нативний код, може виконуватися як на потужних машинах так і на мікроконтролерах у режимі реального часу наприклад на дроні, тож у системі е конфігурація яка дозволяє працювати без важкого та енерго неефективного етапу - а саме третього - опрацювання нейронною мережею. Тож тести також вміють тестувати обидві умови. Тому для демонстрації наступного кроку я обрала другий сценарій який усю міць проганяє нейронну мережу через зображення, задля пошуку об'єктів на них та може бути оброблений лише на потужній машині. Виходячи з цього я обрала зображення з Googde maps - за для демонстрації третього етапу, з насамперед вивчену мережею на хатинки.

Раз вже ми залізли в область комп'ютерного зору, то в першу чергу випробували підхід, релевантний цього самого комп'ютерного зору (див. рис. 4.11).

Спочатку векторна карта растеризуются (багатокутники будиночків, які представлені білими лініями на чорному фоні), фільтр Собеля виділяє ребра на супутниковому знімку. А потім знаходиться зміщення двох

зображень відносно один одного, яке максимізує кореляцію між ними. Ребра після фільтру Собеля досить гучні, тому, якщо застосовувати даний підхід до одного будинку, не завжди виходить прийнятний результат. Однак метод добре працює на територіях з будівлями однакової висоти: якщо шукати зміщення відразу великим ділянці зображення, результат буде більш стабільний.

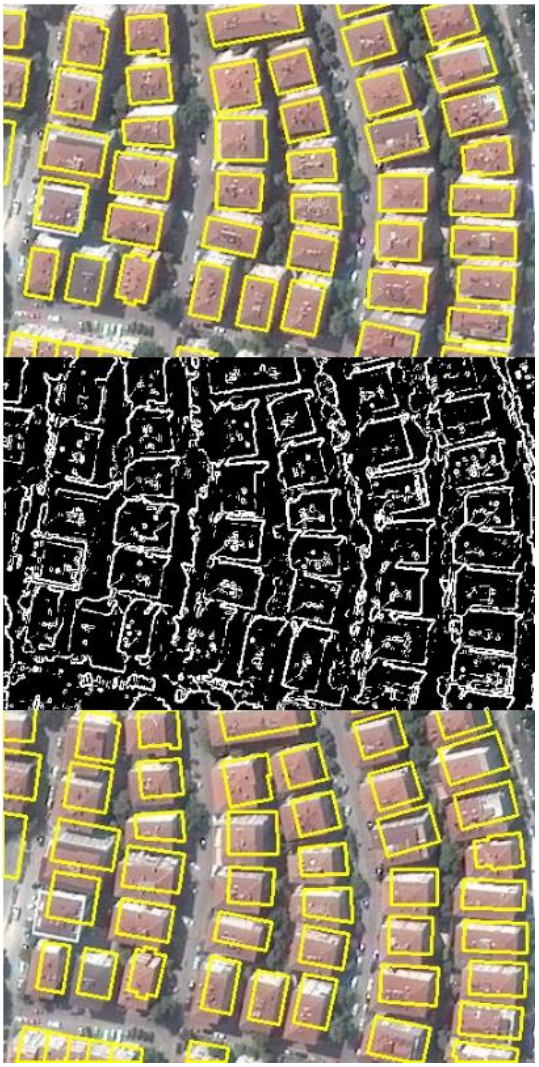


Рисунок 4.11 – «Геометричний» підхід

Якщо територія забудована не однотипні, а різноманітними будинками, попередній метод не підійде. На щастя, іноді ми знаємо висоту

будівель на векторній карті Гугла і положення супутника під час зйомки. Таким чином, ми можемо скористатися шкільними знаннями геометрії і порахувати, куди і на яку відстань зрушиться дах щодо фундаменту (див. рис. 4.12). Цей спосіб дозволив поліпшити датасета на територіях з висотними будівлями (див. рис. 4.13).

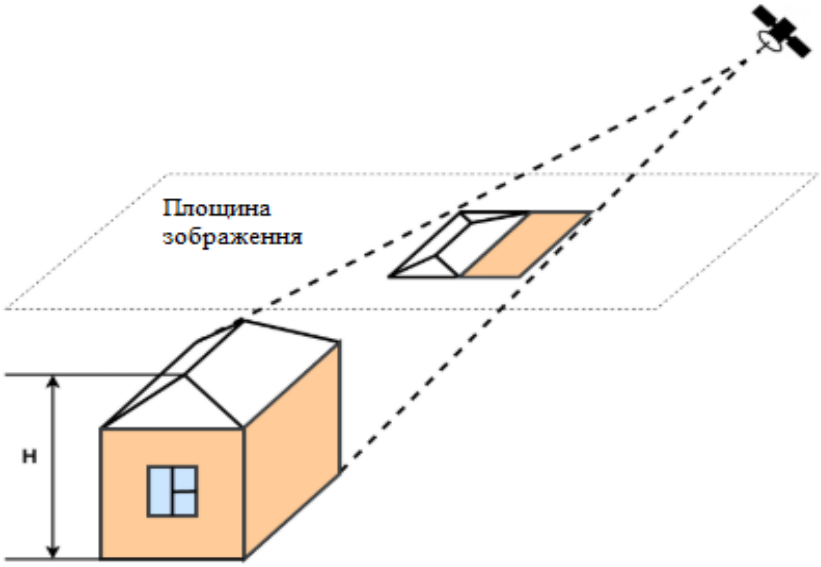


Рисунок 4.12 – Геометрична модель

Рисунок 4.13 – «Ручний» підхід

Самий трудомісткий спосіб: засукати рукава, расчехлить мишку, уп'ястися в монітор і вручну зрушити векторну розмітку будиночків від

фундаментів до дахів. Методика приносить просто приголомшливий за якістю результат, але застосовувати її у великих кількостях не рекомендується: розробники, зайняті такими завданнями, швидко впадають в апатію і втрачають інтерес до життя.

4.5.1 Застосування нейронної мережі

У підсумку ми отримали досить супутникових знімків, добре розмічених по дахах. А значить, з'явився шанс натренувати нейронну мережу (поки, щоправда, не для сегментації, а для поліпшення розмітки інших супутникових знімків). І ми це зробили.

Вхідними даними згортаними нейронної мережі були супутниковий знімок і зміщена растрування розмітка. На виході отримано двовимірний вектор: зміщення по вертикалі і горизонталі (див. рис. 4.14).

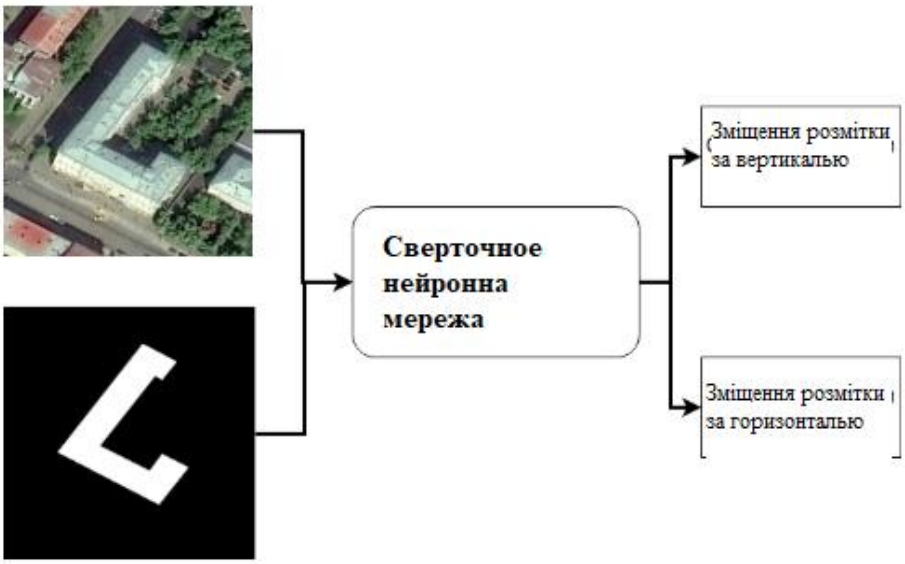


Рисунок 4.14 – Перетворення розмітки

Спочатку векторна карта растеризуются (багатокутники будиночків, які представлені білими лініями на чорному фоні), фільтр Собеля виділяє ребра на супутниковому знімку. А потім знаходиться зміщення двох зображень відносно один одного, яке максимізує кореляцію між ними.

За допомогою нейронної мережі ми знайшли необхідне зміщення, що дозволило добитися добрих результатів на будівлях, у яких не зазначено висота. У підсумку ми значно скоротили виправлення розмітки вручну (рис. 4.15).

Рисунок 4.15 – Різні території – різні будинки

У підсумку значно скорочене виправлення розмітки вручну.

4.5.2 Семантична сегментація

Семантична сегментація - наступний крок у системі, це частина - досліджена задача. Після появи статті Fully Convolutional Networks вона в основному вирішується за допомогою нейронних мереж. Залишається

тільки вибрати мережу (ми розглядали FCN, SegNet і UNet), подумати, чи потрібні нам додаткові хитрощі типу CRF на виході, і визначитися, як і з якою функцією помилки буде проходити навчання (див. рис. 4.16).

У підсумку зупинилися на U-Net-подібної архітектури з узагальненої функцією Intersection Over Union в якості опції помилки. Для тренування нарізали супутникові знімки і відповідну їм розмітку (природно пастеризувати) на квадрати і зібрали в датасета. Вийшло цілком миленько, а іноді так і просто відмінно (рис. 4.17).

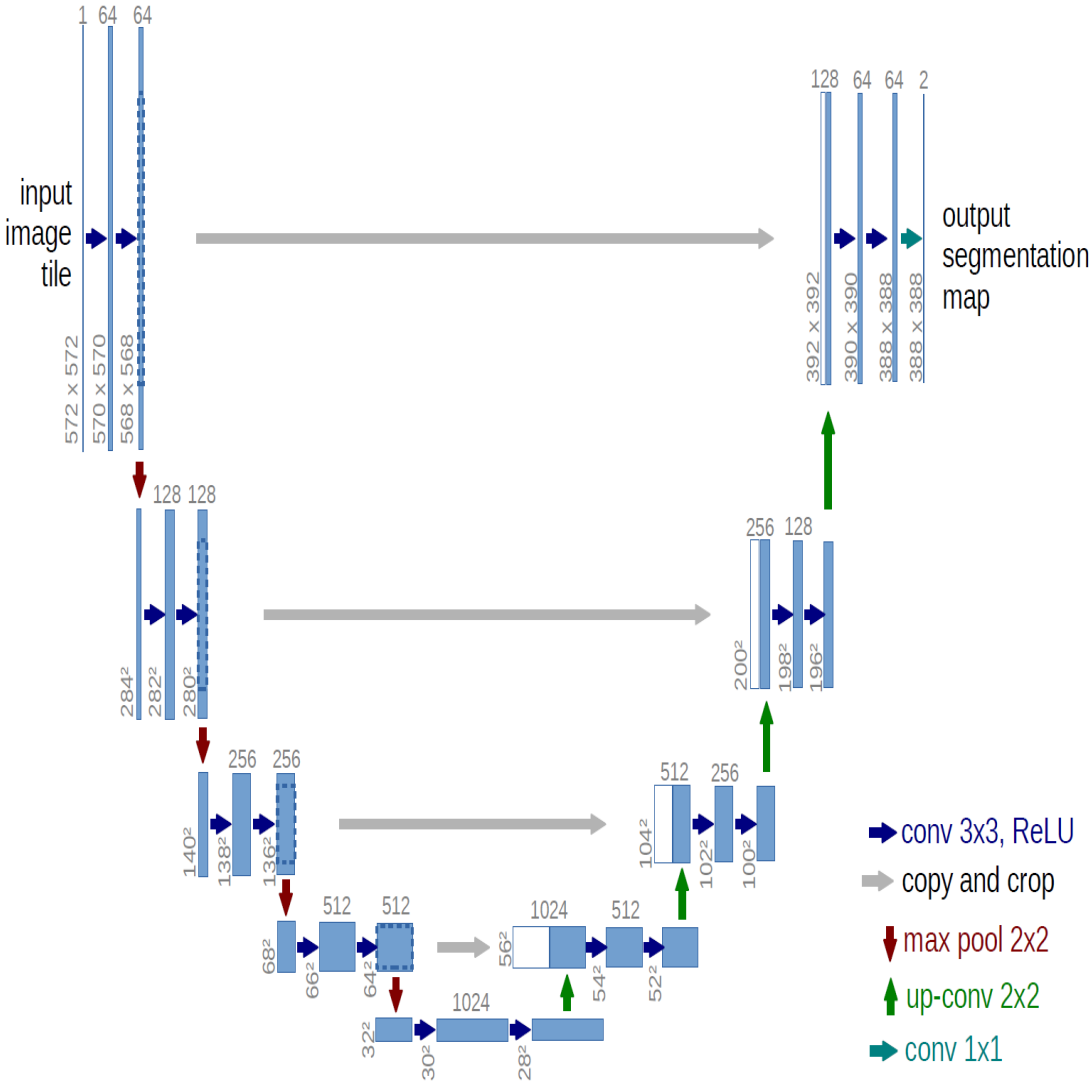


Рисунок 4.16 – Процес семантичної сегментації

Рисунок 4.17 – Результат використання сигментації

На територіях з поодинокими будівлями семантичної сегментації виявилося досить для переходу до наступного етапу - векторизації. Там, де забудова щільна, вдома іноді склеювалися в зв'язну область. Знадобилося розділити їх.

4.5.3 Детектування ребер

Щоб впоратися з цим завданням, можна знайти ребра на зображенні. Для детектування ребер ми вирішили теж навчити мережу (алгоритми пошуку ребер, які не використовують нейронні мережі, явно йдуть в минуле). Тренували мережу типу HED, яка описана в роботі Holistically-

Nested Edge Detection. В оригінальній статті мережу навчалася на наборі даних BSDS-500, в якому на зображеннях розміщені всі ребра. Навчена мережа знаходить все явно виражені ребра: кордону будинків, доріг, озер. Цього вже вистачало, щоб розділити близько розташовані будинки (див. рис. 4.18).

Рисунок 4.18 – Результат використання

Але ми вирішили піти далі і використовувати для навчання той же датасета, що і для семантичної сегментації, тільки при растеризації НЕ зафарбовувати багатокутники будівель цілком, а малювати лише їх межі.

Результат виявився настільки приголомшливо прекрасний, що ми вирішили векторизації будівлі безпосередньо по ребрах, отриманим від мережі.

4.5.4 Детектування вершин

Оскільки мережа типу HED дала відмінний результат на ребрах, ми вирішили натренувати її ж для виявлення вершин. Фактично у нас вийшла мережу з загальними вагами на сверточних шарах. У неї було два виходи одночасно: для ребер і для вершин.

У підсумку ми зробили ще один варіант векторизації будівель, і в деяких випадках він показував цілком осудні результати (див. рис. 4.19).

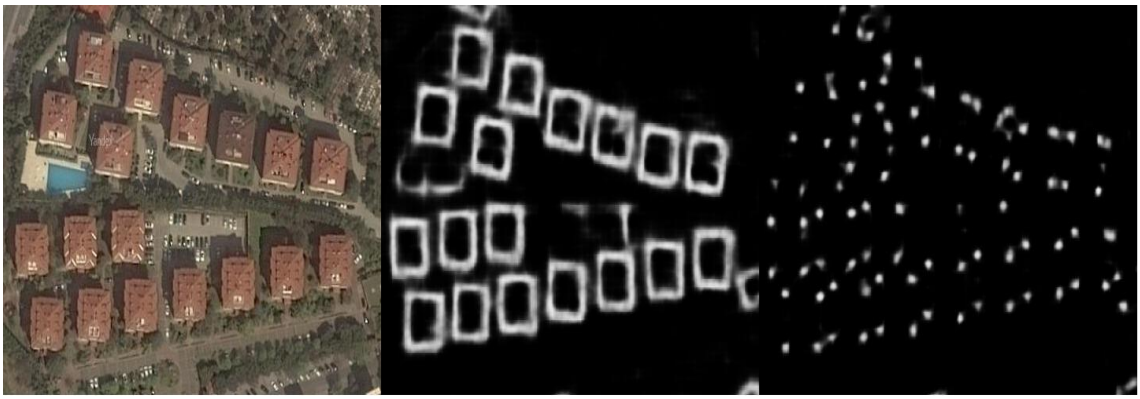


Рисунок 4.19 - Векторизація будинків

В деяких випадках він показував цілком прийнятні результати.

4.5.5 Mask R-CNN

Mask R-CNN - це відносно нове розширення мереж типу Faster R-CNN. Mask R-CNN шукає об'єкти і виділяє для кожного з них маску. В результаті для будинків ми отримаємо не тільки обмежують прямокутники, а й уточнену структуру. Цей підхід вигідно відрізняється від простого

детектування (ми не знаємо, як будівля розташована всередині прямокутника) і від звичайної сегментації (кілька будинків можуть зклеїти в один, і незрозуміло, як їх розділяти). З Mask R-CNN вже не треба думати про додаткові хитрощах: досить векторизовать кордон маски для кожного об'єкта і відразу отримати результат. Є і мінус: розмір маски для об'єкта завжди фіксований, т. Е. Для великих будівель точність розмітки пікселів буде низькою. Результат роботи Mask R-CNN виглядає як на рисунку 4.20.

Рисунок 4.20 – Тегування знайдених об’єктів

Ми спробували Mask R-CNN останнім і переконалися, що для деяких типів забудови цей підхід виграє у інших.

4.5.6 Векторизація прямокутниками

При всьому сучасному архітектурному розмаїтті будинку на супутникових знімках досі найчастіше виглядають як прямокутники. Більш того, для маси територій не потрібна розмітка складними багатокутниками. Але все ж хочеться, щоб вдома на карті були розміщені. (Ну, наприклад, садівниче товариство: будинків там зазвичай багато, вручну розмічати - не так актуально, але позначити прямокутниками на карті дуже непогано.) Тому перший підхід до векторизації був вкрай простий:

– Беремо растрову область, відповідну «дому»;

– Знаходимо прямокутник мінімальної площі, який містить цю область (наприклад, ось так: OpenCV :: minAreaRect). Завдання вирішена.

Зрозуміло, що якість такого підходу далеко від ідеального. Однак алгоритм досить простий і при цьому у багатьох випадках робочий.

4.5.7 Векторизація багатокутниками

Якщо якість сегментації досить гарне, можна більш точно відтворити контур будинку. У більшості будинків складної форми кути в основному прямі, тому ми вирішили звести задачу до задачі побудови багатокутника з ортогональними сторонами.

Вирішуючи її, ми хочемо домогтися відразу двох цілей: знайти найбільш простий багатокутник і якомога точніше повторити форму будівель. Ці цілі конфліктують між собою, тому доводиться вводити додаткові умови: обмежувати мінімальну довжину стін, максимальне відхилення від растрової області і т. д.

Алгоритм, який першим прийшов нам в голову, був заснований на побудові проекції точок на прямі:

–Знайти контур растрової області, що відповідає одному будинку;

– Скоротити кількість точок в контурі, спростивши його, наприклад алгоритмом Дугласа-Пекера;

– Знайти найдовшу сторону в контурі. Саме її кут нахилу визначить кут всього майбутнього ортогонального багатокутника;

– Побудувати проекцію з наступної точки контуру на попередню сторону;

– Продовжити сторону до точки проекції. Якщо відстань від точки до її проекції більше найкоротшою стіни будівлі, додати вийшов відрізок в контур будівлі;

– Повторювати пункти 4 і 5, поки контуру не замкнеться.

Даний алгоритм вкрай простий і швидко приносить результат, але все ж контур будівлі іноді виходить досить зашумленими.

Намагаючись впоратися з цією проблемою, ми натрапили на досить цікавий варіант вирішення завдання, який використовує квадратну сітку в просторі для наближення багатокутника (рис. 4.21).

Якщо описувати коротко, то алгоритм складається з трьох дій:

– Побудувати квадратну сітку в просторі з центром в нулі;

– На точках сітки, які розташовані не далі деякої відстані від вихідного контуру, побудувати різні багатокутники;

– Вибрати багатокутник з мінімальною кількістю вершин.

Рисунок 4.21 – Приклад застосування

Так як необхідний кут повороту сітки заздалегідь невідомий, доводиться перебирати кілька значень, що погано позначається на продуктивності. Однак алгоритм дозволяє домогтися більш візуально красивих результатів.

Даний алгоритм вкрай простий і швидко приносить результат, але все ж контур будівлі іноді виходить досить зашумленими. Намагаючись впоратися з цією проблемою, ми натрапили на досить цікавий варіант вирішення завдання, який використовує квадратну сітку в просторі для наближення багатокутника.

Так як необхідний кут повороту сітки заздалегідь невідомий, доводиться перебирати кілька значень, що погано позначається на продуктивності. Однак алгоритм дозволяє домогтися більш візуально красивих результатів.

4.5.8 Поліпшення векторизації

Поки ми фактично працювали з кожним будинком окремо. Коли перший етап пройдено, можна працювати вже з картиною в цілому і поліпшити результат. Для цього було додано алгоритм постобробки набору багатокутників. Ми скористалися наступними евристиками:

– Зазвичай стіни поруч розташованих будинків паралельні. Більш того: найчастіше вдома можна об'єднати в набори, всередині яких всі елементи вирівняні;

– Якщо на знімку вже розмічені вулиці, то досить імовірно, що сторони багатокутників будуть паралельні вулицях;

– Якщо багатокутники перетнулися, то, швидше за все, є сенс посувати стіни, щоб перетин зникло.

В результаті з'явився такий алгоритм:

– Кластерізуем знайдені будинку по відстані між ними і куту повороту. Усереднюються повороти будівель в кожному кластері. Повторюємо, поки становище будівель не перестає змінюватися або поки вдома не почнуть занадто сильно відхилятися від початкового положення;

– Вибираємо будинку поруч з дорогами, знаходимо найдовший і близьку до дороги сторону. Повертаємо будинок до паралельності обраної сторони і дороги;

– Прибираємо перетину між багатокутниками, які рухаються з боку двох пересічних будівель пропорційно величині сторін.

В результаті ми отримали інструмент, здатний розпізнавати будівлі різних типів забудови. Він допомагає картографом в їх нелегкій роботі: значно прискорює пошук пропущених будинків і заповнення нових, ще НЕ

оброблених територій. На даний момент за допомогою цього інструменту в Народну карту додано вже більше 800 тисяч нових об'єктів.

Нижче ви побачите декілька прикладів розпізнавання (див. рис. 4.22 - 4.29).

Рисунок 4.22 – Розпізнавання на сирому сегменті

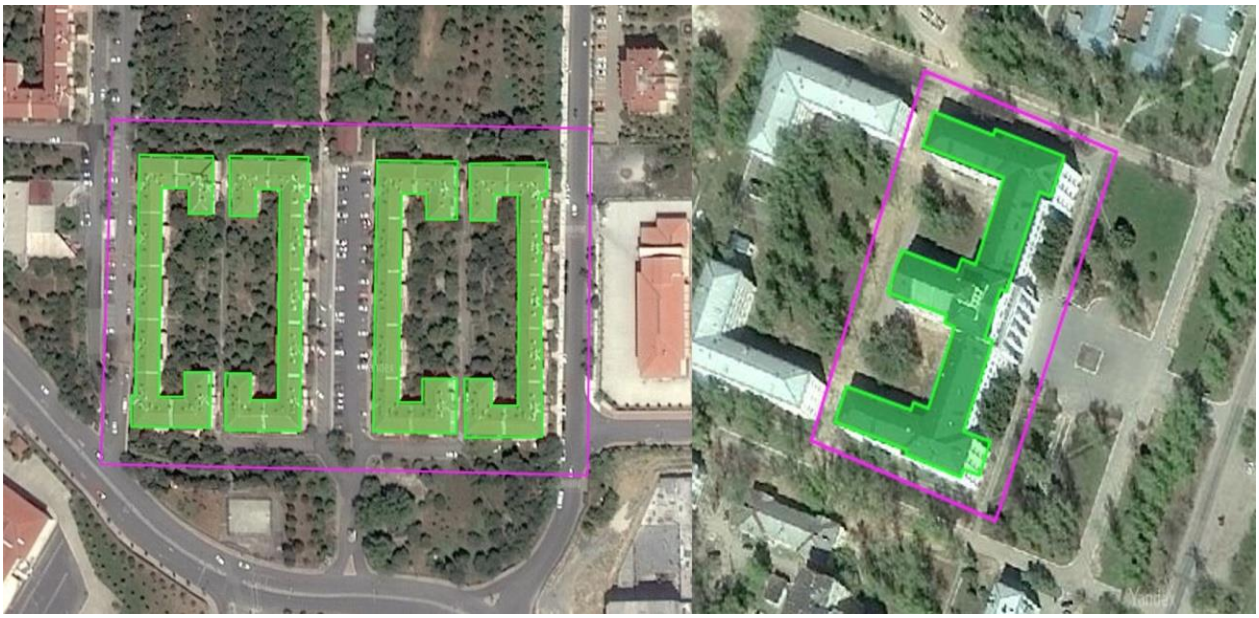


Рисунок 4.23 – Послідовне розпізнання об’єктів

Рисунок 4.24 – Розпізнання в окремому сегменті

Рисунок 4.25 – Розпізнання першого сегменту на готовій мапі

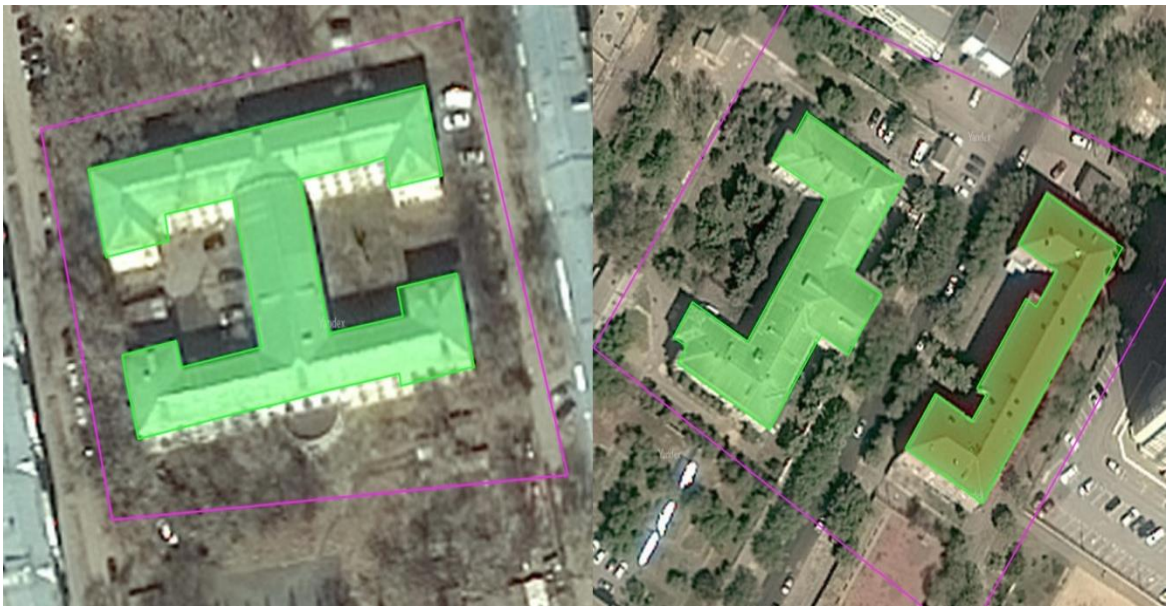


Рисунок 4.26 – Розпізнання без рамок пошуку

На цьому малюнку ми можемо побачити різницю опрацювання окремої ділянки та всієї зони в цілому, не тільки різницю у часі обробки та й ще й точність з швидкістю, проте у контексті цієї роботи рамки та обмеження допускаються тестовими значеннями а у реальному запуску - обмеженням пуску сканування.

Рисунок 4.27 – Розпізнання усієї ділянки

Рисунок 4.28 – Розпізнання на готовій мапі окремого сектору

Рисунок 4.29 – Розпізнання на готовій мапі складного сектору

Результатом останнього малюнку є вибіркове розпізнавання об'єктів у цьому тесті, це - дахи хатинок, отже наша система може працювати з різними заздалегідь навченими об’єктами, а перевагою є факт - що ця обробка може виконуватися у фоновому режимі протє - при онлайн трансляції на сервер.

ВИСНОВКИ

В атестаційної роботі було виконано дослідження методів та використання сучасних програмних засобів і технологій для реалізації створення карт і розпізнаванню об'єктів обраних об'єктів на карті.

На підставі аналізу проблеми і розглянутих існуючих рішень, був запропонований новий і загальнодоступний спосіб створення мап та миттєвої обробки даних на ній. Виявлено найбільш оптимальний спосіб розпізнавання об'єктів на мапах та занесення їх до бази вже існуючої карти.

Також були розглянуті основні проблеми реалізації та їх вирішення. Були виділені основні існуючі аналоги і їх головні функції, які в тому чи іншому вигляді були реалізовані в даному проекті.

Підсумком дипломної роботи була розроблена система, що готова для роботи над створенням нових карт, система розширювана і готова для подальшої модернізації зі збереженням основних канонів головної логіки системи. Алгоритми бізнес-логіки системи зберігають правило бути повністю закритим від змін але відкритим до розширень, що видно по достатком місць для модернізації та навчання його системи. Вся система може розвиватись самостійно за допомогою нейронної мережі, яка становиться розумніше з кожним застосуванням системи. Система може застосовуватись на будь якому носії у який можливо встановити програмний додаток.

Система відкрита та може застосовуватись у різних сферах. Сценарії використання можуть бути від звичайних, як будівництво дома, до секретних воєнних місій, як визначення кількості об’єктів на території.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Инструкция из фотограмметрических работ при создании цифровых топографических карт и планов ГНТА -02-036-02. Москва, Цниигаик, 2002.

2. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ: Про ОБЧІСЛЕННЯ, М.А. Новотарській, Б.Б. Нестеренко – Інститут математики НАН України, Київ 2004 – 127 с.

3. Drummond T., Rosten E. Machine Learning for high-speed corner detection - 9th European Conference on Computer Vision (ECCV), p. 430-443, 2006.

4. Lowe D.G. Distinctive Image Features from Scale-Invariant Keypoints – International Journal of Computer Vision, p. 91-110, 2004.

5. Bay H., Ess A., Yuitelaars T., Van Gool L. SURF: Speeded up robust features – Computer Vision and Image Understanding, v. 110, p. 346-359, 2008.

6. Арлазаров В.Л., Булатов К.Б., Чернов Т.С. Метод нечеткого поиска изображений в больших объемах видеоданных – Системы высокой доступности, Т. 12, № 1, с. 53-58, 2016.

7. Martin A. Fischler, Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography – Comm. of the ACM, v. 24, p. 381-395, 1981.

8. Skoryukina N. et al. Snapscreen: TV-stream frame search with projectively distorted and noisy query – 9th International COnference on Machine Vision (ICMV) — Proc. SPIE V. 10341, P. 103410Y, 2017.

9. Bouguet J.Y. Pyramidal implementation of the affine lucas kanade feature tracker: destription of the algorithm – Intel corporation, V. 5, p. 1-10, 2001.

10. Newman P., Ho K. SLAM-loop closing with visually salient features – IEEE Proc. of International Conference on Robotics and Automation, p. 635-642, 2005.

11. Paalanen P., Kamarainen J.K., Kalviainen H. Image based quantitative mosaic evaluation with artificial video – Scandinavian Conference on Image Analysis, Springer (Berlin, Heidelberg), p. 470-479, 2009.

12. Черепахіна К.К., Назаров О.С., AUTOMATED SYSTEM FOR CONSTRUCTION OF CARDS WITH METADATE IN REAL TIME. X Міжнародної наукової конференції «Perspectives of Science and Education», 2019. – 4с.

13. Черепахіна К.К., Назаров О.С., АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДЛЯ ПОСТРОЕНИЯ КАРТ С МЕТАИНФОРМАЦИЕЙ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ. XII Міжнародної наукової конференції «Science and Society», 2019. – 2с.

14. Р. Гонсалес, Р. Вудс Цифровая обработка изображений [Текст] — М: Техносфера, 2005 – 1007с.

15. Кудрявцев Л.В. Краткий курс математического анализа [Текст] – М.: Наука, 1989 – 736с.

16. Анисимов Б.В. Распознавание и цифровая обработка изображений [Текст] – М.: Высш. школа, 1983 – 295с.

17. Золотых Н.Ю., Кустикова В.Д., Мееров И.Б. Обзор методов поиска и сопровождения транспортных средств на потоке видеоданных – Вестник Нижегородского университета им Н.И. Лобачевского. Выпуск № 5-2 / 2012

18. М.О. Гончаренко. Сравнительный анализ методов формирования дескрипторов изображений в контексте задачи сегментации видеопотока. – Бионика интеллекта. 2015.

19. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных / DEVACADEMY. [Електронний ресурс]. – Режим доступу: URL: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/ – 29.03.2018 р.

20. SQL или NoSQL – вот в чём вопрос. [Електронний ресурс]. – Режим доступу: URL: https://habr.com/company/ruvds/blog/324936/ – 18.05.2018 р.

21. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика, 3-е издание / Т. Коннолли, К. Бегг – М.: «Вильямс», 2017. – 1440 с.

22. Pro C# 7 With .NET and .NET Core / Authors: Troelsen, Andrew, Japikse, Philip, 2017. – 337 с.

23. Professional C# 7 and .NET Core 2.0 / Christian Nagel (You?),Apr 17, 2018 – 95 с.