НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інтелектуальний бот пошуку безпечного маршруту»

Виконав:
студент IV курсу, групи ІС-93
Петренко Владислав Олександрович _____

Керівник:
доц., д.т.н. Дорогий Ярослав Юрійович _____

Рецензент:
доц., к.т.н., доцент Василь ЦУРКАН _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Петренку Владиславу Олександровичу

1. Тема проєкту «Інтелектуальний бот пошуку безпечного маршруту», керівник проєкту Дорогий Ярослав Юрійович, к.т.н., доцент, затверджені наказом по університету від «31» травня 2023 р. No2101-с

2. Термін подання студентом проєкту: 12 червня 2023 року

3. Вихідні дані до проєкту: мова програмування Python, бібліотеки open-cv, numpy, heapq, tkinter, torch, torchvision.

4. Зміст пояснювальної записки: Загальні положення, інформаційне забезпечення, математичне забезпечення, програмне та технічне забезпечення, технологічний розділ

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): діаграма послідовності, блок-схема проекту, графіки точності та витрат під час навчання PathFindingNetwork, структура нейромережі PathFindingNetwork

6. Дата видачі завдання *«4» квітня 2023 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Дослідження актуальності теми ДП, можливості її застосування у різних сферах	07.04.2023	
2	Формування цілей та завдань ДП	12.04.2023	
3	Аналіз теоретичної бази та рекомендованої літератури	15.04.2023	
4	Аналіз існуючих методів реалізації поставлених завдань	29.04.2023	
5	Формування структури на плану проекту	08.05.2023	
6	Розробка інформаційної та програмної частини ДП	11.05.2023	
7	Оформлення пояснювальної записки ДП	02.06.2023	
8	Подання ДП до захисту	12.06.2023	

Студент Владислав ПЕТРЕНКО

Керівник Ярослав ДОРОГИЙ

АНОТАЦІЯ

Петренко В. О. Проектування інтелектуального боту пошуку безпечного шляху. КПІ ім. Ігоря Сікорського, Київ, 2023.

Пояснювальна записка дипломного проекту містить 60 сторінок, 20 рисунків, 6 таблиць, посилання на 21 джерело, 1 додаток та 4 конструкторських документів.

Ключові слова: пошук шляху, оптимальний шлях для користувача, пошук шляху з уникненням загроз

Об’єктом розробки є інтелектуальний бот пошуку безпечного шляху.

Мета розробки – пошук оптимального шляху з початкової до кінцевої точки з уникненням небезпечних зон.

У дипломному проекті розроблена можливість користувачеві вибрати файл з яким буде працювати, можливість самому вибрати початкову та кінцеву точку маршруту та зазначити інформацію щодо безпечних та небезпечних зон. Також був реалізований простий інтерфейс в вигляді вікна, куди виводились результати роботи програми.

Отримані результати можуть бути корисними при плануванні чи перевірці безпечного маршруту без ризиків.

ABSTRACT

Petrenko V. O. Designing an intelligent bot for finding a safe path. Igor Sikorsky KPI, Kyiv, 2023.

The explanatory note of the diploma project contains 60 pages, 20 figures, 6 tables, links to 21 sources, 1 appendix and 4 design documents.

Keywords: path finding, optimal path for the user, path finding with threat avoidance

The object of development is an intelligent bot for finding a safe path.

The purpose of the development is to find the optimal path from the starting point to the final point while avoiding dangerous areas.

The diploma project developed an opportunity for the user to choose a file to work with, an opportunity to choose the starting and ending point of the route and to note information about safe and dangerous zones. A simple interface in the form of a window was also implemented, where the results of the program were displayed.

The obtained results can be useful in planning or checking a safe route without risks.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	І С93.200БАК.004 ПЗ	Пояснювальна записка	60		
6	А3	І С93.200БАК.004 Д1	Інтелектуальний бот	1		
7			пошуку безпечного			
8			маршруту.			
9			Діаграма послідовності			
10						
11	А3	І С93.200БАК.004 Д2	Інтелектуальний бот	1		
12			пошуку безпечного			
13			маршруту.			
14			Блок-схема проекту			
15						
16	А3	І С93.200БАК.004 Д3	Інтелектуальний бот	1		
17			пошуку безпечного			
18			маршруту.			
19			Графіки точності та витрат під			
20			під час навчання			
21			PathFindingNetwork			
22	А3	І С93.200БАК.004 Д4	Інтелектуальний бот	1		
23			пошуку безпечного			
24			маршруту.			
25			Структура нейромережі			
26			PathFindingNetwork			
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	І С93.200БАК.004 ТП		
Розроб.		Петренко В.О.			Інтелектуальний бот пошуку безпечного маршруту. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Дорогий Я. Ю.				Т	
Затв.						КПІ ім. Ігоря Сікорського Група ІС-93	

Пояснювальна записка
до дипломного проєкту
на тему: «Інтелектуальний бот пошуку безпечного маршруту»

Київ – 2023 року

ЗМІСТ

ВСТУП

Проектування інтелектуального боту пошуку безпечного шляху є однією з захоплюючих та важливих задач у галузі штучного інтелекту. З постійним ростом містобудування та автомобільного транспорту, безпека руху стає однією з ключових проблем, яку потрібно вирішувати.

Основною метою цього проекту є створення інтелектуального боту, який здатний аналізувати та знаходити безпечний шлях для користувача. Цей бот буде використовувати передові методи машинного навчання, зокрема нейронні мережі, для розпізнавання та аналізу зображень, пов'язаних з дорожніми умовами та безпекою.

За допомогою навчання з вчителем або навчання з підкріпленням, бот буде тренуватися на великому наборі даних, що містить зображення з доріг та відповідні мітки безпеки шляху. Це дозволить боту вивчити розрізняти безпечні, небезпечні та нормальні зони на дорозі, а також ураховувати оточуюче середовище для прийняття найкращих рішень щодо маршруту.

Перевагою використання нейронних мереж для пошуку безпечного шляху є їх здатність аналізувати складні дані зображень та виявляти складні залежності між дорожніми умовами та безпекою. Це дозволить боту робити більш обґрунтовані рішення та надавати користувачам оптимальний маршрут, який максимізує їх безпеку на дорозі.

Проект по проектуванню інтелектуального боту пошуку безпечного шляху має великий потенціал для забезпечення комфортного та безпечного переміщення для користувачів.

Тому основною метою цієї роботи є проектування інтелектуального бота пошуку безпечного шляху, що базується на передових методах штучного інтелекту та машинного навчання.

Успішне впровадження цього проекту матиме значний вплив на безпеку руху та зручність переміщення великої кількості людей без ризиків. Крім того, цей проект може послужити основою для подальших досліджень та розробок у сфері

автономних транспортних засобів та містобудування. Забезпечення безпечного шляху є важливим кроком у напрямку створення більш сталого, ефективного та безпечного міського середовища.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Предметне середовище цього проекту пов'язане з пошуком безпечного шляху в різних ситуаціях. Завдання полягає у визначенні найбільш безпечного маршруту з урахуванням можливих небезпек чи перешкод.

Для успішного пошуку безпечного шляху необхідно враховувати інформацію про середовище, в якому здійснюється переміщення. Ця інформація може включати дані про доступні маршрути або альтернативні шляхи.

Алгоритми та методи, що застосовуються в предметному середовищі пошуку безпечного шляху, включають аналіз даних про середовище, прогнозування можливих подій або небезпек, а також оптимізацію маршруту з метою мінімізації ризиків.

Такі методи можуть бути засновані на принципах штучного інтелекту, машинного навчання чи інших обчислювальних технологій [1], [14].

У предметному середовищі пошуку безпечного шляху важливо враховувати тимчасові обмеження та динамічні зміни, які можуть вплинути на безпеку шляху. Наприклад, у разі евакуації при надзвичайних ситуаціях потрібно враховувати поточний стан середовища та визначати найбільш безпечний та ефективний шлях для людей.

Також необхідно враховувати взаємодію з користувачем чи іншими системами. Можливе надання користувачеві інтерфейсу для введення параметрів або надання інформації про знайдений безпечний шлях.

В результаті, предметне середовище пошуку безпечного шляху включає аналіз даних про середовище, вибір оптимального маршруту з урахуванням безпеки, а також взаємодія з користувачем або іншими системами для забезпечення ефективності і зручності використання.

1.2 Опис функціональної моделі

Функціональна модель - це математичне представлення або опис функцій та взаємозв'язків між ними, що визначають поведінку системи або процесу. В контексті програмного забезпечення функціональна модель використовується для опису функціональних вимог системи або компонента і слугує основою для подальшого проектування та розробки програмного забезпечення [2].

Вона допомагає уточнити, які задачі та операції повинні бути виконані системою, які дані потрібно обробляти, які функції мають бути реалізовані та які взаємодії між компонентами системи мають бути забезпечені. Також дозволяє уявити систему у вигляді функцій та їх взаємозв'язків, що допомагає визначити необхідні елементи та логіку їх взаємодії.

1 Завдання графа:

- Визначення вершин та ребер графа.
- Завдання ваг ребер, що відображають рівень безпеки або ризику.

2 Отримання початкової та кінцевої вершин: введення початкової та кінцевої вершин, між якими потрібно знайти безпечний шлях.

3 Алгоритми пошуку шляху:

- Застосування алгоритму пошуку найкоротшого шляху, такого як алгоритм Дейкстри чи алгоритм A*.
- Облік ваг ребер визначення найбільш безпечного шляху.

4 Аналіз безпеки шляху:

- Оцінка безпеки кожного ребра на знайденому шляху на основі їх ваг.
- Визначення загальної безпеки шляху на основі безпеки його складових ребер.

5 Перевірка умов безпеки, пов'язаних із конкретними вимогами чи обмеженнями (наприклад, мінімальний рівень безпеки, уникнення небезпечних зон, тощо).

6 Візуалізація знайденого безпечного шляху:

- Візуальна вистава знайденого шляху на графі з відображенням безпеки

кожного ребра.

- Надання інформації про безпеку шляху, включаючи рекомендації щодо дій.

7 Оновлення ваг ребер та повторний аналіз:

- Можливість оновлення ваг ребер графа за зміни умов безпеки або зовнішніх факторів.

- Повторний запуск алгоритму пошуку шляху та аналізу безпеки для оновлених ваг.

1.2.1 Опис алгоритму Дейкстри A*

Алгоритм A* - це евристичний метод пошуку маршруту. Він поєднує дві складові: функцію оцінки окремих вузлів та визначення відстані від поточного вузла до кінцевого.

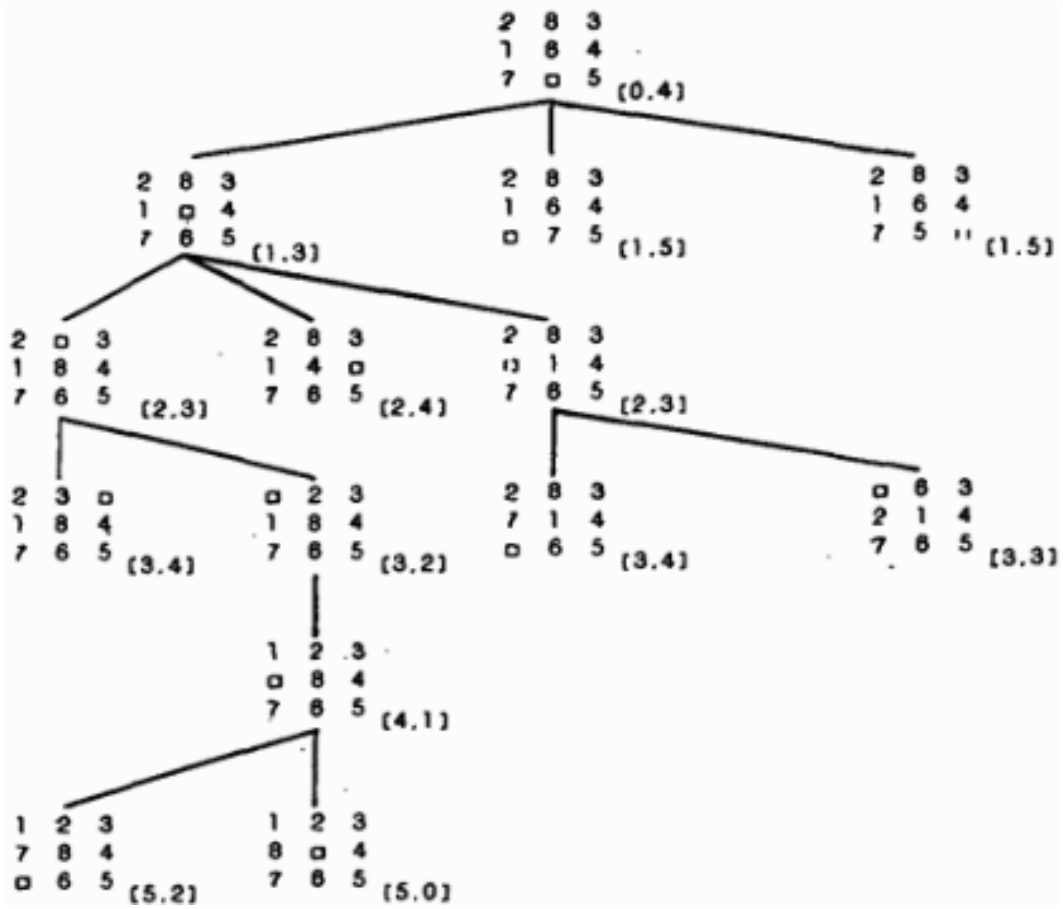

Рисунок 1.1 – Знаходження шляху в графі

Основні кроки алгоритму A* включають створення двох списків, відкритого та закритого. Відкритий список містить нерозглянуті вузли, а закритий - вузли, які вже були перевірені. Початковий вузол ініціалізується з нульовими значеннями та додається до відкритого списку. Далі алгоритм повторює наступні кроки, поки відкритий список не порожній або не знайдено кінцевий вузол.

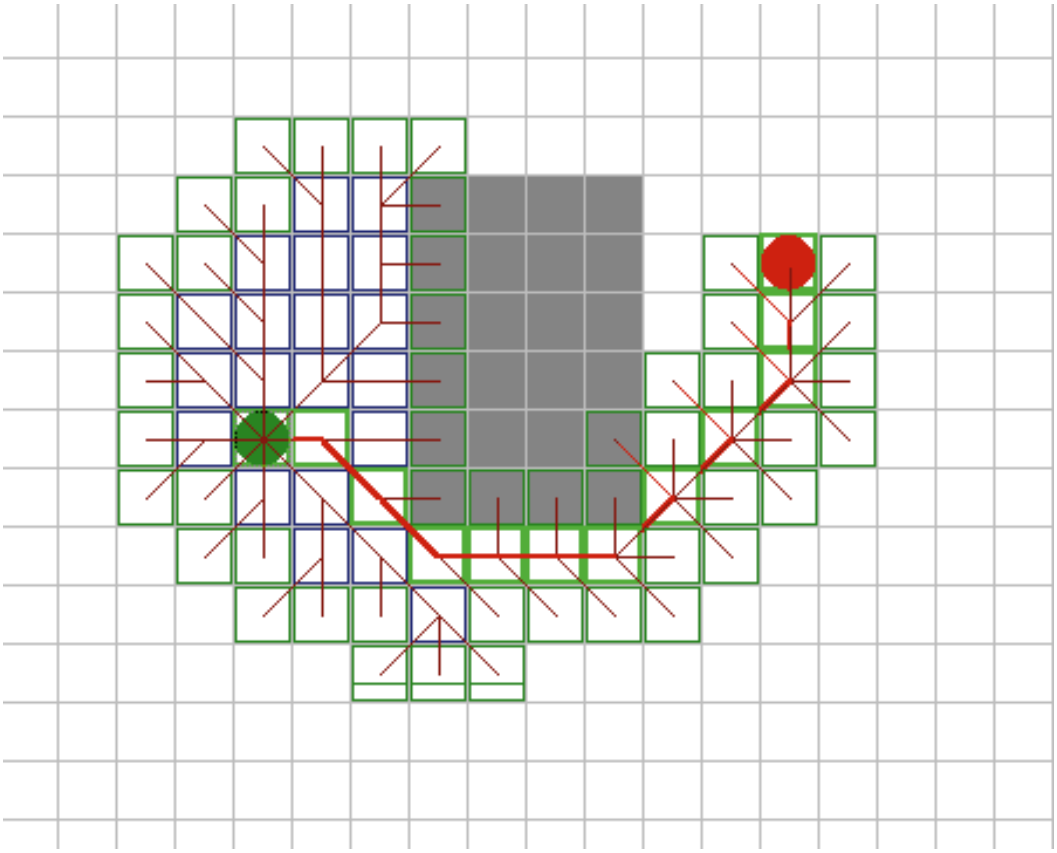

Рисунок 1.2 – Знаходження шляху з перешкодами за допомогою алгоритму

Спочатку обирається вузол з найменшою сумарною оцінкою. Потім цей вузол переміщується з відкритого списку до закритого. Для кожного сусіднього вузла, якщо він не має оцінки, обчислюється нова оцінка шляху та оцінка залишкової відстані.

Якщо знайдено кінцевий вузол, алгоритм відновлює шлях, прослідковуючи назад від кінцевого вузла до початкового за допомогою вказівників на попередні вузли.

Отже, алгоритм A* використовується для пошуку найкращого маршруту в графі, де кожному ребру або вузлу присвоюються вартості. Його ефективність полягає в використанні евристичної функції для прискорення пошуку шляху до кінцевого вузла.

1.2.2 Опис нейромережі PathFindingNetwork

Нейромережа PathFindingNetwork (або мережа пошуку шляху) - це тип нейромережі, призначений для вирішення задач пошуку оптимального шляху в графі або на карті.

Основна ідея застосування нейромережі PathFindingNetwork полягає в тому, щоб навчити її визначати оптимальний шлях в графі або на карті шляхом аналізу оточуючих даних. Зазвичай ця мережа навчається за допомогою набору тренувальних даних, які містять інформацію про стан середовища та оптимальний шлях.

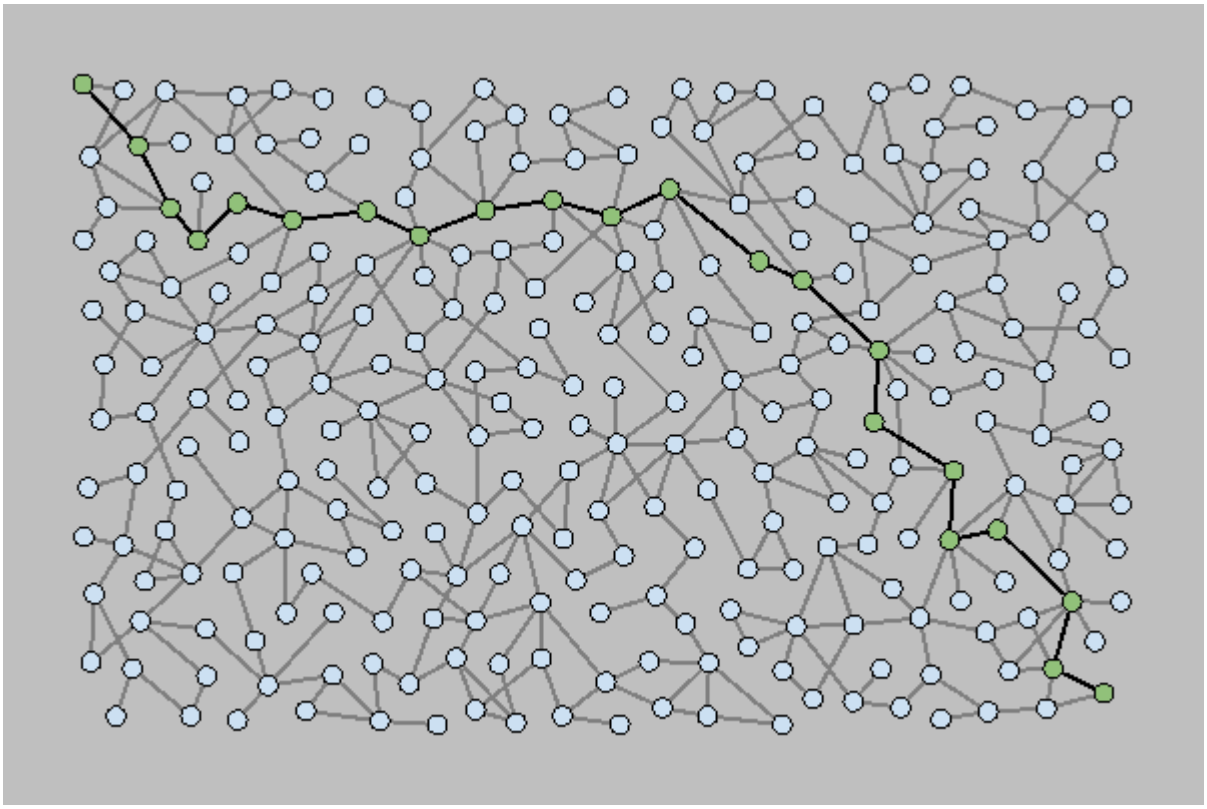

Рисунок 1.3 – Знайдений шлях за допомогою нейромережі

Після навчання PathFindingNetwork може бути використана для визначення оптимального шляху в реальному часі. Шлях, обчислений нейромережею, може бути використаний для навігації агента, рухаючись від початкової до кінцевої точки, уникайучи перешкод або враховуючи обмеження середовища.

1.3 Опис наявних аналогів

Пошук маршруту – найбільш затребуваний функціонал на сьогоднішній день. Кількість сервісів та додатків, що дозволяють дістатися з пункту А до пункту Б по найоптимальнішому шляху зростає з кожним днем. Більшість із них орієнтована виключно на побутове використання, часто з використанням Інтернету. Розглянемо найбільш відомі платформи пошуку маршрутів.

1.3.1 Google Maps

Google Maps є однією з найпопулярніших картографічних платформ. Він пропонує функцію пошуку маршруту, яка враховує дорожні умови, пробки та інші фактори визначення безпечного шляху.

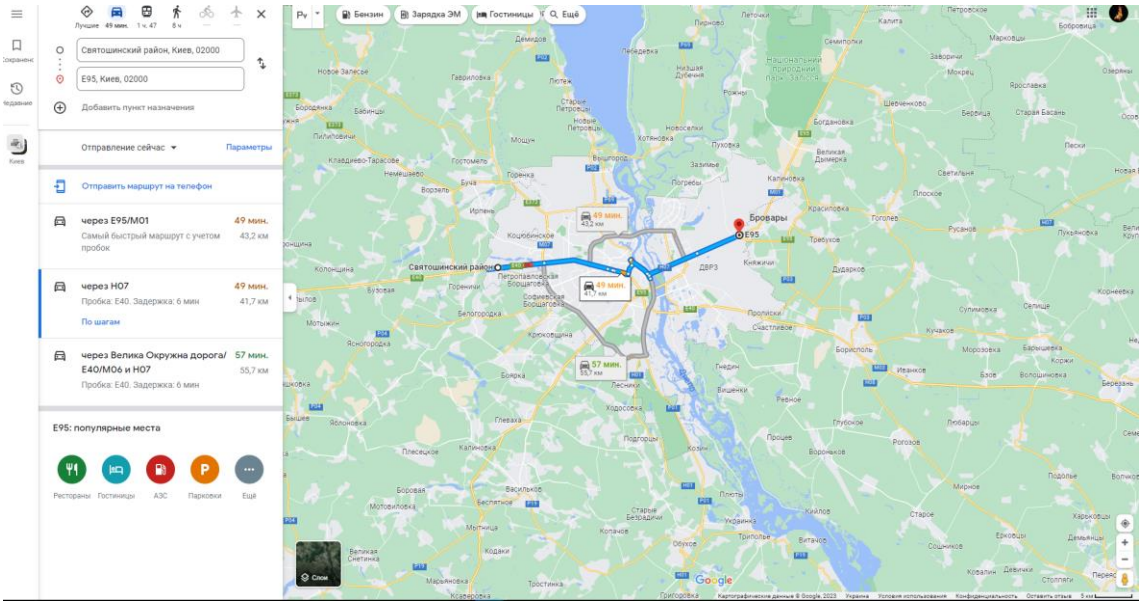

Рисунок 1.4 – Google Maps

Плюси:

- Широке охоплення та актуальність даних
- Інтуїтивно зрозумілий інтерфейс
- Надає додаткову інформацію про місця, такі як відгуки про заклади та фотографії

Мінуси:

- Не завжди враховує поточні умови дороги
- Може бути менш надійним у малонаселених чи віддалених районах
- Потрібне підключення до Інтернету для повноцінного використання
- Надає можливість завантажувати оффлайн-карти місцевості
- Враховує обраний тип відображення карт (графічний, супутниковий)
- Пропонує декілька варіантів маршруту, спираючись лише на найкоротший час шляху
- Рішення щодо найкращого маршруту приймає користувач, не маючи інформації щодо стану доріг, тощо

На сьогодні Google Maps безумовний лідер ринку, що обумовлено широким функціоналом, що постійно оновлюється та адаптується до потреб саме пересічних користувачів. Наприклад, увімкнена опція «Мінімум ходьби» дозволить людині відстроїти шлях таким чином, щоб якнайменше пересуватись пішки, що досить актуально для літніх людей, або батьків з дітьми.

Також слід зазначити безумовно корисну для користувача функцію відображення розкладу руху громадського транспорту. Але ця функція не працює досконало у зв'язку з відсутністю можливості відстежувати транспорт в режимі реального часу.

Незважаючи на багатий функціонал, Google Maps не має можливості оцінювати маршрути щодо якості доріг, що залишається великою проблемою при самостійному виборі маршруту користувачем.

1.3.2 Навігатор Waze

Waze - це соціально-орієнтований додаток для навігації, який ґрунтується на інформації, наданій самими користувачами. Воно дозволяє обмінюватися даними про пробки, аварії та інші події на дорозі, що допомагає вибрати більш безпечний шлях [18].

Плюси:

- Оновлюється в реальному часі за допомогою даних спільноти користувачів
- Активна участь користувачів сприяє точності та актуальності інформації
- Пропонує альтернативні маршрути для обходу пробок та інших перешкод

Мінуси:

- Деякі користувачі можуть надавати неточну інформацію
- Може бути менш надійним у деяких регіонах з невеликою кількістю користувачів
- Потрібне підключення до Інтернету для отримання поточних даних
- Не має критеріїв оцінювання перешкод за ступенем складності
- Визначені користувачем перешкоди на дорозі не можуть бути враховані чітко за часом на їх подолання, тож час шляху лишається лише ймовірним

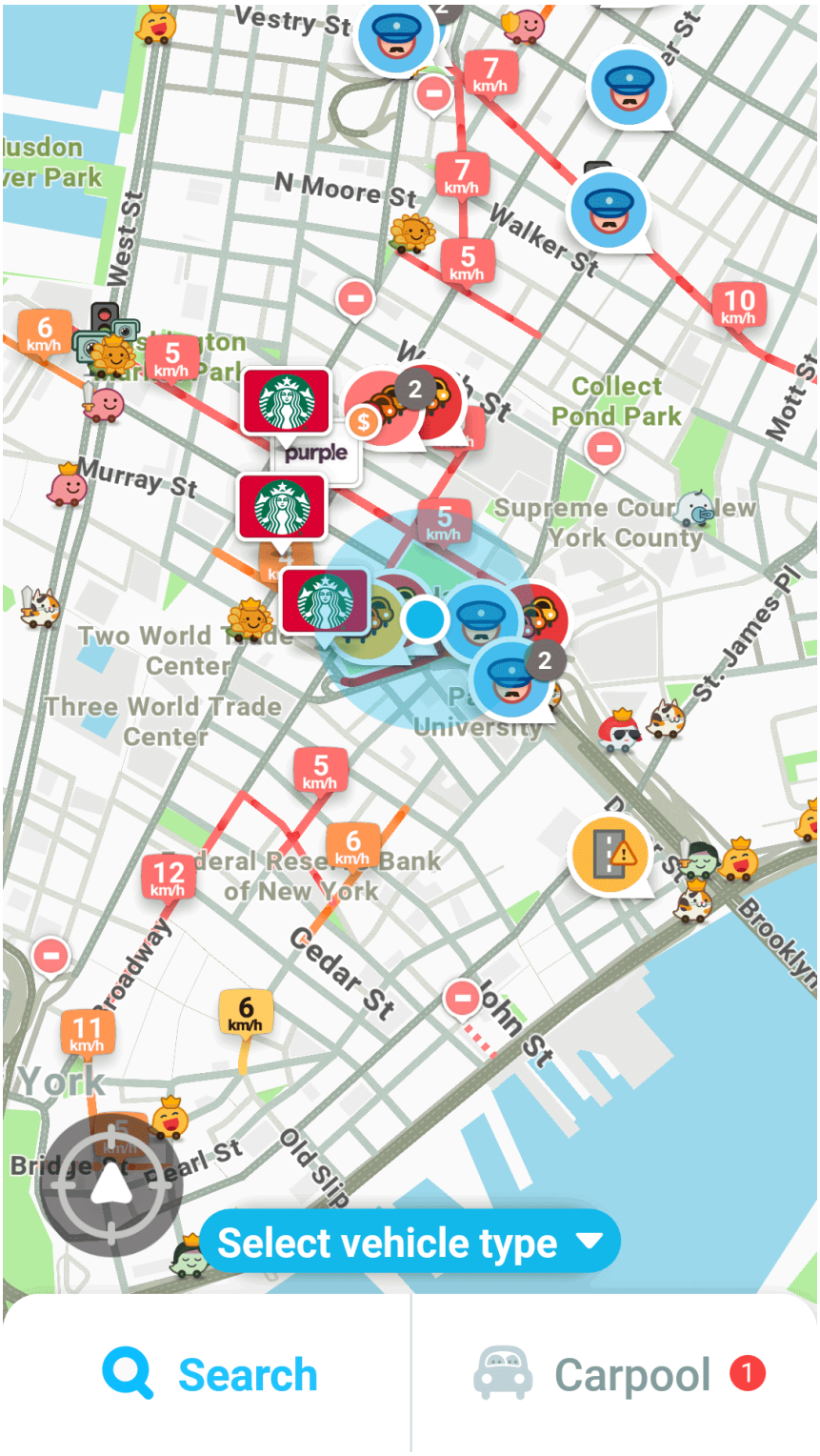

Рисунок 1.5 - Waze

Соціальна орієнтація програми робить його дуже популярним серед водіїв. Голосові підказки про обмеження швидкості, запам'ятовування маршрутів, що часто використовуються – все це робить Waze незамінним помічником у дорозі.

Також помітним кроком до підвищення соціалізації програми стала адаптація навігатора до соціальних мереж, завдяки чому у водіїв з'явилася можливість ділитися ситуацією на дорозі зі своєю спільнотою.

Суттєвим недоліком Waze є неспроможність навігатора оцінювати складність маршруту, навіть при безпосередньою участю користувачів, через що оптимальність шляху залишається сумнівною.

1.3.3 Платформа Here WeGo

Here WeGo - це безкоштовна картографічна платформа, яка пропонує маршрути та навігацію. Вона враховує поточний рух, пробки, громадський транспорт та інші чинники знаходження безпечного шляху [20].

Плюси:

- Дозволяє планувати маршрути для різних видів транспорту
- Забезпечує інформацію про транспортний засіб та розклад громадського транспорту
- Можна використовувати офлайн без підключення до інтернету

Мінуси:

- У деяких регіонах може бути менш докладним та актуальним у порівнянні з іншими платформами
- Інтерфейс може бути менш інтуїтивним для деяких користувачів
- Часта невідповідність маршрутів та карт поточного часу
- Не завжди коректне визначення оптимального маршруту руху, робить заїзди в глухий кут
- Не враховує ступінь складності доріг при прокладанні маршрутів

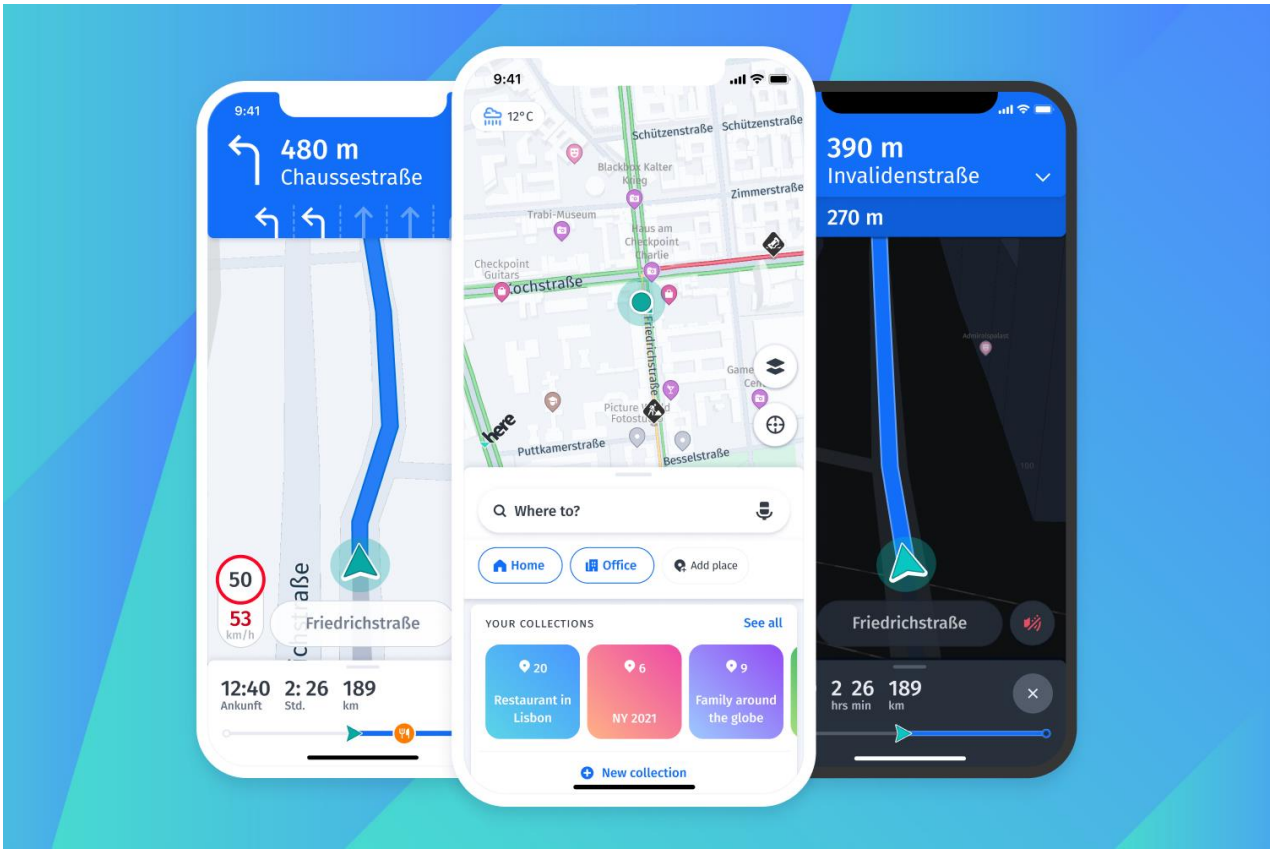

Рисунок 1.6 – HERE WeGo

Навігатор Here WeGo досить популярний серед професійних водіїв, робота яких пов'язана з відвідуванням великої кількості місць на маршруті. Досить цінною є функція голосового помічника, що своєчасно та докладно вказує назву вулиць, поворотів та з'їздів, що дозволяє не відволікатися у дорозі на екран навігатора.

Проте, найбільшим недоліком програми є повністю відсутня можливість обирати маршрут не лише за часом, а і за станом доріг, складністю проходження обраного маршруту, тощо.

1.3.4 Офлайн-навігатор MAPS.ME

Найкращий навігатор в офлайн-сегменті. Найпопулярніша програма серед користувачів, які багато подорожують або часто перебувають у поїздках до регіонів з недостатньо хорошим Інтернет-покриттям.

Плюси:

- Офлайн-картки, що відповідають параметрам «тут і зараз»
- Висока деталізація об'єктів
- Логічна побудова маршрутів залежно від виду транспорту та пересування

Мінуси:

- Необхідність заздалегідь завантажувати карти міста/країни перед кожною поїздкою для підтримки їхньої актуальності
- Іноді неточна відповідність маршрутів (заїзд у глухий кут)
- Часта відсутність об'єктів, що недавно з'явилися.
- Відсутність критеріїв вибору якості доріг, наявних перешкод
- Час обраного маршруту має лише ймовірне значення

Рисунок 1.7 – MAPS.ME

Особливо важливою функцією є точне визначення всіх об'єктів на карті, а також можливість їхньої фільтрації за категоріями. При цьому відповідні об'єкти можуть бути показані як у вигляді переліку, так і у вигляді прапорців на карті. Навігатор відрізняється досить непоганою інформативністю (у вигляді історичної довідки) щодо деяких об'єктів, таких як видатні місцевості [17].

Для мандрівників особливо цінною є наявність додаткових фільтрів у самій категорії, наприклад, можливість відсортувати готелі за ціною, рейтингом, а також датою бронювання. Більшість регіонів навігатор здатний відбудувати передбачуваний маршрут таксі і навіть вказати його приблизну вартість.

Функція складання картки міток у будь-якій точці карти, включаючи лісову зону, дозволяє будувати маршрути поза дорогами. У вигляді платної послуги надається можливість скористатися готовими туристичними маршрутами.

Нажаль, MAPS.ME також не враховує такі важливі параметри на маршруті, як якість дорожнього покриття, складність проходження перешкод.

1.3.5 Висновки

Розглянуті варіанти навігаційних додатків мають різний, але досить багатий функціонал, будучи кращими у своєму сегменті. Усі вони дозволяють визначати існуючі маршрути, виділяти оптимальні, де головним критерієм відбору є час маршруту.

Основним недоліком розглянутих засобів навігації є відсутність можливості оцінювати маршрут щодо категорії зношеності доріг, можливих перешкод та їхнього ступеня важливості. Через це оптимальність маршруту є лише найбільш ймовірною, а час проходження маршруту – приблизним. Жодна з програм навігації на визначає маршрути з точки зору їх безпечності.

1.4 Постановка задачі

В процесі проектування інтелектуального боту пошуку безпечного шляху, постановка задачі є ключовим етапом. Постановка задачі визначає мету проекту, описує проблему, яку треба вирішити, та намічає шляхи досягнення цієї мети.

Основною метою проекту є створення імплементації боту, який повинен мати функції отримання та аналізу вхідної інформації, визначати безпечні маршрути для користувачів.

Процес реалізації проекту може включати етапи збору даних, побудови моделі, навчання, тестування та впровадження, зазначаючи терміни та приблизну послідовність робіт.

1.4.1 Визначення цілей та задач розробки

Основними цілями розробки є: забезпечення безпеки, ефективність, адаптивність.

Головна мета забезпечення безпеки полягає у розробці алгоритму або системи, яка дозволяє знаходити шляхи з мінімальним ризиком та забезпечує безпечне переміщення від однієї точки до іншої. Це може бути важливо, наприклад, в автономних транспортних системах або робототехніці, де безпека та запобігання аваріям є критичними факторами.

Розробка доцільного та ефективного алгоритму, здатного знаходити безпечні шляхи з мінімальними витратами часу та ресурсів, дозволить зменшити час переміщення та підвищити загальну продуктивність системи.

Створення гнучкого алгоритму, який здатний адаптуватися до умов, що змінюються, і перешкод у навколишньому середовищі, може включати динамічне оновлення шляхів у разі виникнення нових загроз або перешкод на маршруті.

Основні задачі розробки:

1. Дослідження та вибір відповідного алгоритму

Вивчення різних алгоритмів, таких як A*, Дейкстра, Флойда-Уоршелла та нейромережі, вибір найбільш відповідного алгоритму для вирішення конкретної задачі безпечного пошуку шляху.

2. Розробка моделі навколишнього середовища

Створення моделі навколишнього середовища, включаючи карту, перешкоди, обмеження та інші фактори, які можуть впливати на безпеку та вибір шляху.

3. Реалізація алгоритму

Розробка програмного коду реалізації обраного алгоритму пошуку безпечного шляху. Це може включати реалізацію алгоритму самостійно або використання існуючих бібліотек та інструментів.

4. Тестування та налагодження

Проведення тестових сценаріїв та аналіз результатів для перевірки правильності роботи алгоритму. Налагодження помилок та покращення продуктивності.

5. Інтеграція та розгортання

Інтеграція розробленого алгоритму до цільової системи або платформи. Це може містити інтеграцію з іншими компонентами, такими як датчики, системи керування або візуалізація результатів. Після інтеграції відбувається розгортання системи практичного використання.

6. Оптимізація та покращення

Постійне дослідження та аналіз можливих поліпшень алгоритму та його продуктивності. Оптимізація може включати покращення швидкості виконання, врахування динамічних змін у середовищі або додавання додаткових критеріїв безпеки.

1.4.2 Призначення розробки

Призначення розробки полягає у створенні алгоритму або системи, яка здатна знаходити оптимальні та безпечні шляхи у різних контекстах.

В автономних транспортних системах розробка пошуку безпечного шляху може бути спрямована на забезпечення безпеки та ефективності руху автономних транспортних засобів, таких як автомобілі, безпілотні літальні апарати (дрони) та роботи. Пошук безпечного шляху дозволяє користувачам уникати перешкод, враховувати дорожні правила та забезпечувати безпеку пасажирів та оточуючих.

У разі надзвичайних ситуацій, таких як пожежі, повені чи інші лиха, розробка пошуку безпечного шляху може допомогти людям евакуюватися з небезпечних зон та оптимізувати планування маршрутів для рятувальних служб.

Пошук безпечного шляху у робототехніці може бути застосований для безпечного переміщення роботів у навколишньому середовищі та маніпуляції об'єктами. Наприклад, роботи на виробничих лініях можуть використовувати пошук безпечного шляху, щоб уникнути зіткнень з людьми або іншими роботами.

Також, розробка пошуку безпечного шляху може бути використана в ігровій індустрії для управління поведінкою віртуальних персонажів та NPC. Це дозволяє їм уникати перешкод, шукати безпечні маршрути та взаємодіяти з навколишнім середовищем.

Загалом, призначення розробки полягає у забезпеченні безпеки, ефективності та оптимальності переміщення у різних сценаріях та сферах застосування.

1.5 Сфери застосування

Цей проект має потенціал для застосування в різних галузях, де потрібна обробка зображень та аналіз даних. Основна мета програми – виявлення жовтих та червоних кіл на зображенні та побудова шляху між ними.

Програма може бути використана для навігації роботів у середовищах, де присутні жовті та червоні кола. Наприклад, це може стосуватися автономних автомобілів або рухомих роботів, які повинні навігувати в заданих областях.

В медицині програма може бути використана для виявлення та аналізу жовтих та червоних кіл на медичних зображеннях, що можуть свідчити про певні структури або патології, такі як пухлини чи аномалії.

В системах комп'ютерного зору програма може знаходити застосування в розпізнаванні об'єктів або детектуванні маркерів на зображеннях. Жовті та червоні кола можуть використовуватися як маркери для визначення позиції об'єктів або їх орієнтації у просторі.

У галузі відеоігор програма може допомагати побудові шляхів між об'єктами або зонами. Наприклад, в грі з відкритим світом, де гравцю дозволено вільно рухатися, програма може автоматично створювати шляхи для штучного інтелекту або навігації гравця.

Проект також може бути використаний в системах розпізнавання образів для виявлення та ідентифікації жовтих та червоних кіл на зображеннях. Це може бути корисно у сферах, таких як роботизоване сортування, контроль якості або розпізнавання об'єктів на зображеннях.

В сучасних системах віртуальної реальності цей проект може бути використаний для відстеження руху користувача або об'єктів.

Це лише кілька прикладів можливого застосування програми. Завдяки можливостям обробки зображень та аналізу даних, програма може бути адаптована для використання в різних ситуаціях, де потрібно виявити та аналізувати кольори на зображеннях.

Крім того, цей проект може мати застосування в системах безпеки та відеоспостереження. Наприклад, програма може виявляти жовті та червоні кола на відеозаписах і сповіщати оператора про потенційно небезпечні ситуації, такі як втручання або пожежа.

У галузі досліджень та науки цей проект може бути використаний для вивчення поведінки об'єктів або процесів, що включають жовті та червоні круги. Наприклад, програма може аналізувати рух транспортних засобів на дорозі, міграцію птахів або розподіл елементів на хімічному реакторі.

Застосування цього проекту також може бути в області розпізнавання обличь, де виявлення жовтих та червоних кіл може використовуватись для розпізнавання та ідентифікації особливих ознак або елементів на обличчях людей.

Одним із можливих застосувань є індустрія дронів, де програма може бути використана для автоматичного визначення місцезнаходження дронів у просторі шляхом розпізнавання жовтих та червоних маркерів на землі.

В цілому, можливості цього проекту є досить широкими і можуть бути використані у багатьох індустріях, де виявлення та аналіз об'єктів на зображеннях є важливим завданням.

Висновок до розділу

Під час написання даного розділу дипломного проекту було розглянуто предметне середовище, що є об'єктом дослідження. Крім цього, було описано процес діяльності, який має бути автоматизованим, а також очікувану взаємодію користувача з системою. Були визначені функціональні вимоги до системи та їх пріоритетність, а також побудована схема варіантів використання.

Деякий аналіз існуючих аналогів був проведений, з'ясовано їх відмінності від розроблюваного продукту. Було описано технології, які використовуються в існуючих аналогах, та вивчено наявні методи виконання задачі. Сформульовано призначення та мету розробки, а також визначено задачі, які необхідно вирішити для досягнення поставленої мети.

Для досягнення поставленої мети, було проведено аналіз існуючих технологій та платформ, які можуть бути використані для розробки системи. Були розглянуті різні мови програмування, фреймворки та інструменти, що можуть забезпечити ефективну реалізацію функціональності системи. На основі цього аналізу було прийнято рішення вибрати конкретну технологічну стек для подальшої розробки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Визначення вхідних та вихідних даних

У загальному випадку, вхідні та вихідні дані можуть бути визначені таким чином:

Вхідні дані:

1. Граф або модель навколишнього середовища

Це структура даних, яка є сіткою, графом або картою, де кожен вузол або вершина являє місце розташування, а ребра визначають зв'язки між місцезнаходженнями. Кожне ребро може бути зважене для врахування різних факторів, таких як відстань, час або рівень безпеки.

2. Початкова та кінцева точки

Це місце, звідки починається пошук безпечного шляху (початкова точка) та місце, до якого потрібно дістатися (кінцева точка).

3. Додаткові обмеження та критерії безпеки

В деяких випадках можуть бути задані додаткові обмеження або критерії безпеки, які мають бути враховані під час пошуку шляху.

Вихідні дані:

1. Безпечний шлях

Це послідовність вузлів або вершин, яка є оптимальним шляхом від початкової точки до кінцевої точки. Кожна вершина шляху визначає місце, яке необхідно відвідати в порядку прямування.

2. Властивості шляху

Разом із безпечним шляхом можуть бути надані додаткові властивості шляху які допомагають оцінити якість та ефективність знайденого шляху.

2.2 Структура масивів інформації

Image - масив зображення, яке зчитується з файлу за допомогою функції cv2.imread(). Він містить числові значення пікселів зображення та інформацію про кольори.

hsv_image також масив зображення, яке отримується шляхом конвертації початкового зображення (image) у кольоровий простір HSV за допомогою функції cv2.cvtColor(). Він містить числові значення пікселів зображення в просторі кольорів HSV.

lower_yellow і upper_yellow два масиви, які визначають діапазон кольорів жовтого кольору в просторі кольорів HSV. Вони використовуються для створення маски, яка виділяє жовті об'єкти на зображенні.

lower_red1, upper_red1, lower_red2 і upper_red2 потрібні для визначення діапазону кольорів червоного кольору в просторі кольорів HSV. Вони використовуються для створення маски, яка виділяє червоні об'єкти на зображенні. У випадку червоного кольору використовується два діапазони (lower_red1, upper_red1 і lower_red2, upper_red2), оскільки червоний колір простору HSV знаходиться на межі діапазонів.

Масив kernel представляє ядро (структуру) для морфологічних операцій. В даному випадку, використовується ядро розміром 5x5, що складається з одиниць (np.ones((5, 5), np.uint8)). Це ядро використовується для морфологічної обробки масок (застосування операцій відкриття та закриття).

yellow_mask і red_mask - маски, які отримуються застосуванням операції cv2.inRange() до hsv_image з використанням відповідних діапазонів кольорів (lower_yellow, upper_yellow і lower_red, upper_red). Маска представляє собою бінарне зображення, де значення 255 відповідають пікселям, що потрапляють у відповідний діапазон кольорів, а значення 0 - пікселям, які не відповідають діапазону.

yellow_contours і red_contours два списки контурів, які отримуються за допомогою функції cv2.findContours() для масок yellow_mask і red_mask

відповідно. Контур представляє собою послідовність точок, які утворюють замкнену криву. Вони використовуються для отримання координат центрів і радіусів мінімальних обкладинок для жовтих і червоних об'єктів.

yellow_centers і red_centers: Це списки центрів мінімальних обкладинок, які отримуються за допомогою функції cv2.minEnclosingCircle() для кожного контура в yellow_contours і red_contours відповідно. Кожен центр представлений координатами (x, y) у вигляді кортежу і додається до відповідного списку.

Словник graph представляє собою граф, де ключами є центри жовтих і червоних об'єктів, а значеннями - списки сусідніх центрів. Граф використовується для виконання алгоритму для пошуку безпечного шляху між початковим і кінцевим центрами.

start і end - початковий і кінцевий центри, які вибираються зі списку жовтих центрів.

path це список координат, що представляють шлях від початкового до кінцевого центру, який отримується в результаті виконання програми. Кожна координата представлена кортежем (x, y).

Висновок до розділу

У цьому розділі було розглянуто дані, які використовуються в програмному продукті, а також результати, які вони виробляють.

Був наданий опис структури масивів, які містять інформацію про різні аспекти зображення, такі як кольори, контури, центри та шляхи.

Додатково, було розглянуто можливість розширення структури масивів для забезпечення підтримки інших аспектів зображень, таких як текстура, освітлення та форма об'єктів. Це відкриває перспективи для подальшого розвитку програмного продукту та збільшення його функціональності.

Паралельно з розглядом даних, важливо враховувати й результати, які виробляються програмним продуктом за допомогою нейронної мережі.

Завдяки використанню нейронної мережі, програмний продукт може досягати високої точності та ефективності в розпізнаванні об'єктів, класифікації зображень. Нейронна мережа може вивчати складні закономірності в даних та автоматично витягувати корисні ознаки, що дозволяє їй ефективно вирішувати завдання обробки зображень.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Змістовна постановка завдання полягає у знаходженні оптимального маршруту від вихідної точки до цільової точки з урахуванням різних факторів безпеки та обмежень у навколишньому середовищі. Завдання ставиться у контексті переміщення об'єкта чи виконавця через певну територію чи сітку розташування, де потрібно мінімізувати ризики виникнення аварій, зіткнень із перешкодами чи інших небезпечних ситуацій.

При постановці завдання необхідно врахувати такі аспекти:

- Модель навколишнього середовища

Потрібно визначити структуру даних, яка є сіткою, графом або картою, де кожен вузол або вершина представляє місце розташування, а ребра або зв'язку визначають доступні шляхи між місцезнаходженнями. Це може бути двовимірна або тривимірна сітка, граф, що базується на дорожній мережі або інша модель, яка краще відповідає конкретному контексту.

- Початкова та кінцева точки

Необхідно вказати вихідне місце, звідки починається пошук безпечного шляху (початкова точка), а також цільове місце розташування, якого потрібно досягти (кінцева точка). Ці точки можуть бути задані координатами або іншим чином, що відповідає моделі навколишнього середовища.

- Фактори безпеки

Необхідно врахувати різні фактори безпеки, які можуть змінюватись в залежності від контексту. Це можуть бути обмеження швидкості руху, уникнення певних зон або об'єктів, облік пріоритетів, рівень ризику або інші критерії, що визначають безпеку маршруту.

- Обмеження та умови

У пошуку безпечного шляху можуть бути різні обмеження та умови, які необхідно врахувати при пошуку оптимального маршруту. Це може бути географічні обмеження, обмеження доступність певних шляхів чи об'єктів,

обмеження зміну швидкості чи інші чинники, які можуть проводити вибір шляху.

- Цільові показники

Завдання пошуку безпечного шляху може мати різні цільові показники, які потрібно враховувати під час вибору оптимального маршруту. Це може бути мінімізація часу переміщення, мінімізація витрат або максимізація безпеки залежно від конкретного завдання та контексту.

У результаті змістовна постановка завдання пошуку безпечного шляху вимагає визначення моделі навколишнього середовища, початкової та кінцевої точок, урахування факторів безпеки, обмежень та цільових показників. Це дозволяє сформулювати завдання таким чином, щоб знайти оптимальний маршрут, враховуючи безпеку та обмеження у навколишньому середовищі.

3.2 Математична постановка задачі

Нехай G = (V, E) – орієнтований чи неорієнтований граф, де V – безліч вершин (місцеположень), E – безліч ребер (зв'язків) між вершинами.

Нехай s – вихідна вершина (початкова точка), t – цільова вершина (кінцева точка).

Кожне ребро $e \in E$ має вагу w(e), яка відображає вартість або безпеку проходження через це ребро.

Завдання полягає у знаходженні шляху P = (v1, v2, ..., vk) з вершини s до вершини t з мінімальною вартістю або максимальною безпекою. Шлях P повинен відповідати заданим критеріям безпеки, таким як уникнення небезпечних зон, облік обмежень швидкості або інших параметрів.

Вхідні дані:

- Граф G = (V, E), де V – безліч вершин (місцеположень), E – безліч ребер (зв'язків) між вершинами.
- Початкова вершина $s \in V$ - вихідна точка (початкове розташування).
- Цільова вершина $t \in V$ - кінцева точка (цільове розташування).

- Ваги ребер w(e) для кожного ребра e ∈ E, що відображають вартість або безпеку проходження через нього.
- Додаткові критерії безпеки чи обмеження, які мають бути враховані під час вибору шляху.

Вихідні дані:

- Шлях P = (v1, v2, ..., vk) з вершини s у вершину t з мінімальною вартістю або максимальною безпекою, де кожна вершина vi ∈ V є місцем, яке необхідно відвідати в порядку прямування.
- Додаткові властивості шляху, такі як загальна вартість чи безпека шляху.

3.3 Опис методу розв’язання

Використовуючи алгоритми на графі, такі як A*, Дейкстри або Флойда-Уоршелла, здійснюється перегляд та оцінка вартості або безпеки шляху від вихідної вершини s до інших вершин графа.

На основі цих оцінок вибирається оптимальний шлях із урахуванням заданих критеріїв безпеки. Додаткові техніки та евристики можуть бути застосовані для оптимізації процесу пошуку та врахування конкретних обмежень чи критеріїв безпеки.

Однак, використання інтелектуального боту в процесі пошуку найбезпечнішого шляху може мати деякі переваги. Замість того, щоб обмежуватись лише статичними вагами ребер та критеріями безпеки, він може враховувати динамічні фактори, які впливають на безпеку шляху, такі як наявність аварійних ситуацій.

Бот може бути навчений на великій кількості даних, що включають інформацію про безпеку шляхів у різних умовах. Він може вивчити складні залежності та особливості, які можуть бути недосяжними для класичних алгоритмів на графі. Це дозволяє робити більш точні прогнози та приймати обґрунтовані рішення щодо безпеки шляху.

Одним із важливих переваг його використання є гнучкість та адаптивність. Після навчання він може бути легко застосований до різних сценаріїв та змінюваних умов.

3.4 Обґрунтування методу розв’язання

Розв'язання задачі знаходження безпечного шляху з використанням інтелектуального боту може бути кращим з кількох причин:

1. Навчання на основі даних

Бот навчається на великих обсягах даних, що дозволяє йому отримувати складні закономірності та особливості, які можуть бути непомітними для людини. У разі знаходження безпечного шляху, він може виявляти приховані ознаки чи зразки, пов'язані з безпекою, які може бути складно визначити вручну або за допомогою класичних алгоритмів.

2. Облік контексту

Бот може враховувати ширший контекст та взаємодію між різними факторами. Наприклад, він може аналізувати дані про трафік, погоду, час доби, географічні особливості та інші параметри для прийняття більш поінформованих рішень. Це дозволяє врахувати різноманітні фактори, які можуть впливати на безпеку шляху.

3. Автоматизація та швидкість

Бот здатен обробляти інформацію з високою швидкістю, що робить його ефективним для реального часу. Це особливо важливо для завдання знаходження безпечного шляху, де оперативне ухвалення рішень може бути критично важливим для безпеки користувачів.

4. Гнучкість і адаптивність

Інтелектуальний бот може бути легко налаштованим для обліку нових даних та змін у навколишньому середовищі. При оновленні даних та перенавченні можна врахувати нові ризики та фактори, щоб зберігати актуальність рішень.

3.5 Обґрунтування використання нейронної мережі в задачі

Нейронні мережі широко використовуються в задачах комп'ютерного зору, включаючи розпізнавання об'єктів, сегментацію зображень, виявлення шляху та багато інших. В даному випадку, нейронна мережа використовується для навчання моделі, яка здатна знаходити шлях між жовтими та червоними колами на зображеннях.

Зображення проходять через кілька згорткових шарів, що дозволяє виявити різні функціональні особливості зображень, такі як границі, форми та текстури. Згорткові шари допомагають виділити важливі ознаки зображень, які можуть бути корисними для пошуку шляху.

Далі воно проходить через повнозв'язані шари. Це дозволяє моделі використовувати виділені ознаки для прийняття рішень та передбачення вихідного шляху. Повнозв'язані шари здатні моделювати складні взаємозв'язки між ознаками та здійснювати розрахунки, необхідні для визначення шляху.

Мережа використовує активаційні функції, такі як ReLU, для впровадження нелінійності у модель. Це дозволяє нейронній мережі моделювати складні нелінійні відношення між ознаками зображень та забезпечує більшу гнучкість у прийнятті рішень.

$$f(x) = x^+ = \max(0, x)$$

Рисунок 3.1 – Формула ReLU

Модель навчається за допомогою алгоритму оптимізації Adam, який використовує градієнтний спуск для знаходження оптимальних значень параметрів мережі. Цей підхід дозволяє моделі автоматично налаштовувати ваги та зміщення, щоб мінімізувати втрати та забезпечити кращу продуктивність.

Далі вже навчена модель використовується для знаходження шляху між жовтими та червоними колами на зображеннях. Вона використовує алгоритм A* для ефективного пошуку оптимального шляху. Модель враховує геометричні

відстані між колами та використовує їх для прийняття рішень про найкоротший шлях.

Загалом, використання нейронної мережі дозволяє моделі автоматично вивчати залежності між вхідними зображеннями та вихідними шляхами. Це може бути корисно в задачах, де вручну налаштувати правила та евристики для пошуку шляху є складно або незручно.

Висновок до розділу

У цьому розділі було описано постановку задачі з використанням зрозумілого та математичного формалізму. Було надано детальний математичний опис моделі, яка використовується для розв'язання задачі.

Далі були розглянуті різні можливі структури даних та функції втрат, які можуть бути використані для задачі класифікації. Було обговорено переваги та недоліки різних підходів до реалізації рішення задачі.

В цьому розділі було звернуто увагу на важливі аспекти задачі та було розглянуто різні аспекти вибору моделі та алгоритмів для досягнення оптимального розв'язку. Описані підходи дозволяють краще зрозуміти сутність задачі та вибрати оптимальний підхід до її розв'язання.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до технічного забезпечення

Вимоги до технічного забезпечення для пошуку безпечного шляху можуть включати наступні аспекти:

1. Обчислювальні ресурси

Для ефективного вирішення завдання потрібні достатні обчислювальні ресурси, такі як процесорна потужність та оперативна пам'ять. Великі та складні графи можуть вимагати більше ресурсів для обробки та зберігання даних.

2. Алгоритмічна ефективність

Гарний вибір алгоритмів та евристик може значно вплинути на ефективність розв'язання задачі. Потрібно використовувати алгоритми з низькою обчислювальною складністю та оптимальним використанням пам'яті, щоб забезпечити швидке та точне рішення.

3. Графічний інтерфейс та візуалізація

Для зручності взаємодії з користувачем та наочної візуалізації розв'язання задачі може знадобитися розробка графічного інтерфейсу. Це дозволяє користувачеві вводити дані, переглядати результати та візуалізувати знайдені шляхи на графі чи карті.

4. Моделювання навколишнього середовища

Для точного моделювання навколишнього середовища, включаючи територію, перешкоди, об'єкти та інші фактори безпеки, може знадобитися використання спеціалізованих інструментів або бібліотек. Це дозволяє створити точну модель, що базується на реальних даних або симуляціях.

5. Інтеграція з іншими системами

Залежно від конкретної задачі та сценарію використання, може знадобитися інтеграція вирішення пошуку безпечного шляху з іншими системами або програмами. Наприклад, інтеграція із системами управління рухом, навігаційними системами чи системами моніторингу безпеки.

6. Масштабованість

Якщо завдання вимагає обробки великих обсягів даних або роботи в реальному часі, потрібна масштабована архітектура та ресурси, щоб забезпечити ефективну обробку даних та високу продуктивність системи.

7. Безпека та конфіденційність даних

Під час роботи з чутливими даними, такими як дані про навколишнє середовище або особисту інформацію, потрібно забезпечити високий рівень безпеки та конфіденційності даних. Це може включати шифрування даних, контроль доступу та інші заходи безпеки.

Вимоги до технічного забезпечення залежатимуть від конкретного завдання, його масштабу та контексту використання. Важливо вибрати відповідні технічні рішення та інструменти, щоб забезпечити ефективне та надійне вирішення задач пошуку безпечного шляху.

4.2 Вибір мови програмування та технологій розробки

Було обрано мову програмування Python для реалізації програми пошуку безпечного шляху для пішоходів. Python є популярною мовою програмування, яка відома своєю простотою, зрозумілістю синтаксису та широким спектром наявних бібліотек [21].

Використання Python дозволяє швидко реалізувати функціональність програми та забезпечити її ефективну роботу. Python має велику спільноту розробників, що сприяє наявності багатофункціональних бібліотек та фреймворків, які можна використовувати для розробки програми пошуку безпечного шляху. Для реалізації програми можна скористатись такими бібліотеками :

cv2: Бібліотека OpenCV (cv2) забезпечує можливості обробки зображень та комп'ютерного зору. Вона може бути використана для аналізу відео або зображень, наприклад, для виявлення перешкод на шляху пішохода або аналізу дорожніх знаків.

numpy: Бібліотека NumPy (numpy) надає потужні функції для роботи з масивами та матрицями даних. Вона може бути використана для ефективної обробки числових даних, таких як розрахунок відстаней між точками або ваги маршрутів.

heapq: Бібліотека Heapq надає інструменти для роботи зі структурами даних, що підтримують пріоритети. Вона може бути використана для ефективного вибору найкращих маршрутів залежно від їхньої безпеки.

Використання цих бібліотек та інструментів дозволить розробити потужну та функціональну програму пошуку безпечного шляху для пішоходів. Python є зручним інструментом для розробки програмного забезпечення та має великий вибір інструментів та бібліотек для реалізації різноманітних функцій програми.

4.3 Засоби розробки

Бібліотеки і інструменти, такі як NumPy, Tkinter, OpenCV (cv2), PyCharm і Heapq, грають важливу роль у розробці програмного продукту.

Бібліотека NumPy найбільше підходить для ведення обчислень, досліджень, підтримує матриці, багатовимірні масиви і має великий математичний функціонал для проведення робіт та розрахунків [16].

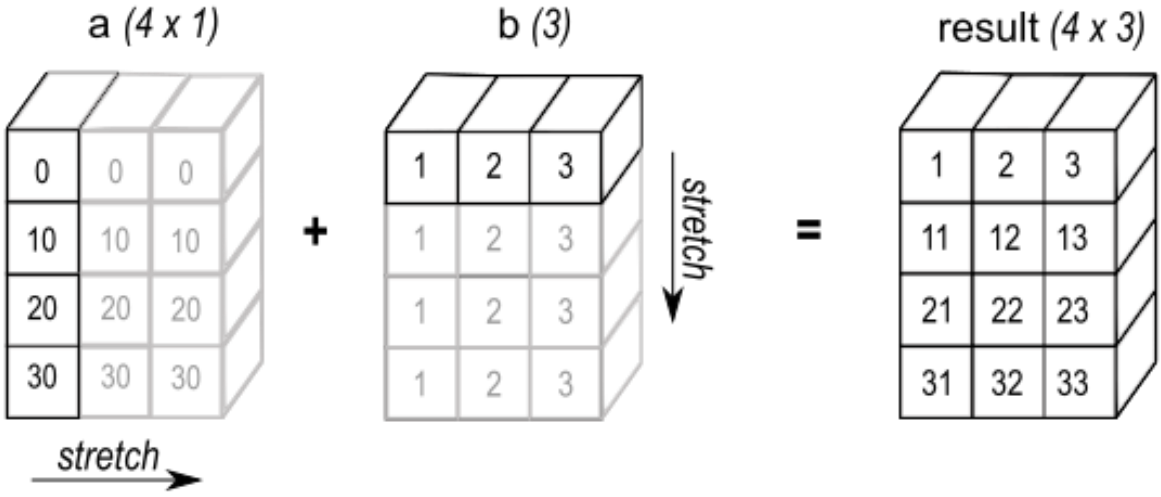

Рисунок 4.1 – Автоматична зміна розмірності масивів NumPy

Особливу увагу, у розрізі поставленої задачі, представляють можливості NumPy у сфері обробки зображень, переведення їх у матрицю, роботою як з кольоровою, так і чорно-білою графікою

Для роботи з графічним відображенням проекту найкраще підходить вбудований модуль Tkinter. Він має функцію самостійного підстроювання зовнішнього вигляду оболонки вікна під стиль операційної системи . Бібліотека використовується у разі потреби створення додатків користувача.

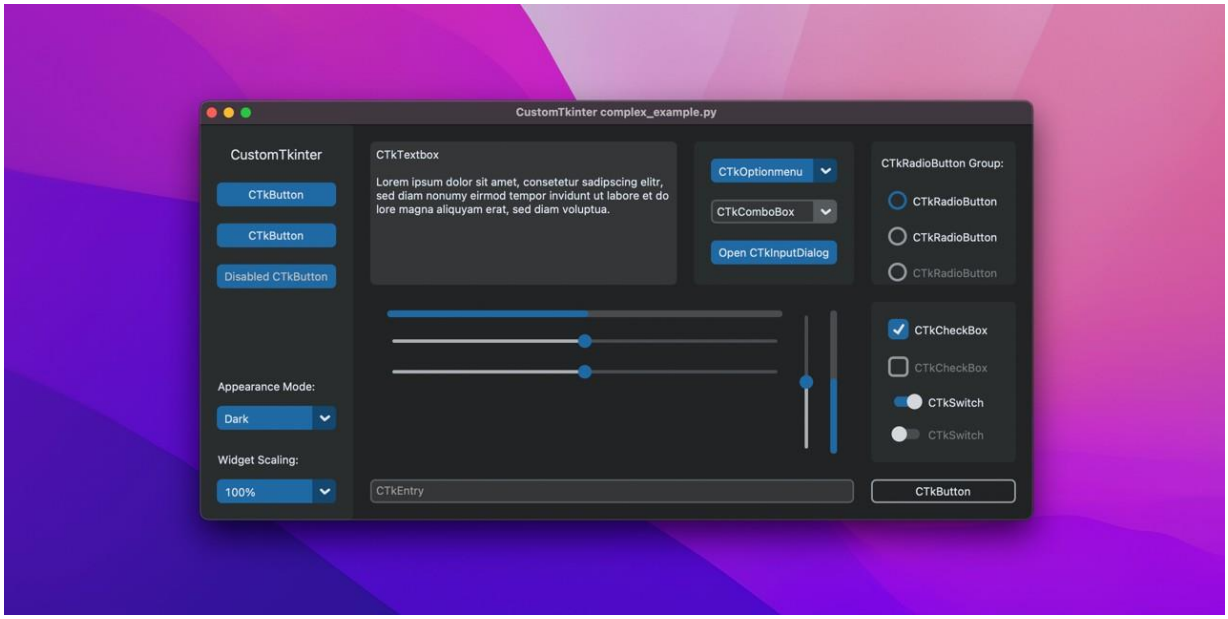

Рисунок 4.2 - Tkinter

Бібліотека OpenCV – одна з найзручніших бібліотек машинного навчання, пов'язаних із графічним контентом, що мають відкритий код. Широта застосування її алгоритмів дозволяє використовувати OpenCV у вирішенні як простих, так і складних завдань. Серед них: відображення, читання, зміна параметрів зображення (поворот, зміна габаритних значень, перетворення колірного простору), виділення меж об'єктів на зображенні [15].

Більш складними завданнями, пов'язаними з машинним навчанням, є алгоритми розпізнавання облич, зіставлення об'єктів для подальшого зшивання або розпізнавання, а також визначення місця знаходження об'єкта, що знаходиться в русі, для відстеження траєкторії його переміщення в просторі.

У контексті поставленого завдання OpenCV відіграє важливу роль у відображенні та перетворенні вступних графічних даних для подальшої роботи інтелектуального бота.

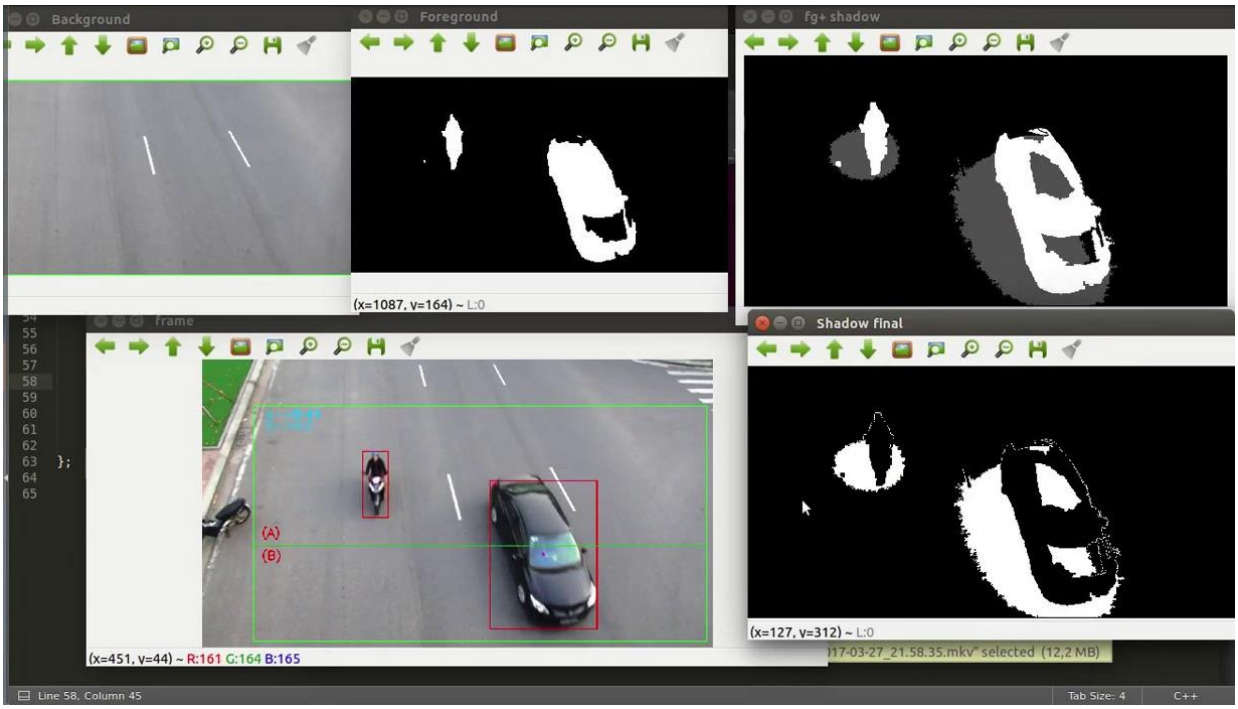

Рисунок 4.3 – Розпізнаання об’єктів за допомого OpenCV

Як середовище розробки для Python під реалізацію поставлених завдань було обрано PyCharm, як найефективніший інструмент із широким спектром функцій [19].

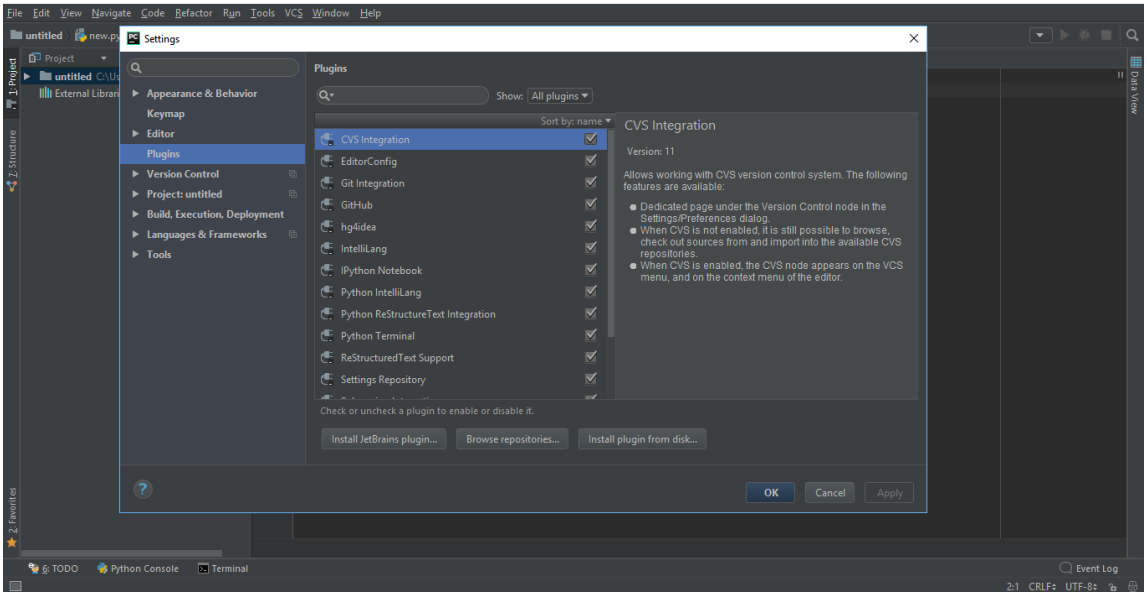

Рисунок 4.4 – інтерфейс PyCharm

На сьогоднішній день PyCharm є найкращим вибором при створенні складних потужних проектів, що стосуються виконання наукових розрахунків, візуалізації масивів даних і т.д.

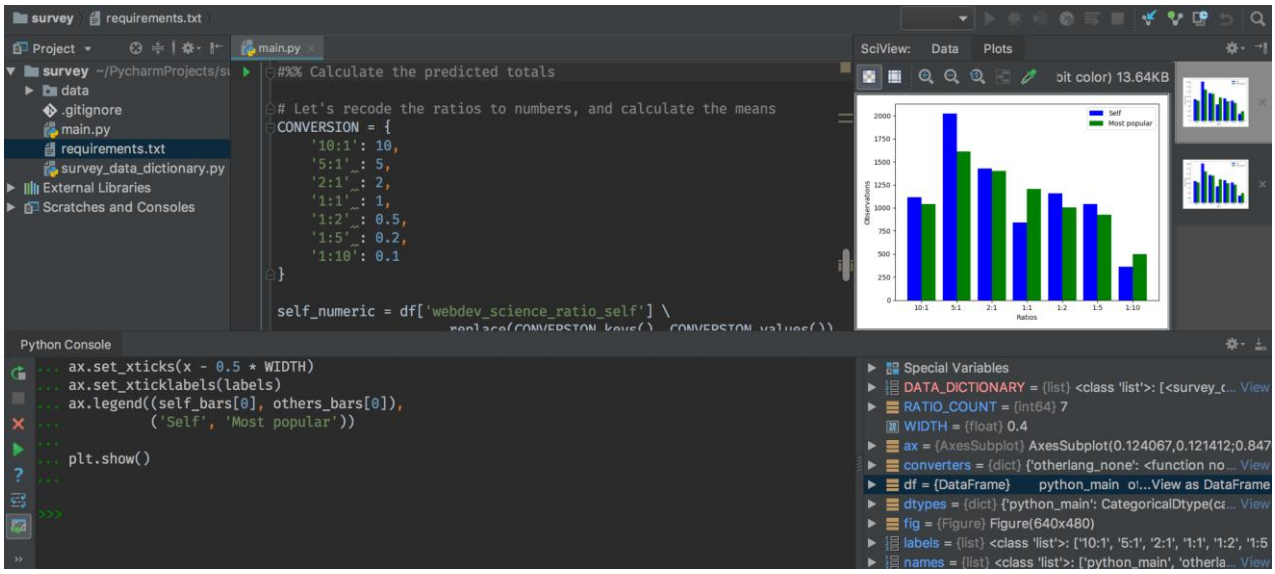

Рисунок 4.5 – Інструмент для наукових обчислень PyCharm

Модуль Heapq є простим і ефективним інструментом, що дозволяє вдосконалити роботу з алгоритмами, що працюють з даними, структурованими за принципом «черги купи».

Здатність витягувати максимальний або мінімальний елемент пріоритетної черги робить модуль максимально зручним для вирішення поставленого завдання.

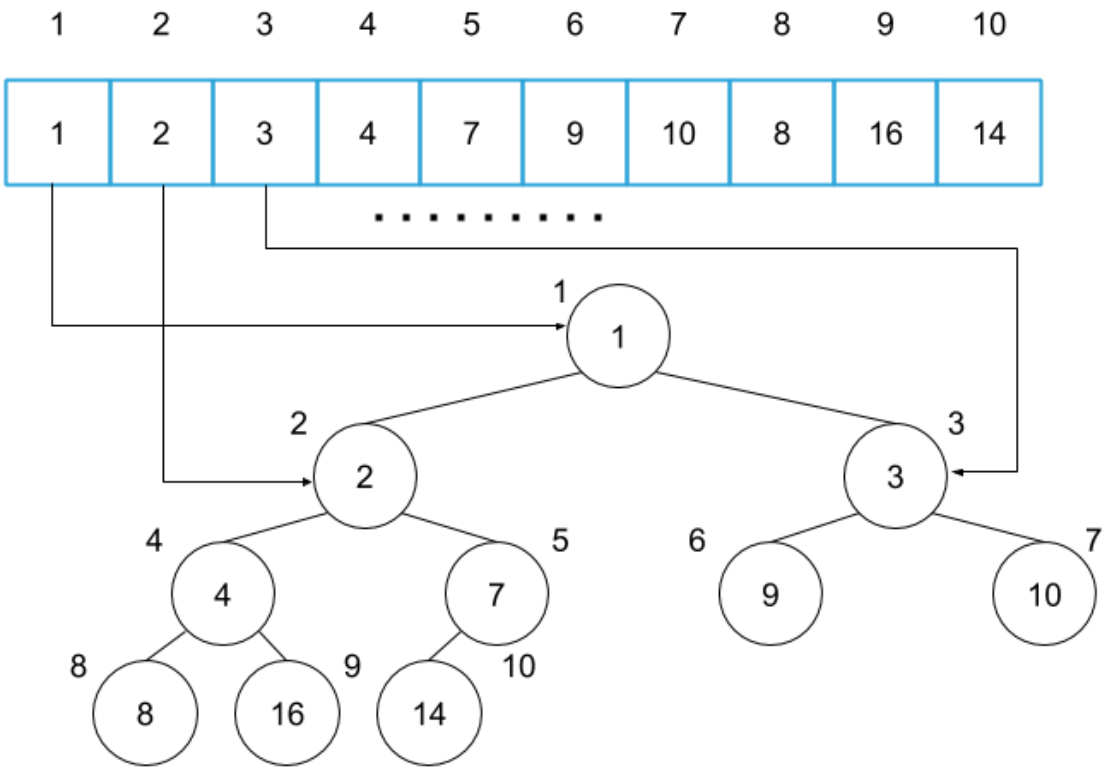

Рисунок 4.6 – Побудова бінарного дерева heapq

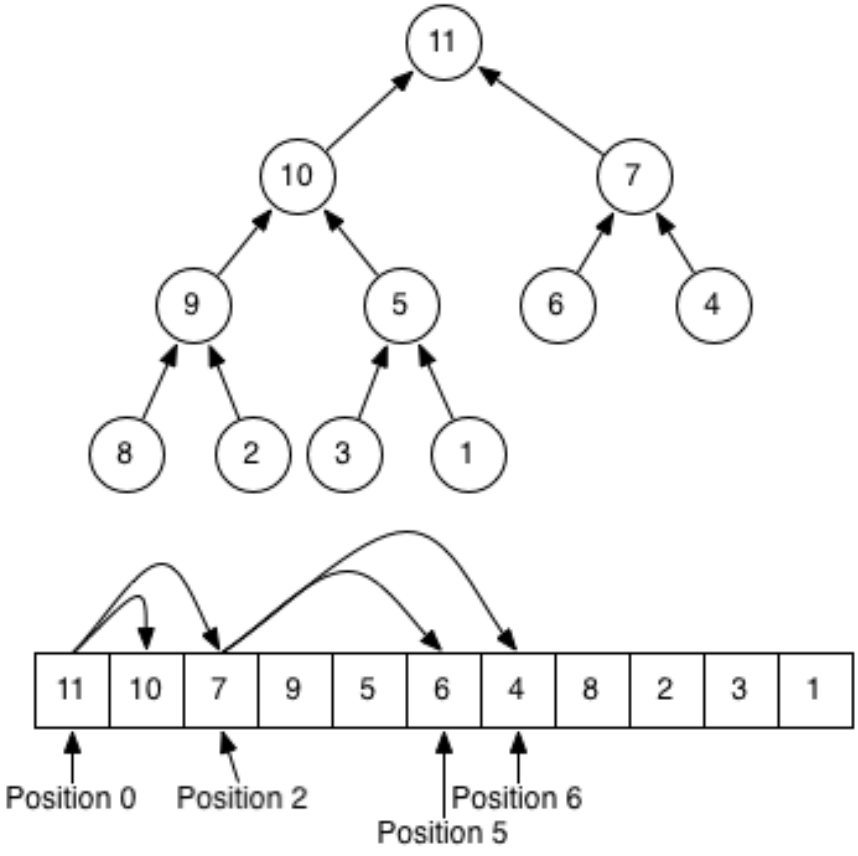

Рисунок 4.7 – Знаходження послідовностей heapq

Використання цих засобів та бібліотек у поєднанні з PyCharm дозволяє розробникам ефективно працювати над проектом, обробляти дані, створювати інтерактивні інтерфейси та виконувати обробку зображень та відео. Вони створюють потужне середовище для розробки програмного продукту з використанням нейромереж та інших алгоритмів машинного навчання.

Бібліотека PyTorch є однією з найпопулярніших бібліотек для роботи з нейронними мережами та глибоким навчанням. Вона надає зручний та ефективний інтерфейс для виконання різних операцій у галузі машинного навчання, таких як побудова та тренування нейронних мереж, обробка даних, робота з тензорами та оптимізація моделей.

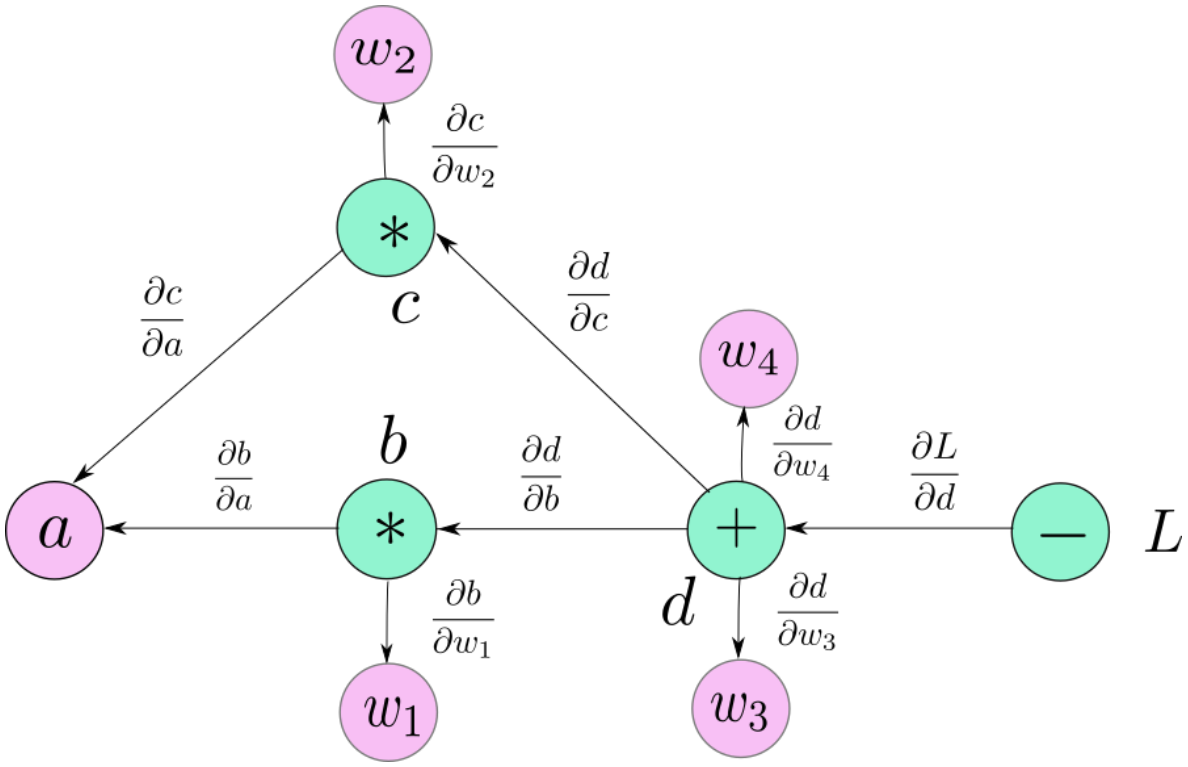

Рисунок 4.8 – обчислювальний граф PyTorch

Одним з пакетів, який входить до складу PyTorch, є torchvision. Це додаткова бібліотека, спеціалізована на комп'ютерному зорі та обробці зображень. Вона надає набір інструментів для завантаження, попередньої обробки та розширення наборів даних зображень, а також для побудови та тренування моделей комп'ютерного зору. Основні функціональні можливості

torchvision це завантаження та попередня обробка даних зображень, колекція популярних моделей комп'ютерного зору, розширення набору даних та візуалізація результатів.

В цілому, torchvision робить роботу з обробкою зображень та побудовою моделей комп'ютерного зору більш зручною та ефективною, спрощуючи процес розробки та експериментів з нейронними мережами для комп'ютерного зору.

4.4 Реалізація алгоритму пошуку безпечного шляху

Алгоритм є основною складовою програми пошуку безпечного шляху і відповідає за знаходження маршруту. Нижче наведено алгоритм пошуку безпечного шляху:

Початок

Імпортувати необхідні бібліотеки

Визначити клас Node

- position
- parent
- g
- h
- f

Визначити функцію find_yellow_and_red_circles з аргументом image_path

- Завантажити зображення з image_path
- Перетворити зображення в HSV
- Визначити межі для жовтого кольору
- Визначити межі для червоного кольору
- Створити маски для жовтих об'єктів
- Створити маски для червоних об'єктів

Застосувати морфологічні операції на масках

Знайти контури жовтих об'єктів

Знайти контури червоних об'єктів

Отримати центри жовтих об'єктів

Отримати центри червоних об'єктів

Створити граф

Для кожного центру об'єкта:

- Знайти два найближчих сусіда
- Додати їх до графа

Визначити початкову і кінцеву точки шляху

Виконати алгоритм для знаходження шляху

Якщо шлях знайдений:

- Побудувати шлях від кінцевої до початкової точки
- Вивести шлях на зображенні

Завершити програму

4.5 Архітектура програмного забезпечення

Архітектура коду дипломної роботи базується на обробці зображень і застосуванні алгоритму для пошуку шляху між жовтими та червоними колами на зображенні. Основні компоненти архітектури включають такі елементи:

1 Завантаження та обробка зображення:

- Зображення завантажується з вказаного шляху за допомогою бібліотеки OpenCV
- Зображення перетворюється з формату BGR до HSV для полегшення обробки кольорів

2 Визначення меж кольорів:

- Визначаються нижня та верхня межі для жовтого та червоного кольорів в просторі кольорів HSV
- Ці межі використовуються для створення масок, що виділяють жовті та червоні об'єкти на зображенні

3 Морфологічна обробка масок:

Використовуються морфологічні операції відкриття та закриття для зменшення шуму та з'єднання областей

4 Знаходження контурів об'єктів:

Використовуються функції з бібліотеки OpenCV для знаходження контурів жовтих та червоних об'єктів

5 Отримання центрів колів:

Для кожного контуру обчислюється мінімальне описуюче коло, а його центр додається до відповідного списку центрів

6 Побудова графу:

- Створюється порожній граф, в якому кожному центру кола ставиться у відповідність порожній список сусідів
- Для кожного центру обчислюються відстані до інших центрів та додаються два найближчих центри до списку сусідів

7 Виконання алгоритму:

- Створюються списки відкритих та закритих вузлів
- Стартовий вузол додається до списку відкритих вузлів
- Поки список відкритих вузлів не порожній:
- Обирається вузол з найменшою оцінкою і переводиться до списку закритих вузлів
- Якщо цей вузол є кінцевим, побудований шлях повертається
- Для кожного сусіднього вузла:
- Якщо вузол вже в списку закритих вузлів, процес продовжується з наступним сусідом
- Розраховуються значення g, h та f для сусіднього вузла
- Якщо вузол вже в списку відкритих вузлів:
- Якщо нове значення g менше, ніж поточне значення g вузла, оновлюється поточний вузол
- В іншому випадку, сусідній вузол додається до списку відкритих вузлів

- Якщо список відкритих вузлів стає порожнім, алгоритм завершується без знайдення шляху

8 Візуалізація результату:

Побудований шлях відображається на вихідному зображенні шляхом малювання ліній між точками шляху.

9 Завершення програми.

4.5.1 Структура коду

Клас Node:

Використовується для представлення вузла у графі шляху.

Властивості position, parent, g, h та f визначають позицію вузла, батьківський вузол, значення функцій ваги шляху та оцінки ваги шляху від початкового вузла до цього вузла.

Метод __lt__ визначає порівняння вузлів за значенням f, що потрібно для використання у чергах з пріоритетом.

Функція find_yellow_and_red_circles(image_path):

- Завантажує зображення за вказаним шляхом
- Перетворює зображення з BGR в HSV для спрощення обробки кольорів
- Створює маски для виділення жовтих і червоних точок на основі заданих порогових значень в просторі кольорів HSV
- Застосовує морфологічні операції для видалення шуму та з'єднання областей в масках
- Знаходить контури для жовтих і червоних областей
- Визначає центри масових контурів і зберігає їх
- Створює граф, де кожному центру точки присвоюється список сусідніх центрів
- Виконує алгоритм пошуку шляху A*, щоб знайти найкоротший шлях між початковою і кінцевою точками у графі

- Малює знайдений шлях на зображенні
- Відображає зображення зі шляхом у вікні

Функція heuristic(node, goal):

- Обчислює евристичну оцінку (відстань) між вузлом node та цільовим вузлом goal
- Використовує евклідову відстань між координатами вузлів у просторі

Функція astar(graph, start, end):

- Реалізує алгоритм пошуку шляху A*
- Використовує пріоритетну чергу для ефективного оброблення вузлів з найменшою оцінкою ваги шляху
- Розраховує значення функцій ваги шляху та оцінки ваги шляху для кожного вузла
- Перевіряє сусідні вузли та оновлює їх значення, якщо знайдено кращий шлях
- Повертає знайдений шлях або None, якщо шлях не знайдено

4.5.2 Специфікація функцій

Таблиця 4.1 – Специфікація функцій програми

Назва	Вхідні параметри	Вихідні параметри	Опис
find_yellow_and_red_circles	image_path	Зображення зі знайденим шляхом	Знаходить жовті та червоні кола на зображенні, обчислює найкоротший шлях між ними та малює цей шлях на зображенні.

astar	Graph, список сусідніх точок, start, end	Повертає список точок, що утворюють безпечний шлях	Виконує алгоритм пошуку шляху у заданому графі з початковою точкою start та кінцевою точкою end.
imread	Зображення	Повертає масив пікселів зображення у форматі BGR	Зчитування зображення з вказаного шляху image_path.
cvtColor	Масив пікселів зображення у форматі BGR	Масив пікселів зображення у форматі HSV	Перетворює простір кольорів з BGR в HSV
inRange	HSV-зображення	Маски для червоних, жовтих та чорних кольорів	Створення маски шляхом виділення пікселів
morphologyEx	image, operation, kernel	Зображення без шуму	Застосовує морфологічні операції
findContours	image, mode, method	Список контурів і їх ієрархію	Пошук контурів в бінарному зображенні
line	image, start_point, end_point, color, thickness	Шлях між точками.	Малювання шляху на зображенні
array	Об’єкт	-	Перетворення вхідного об'єкту на масив NumPy

norm	vector	-	Обчислення норми (відстані) вектора
ones	-	Масив з вказаною формою shape та типом даних dtype	Створює масив, заповнений одиницями

4.6 Опис використаних бібліотек

cv2 - це бібліотека OpenCV (Open Source Computer Vision Library), яка надає набір функцій для роботи з комп'ютерним зором і обробки зображень. Вона використовується для завантаження, обробки та відображення зображень, а також для знаходження контурів і морфологічної обробки.

numpy (np) - це бібліотека для мови програмування Python, яка надає підтримку для роботи з багатовимірними масивами і матрицями. Вона використовується для обчислення математичних операцій та маніпуляцій зі зображеннями, зокрема для роботи з кольоровим простором HSV і обчислення відстаней між точками на зображенні.

heapq - це модуль Python, який надає функціональність для роботи з чергами з пріоритетом. В даному коді використовується для реалізації алгоритму для пошуку шляху.

Висновок до розділу

В даному розділі було розглянуто програмне та технічне забезпечення проекту. Було надано детальний огляд структури та процесу роботи системи. Описано використані засоби розробки та бібліотеки. Також було розглянуто архітектуру програмного забезпечення та надано діаграми класів, послідовності та компонентів. У розділі наведено детальну специфікацію функцій та надано інформацію про вхідні та вихідні дані функцій, а також їх опис.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користувачу перед запуском програми необхідно підготувати вхідні дані у вигляді зображення з позначками. На цьому зображенні жовтим кольором відмічається початок та кінець маршруту, червоним кольором позначаються безпечні зони, а чорним кольором - зони, де спостерігалася небезпека для користувача. На Рисунку 5.1 наведено приклад вхідного зображення з такими позначками.

Рисунок 5.1 – Вхідне зображення з позначками

Під час запуску програми користувачу пропонується вибрати файл, який буде використовуватися для пошуку шляху (Рисунок 5.2). Цей файл містить вхідні дані, необхідні для алгоритму.

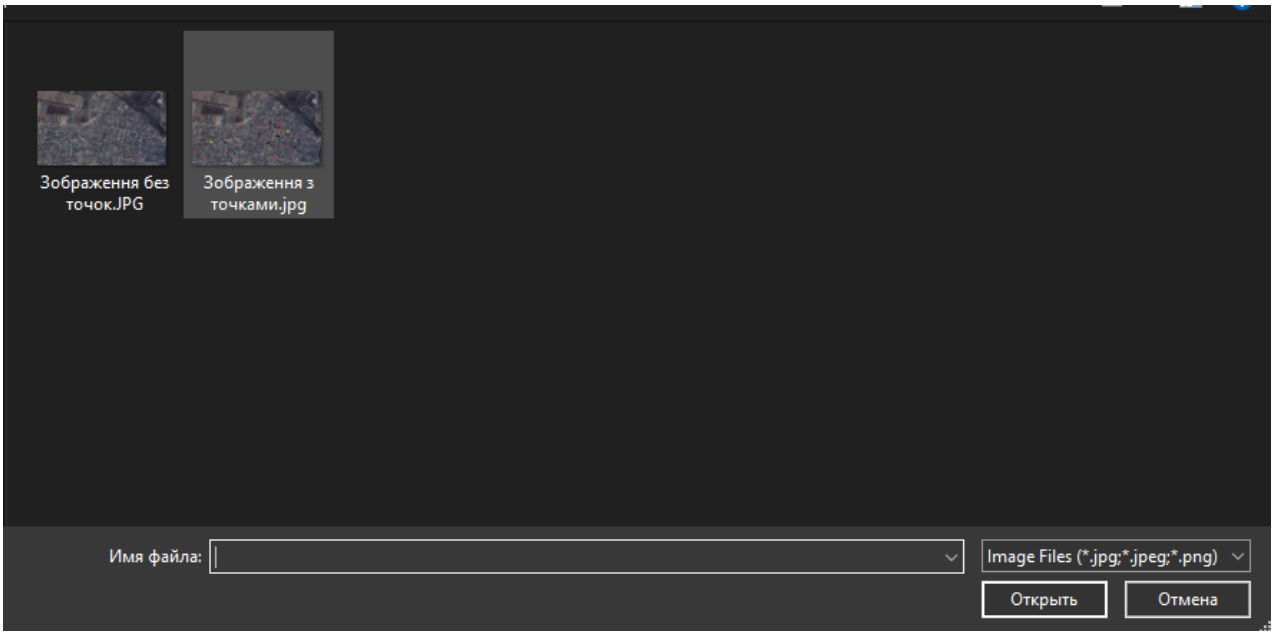

Рисунок 5.2 – Вибір файлу для знаходження шляху

Після обробки вхідних даних програма надасть користувачу готовий маршрут, який допоможе зменшити ризик потрапити у небезпеку. На Рисунку 5.3 показано приклад отриманого маршруту.

Рисунок 5.3 – Виведення результату пошуку шляху

Також під час виконання програми буде навчатись створена нейромережа яка збільшить швидкість обробки зображення та знаходження маршруту.

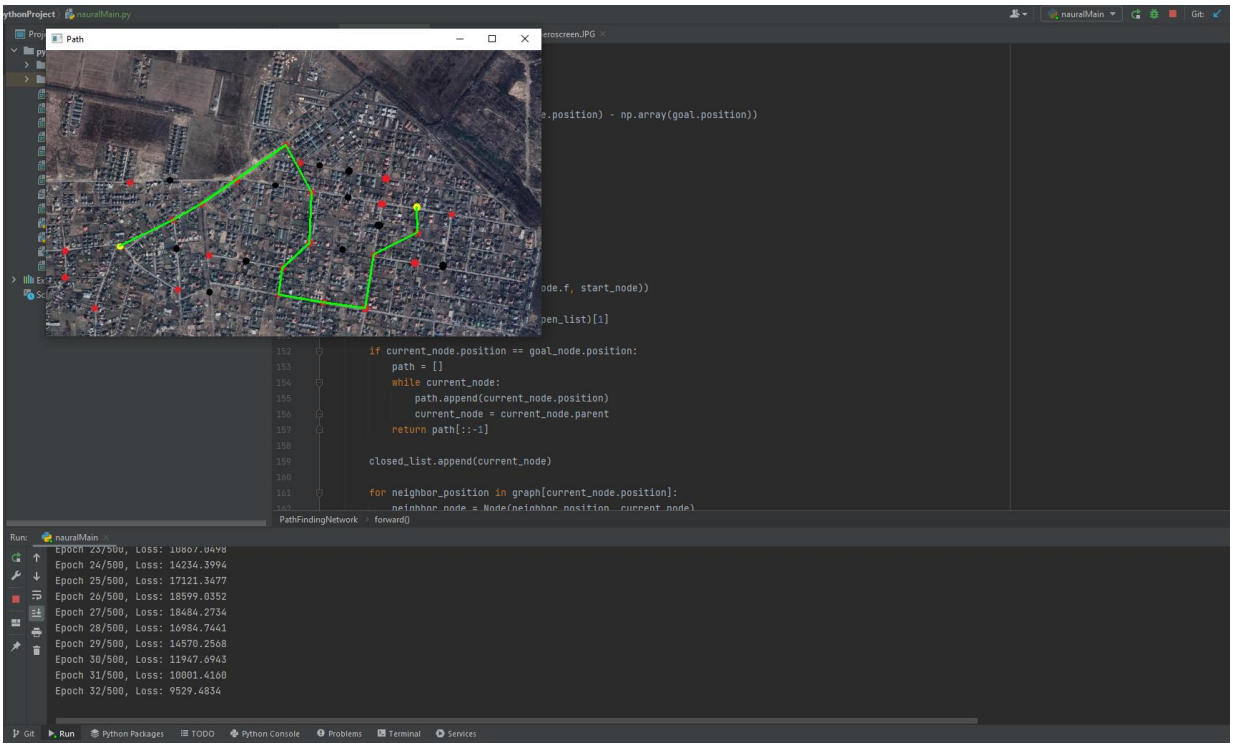

Рисунок 5.4 – Навчання нейромережі PathFindingNetwork

5.2 Випробування програмного продукту

Випробування програмного продукту є необхідною складовою для забезпечення якості, надійності та відповідності вимогам. Цей процес включає систематичну перевірку програми з метою виявлення помилок та підтвердження правильного функціонування. Випробування допомагає покращити досвід користувачів, зменшити ризики та підвищити конкурентоспроможність продукту.

5.2.1 Мета випробувань

Мета випробувань для програми пошуку безпечного шляху полягає у перевірці правильності та ефективності алгоритму пошуку. Це включає виявлення та усунення помилок, перевірку коректності знаходження безпечного шляху, а також оцінку швидкості та надійності пошуку. Головна мета полягає у

забезпеченні, щоб програма надавала точні та оптимальні шляхи, уникати небезпек та забезпечувала надійність та стабільність у випадку різних умов та ситуацій.

5.2.2 Результати випробувань

Після проведення повного тестування програми для пошуку безпечного шляху, включаючи всі функції, було отримано дуже задовільні результати.

Програмний продукт працює стабільно і здатний ефективно знаходити оптимальний безпечний шлях у великому спектрі ситуацій.

Доступ до вихідного коду програмного продукту надається у Додатку А, що дозволяє іншим дослідникам та розробникам ознайомитись з деталями реалізації алгоритмів пошуку безпечного шляху. Це сприяє зростанню взаємодії та спільній роботі у галузі безпечного транспортування та надає можливість для подальшого розвитку програми.

Результати випробувань програми можна знайти у таблицях 5.1 - 5.5. Аналіз цих результатів дозволяє оцінити продуктивність програмного продукту та порівняти його з іншими аналогами на ринку. Дані таблиці включають в себе результат та мету випробувань для різних вхідних сценаріїв. Вони є важливою частиною оцінки якості та функціональності програми, і допоможуть у вдосконаленні її роботи у майбутньому.

Таблиця 5.1 – Тестування введення картинки

Мета випробування	Перевірка коректності завантаження вибраного користувачем файлу
Вхідні дані	Файл картинка в форматі .jpg
Схема проведення тесту	Натиснути на кнопку запуску коду
Очікуваний результат	На екрані з’явиться вікно з вибраним файлом

Результат	Створено вікно де користувач може побачити файл який вибрав

Таблиця 5.2 – Тестування коректності виведення маршруту

Мета випробування	Перевірка правильності виведення безпечного маршруту на мапі
Вхідні дані	Список точок, початкова точка та кінцева, зображення
Схема проведення тесту	Вибрати зображення після запуску проекта
Очікуваний результат	Повністю побудований маршрут який уникає небезпечних зон
Результат	В створене вікно виводиться зображення зі шляхом який прокладено через безпечні зони

Таблиця 5.3 – Тестування на неправильно вибраний файл

Мета випробування	Перевірити працездатність програми при введенні неправильного файлу
Вхідні дані	Будь який файл який не відповідає запиту програми
Схема проведення тесту	Запустити програму
Очікуваний результат	Програма видає помилку

Результат	Після проведення випробування, програма видала помилку в консолі

Таблиця 5.4 – Тестування на кількість початкових та кінцевих точок

Мета випробування	Перевірка на працездатність програми, якщо шляхів буде декілька
Вхідні дані	Зображення з додатковими кінцевими та початковими точками
Схема проведення тесту	Після запуску програми вибрати файл з кінцевими та початковими точками
Очікуваний результат	Програма видає помилку
Результат	Програма знаходить шлях між випадково вибраними початковими та кінцевими точками

Таблиця 5.5 – Тестування на кількість небезпечних зон

Мета випробування	Перевірити на уникнення небезпечних зон
Вхідні дані	Зображення з небезпечними зонами та початковою і кінцевою точками
Схема проведення тесту	Запустити програму та вибрати файл з небезпечними зонами
Очікуваний результат	Ігнорування небезпечних зон та виведення шляху

Результат	Після запуску програми та обрання файлу для тестування, було знайдено безпечний шлях з уникненням небезпечних зон

Висновок до розділу

У цьому розділі було підготовлено керівництво користувача, яке описує можливі способи взаємодії користувача з інтерфейсом системи та методи їх виконання. Також було проведено тестування функціоналу програми, зокрема пошуку безпечного шляху та виведення результатів. Були визначені цілі випробувань та загальні принципи, за якими вони проводилися.

В цілому, проведене тестування функціоналу сприяє поліпшенню взаємодії користувача з системою, забезпечує більшість необхідної інформації та допомагає виявити можливі проблеми. Це робить програму більш зручною та ефективною для користувачів, допомагаючи їм досягати своїх цілей у найкращий спосіб.

Крім того, підготовлене керівництво користувача відіграє важливу роль у забезпеченні ефективного використання програмного продукту. Воно надає детальні інструкції щодо взаємодії з інтерфейсом системи, пояснює різні функціональні можливості та навчає користувачів ефективно їх використовувати.

ВИСНОВКИ

Під час реалізації цього проекту було розроблено програму для знаходження безпечного маршруту. В розділі "Загальні Положення" було детально описано предметну область проекту, включаючи варіанти використання та функціональні можливості програми. Використання зображень, перетворення кольорових просторів, створення масок та побудова безпечного маршруту - це основні етапи роботи програми.

Також було проведено аналіз існуючих рішень для пошуку безпечного маршруту, з'ясовано їх особливості та відмінності від розробленого проекту. Результатом є програмний продукт, який використовує менші обчислювальні ресурси порівняно з альтернативами.

У розділі "Інформаційне забезпечення" розглянуті вхідні та вихідні дані системи, а лекції охоплюють широкий спектр тем, що відповідає завданням програми знаходження безпечного маршруту.

В розділі "Математичне забезпечення" була сформульована задача, математично описано роботу програми та з'ясовано її точність. Також були розглянуті доступні структури для обробки зображень.

В розділі "Програмне та технічне забезпечення" було надано опис використаних засобів розробки та вимог до технічних характеристик для реалізації програми. Були розроблені діаграми класів, компонентів та послідовності, а також надано характеристику функцій програми.

У технологічному розділі була описана взаємодія користувача з інтерфейсом, проведено тестування правильності роботи різних засобів та оновлення результатів.

Розроблений програмний продукт може бути використаний користувачами через графічний інтерфейс або як складова більших проектів через виклик функцій без інтерфейсу. Проект здатний не тільки знаходити
безпечний маршрут, але й враховувати небезпечні зони для знаходження найкоротшого та оптимального маршруту. Розробка включає функції для

завантаження файлів, що дозволяє роботі програми працювати з різними розмірами та кількістю перешкод, і може бути застосована в різних галузях.

Подальший розвиток проекту може включати розширення функціональності програми та більш широкі можливості використання.

Повертаючись до нашого проекту, можемо сказати, що розроблений програмний продукт має потенціал для подальшого розвитку. В майбутньому він може бути використаний для покращення діяльності української армії та забезпечення безпеки її жителів шляхом надання ефективних рішень щодо евакуації та бойових дій. Цей проект може стати важливим інструментом для оптимізації стратегічного планування та виконання завдань, сприяючи підвищенню ефективності та безпеки українських військових та цивільних осіб.

СПИСОК ДЖЕРЕЛ

1. Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. *Pearson Education, Inc.,Upper Saddle River, New Jersey 07458*, 2010. URL: https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf (date of access: 14.06.2023)
2. Richard Szeliski, Computer Vision: Algorithms and Applications, *Springer*, 2010. URL: https://www.cs.ccu.edu.tw/~damon/tmp/SzeliskiBook_20100903_draft.pdf (date of access: 14.06.2023)
3. Eric Matthes, Python Crash Course, *No Starch Press, Inc., San Francisco*, 2016. URL: https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/No.Starch.Python.Oct_.2015.ISBN_.1593276036.pdf (date of access: 14.06.2023)
4. Wes McKinney, Python for Data Analysis. *O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol* , 2013. URL: https://bedford-computing.co.uk/learning/wp-content/uploads/2015/10/Python-for-Data-Analysis.pdf (date of access: 14.06.2023)
5. Joe Minichino, Joseph Howse, Learning OpenCV 3 Computer Vision with Python. Second Edition. *Packt Publishing Ltd., Birmingham B3 2PB, UK.*, 2015. URL: https://repository.unikom.ac.id/67052/1/Learning%20OpenCV%203%20Computer%20Vision%20with%20Python%20%28%20PDFDrive.com%20%29.pdf (date of access: 14.06.2023)
6. Michael Sipser, Introduction to the Theory of Computation. 3rd ed., *Cengage Learning, Boston, USA*, 2012. URL: http://staff.ustc.edu.cn/~huangwc/book/Sipser_Introduction.to.the.Theory.of.Computation.3E.pdf (date of access: 14.06.2023)
7. Kevin Wayne, Robert Sedgewick, Algorithms. Fourth edition., *Pearson Education, Inc., Boston, USA*, 2011. URL: https://bank.engzenon.com/tmp/5e7f6ee5-d4dc-4aa8-9b0a-42d3c0feb99b/6062caf3-c600-4fc2-b413-4ab8c0feb99b/Algorithms-4th-Edition.pdf

8. Alberto Artasanchez & Prateek Joshi, Artificial Intelligence with Python. Second Edition., *Birmingham B3 2PB, UK*, 2020. (235 c.) URL: https://library.samdu.uz/files/47c903d12731fd74e988ab1627d1b2b3_Artificial%20Intelligence%20with%20%20Python.pdf (date of access: 14.06.2023)
9. Michael Nielsen, Neural Networks and Deep Learning, *Sentdex*, *Kinsley Enterprises Inc.*, 2019. URL: https://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf (date of access: 14.06.2023)
10. Harrison Kinsley & Daniel Kukieła, Neural Networks from Scratch in Python, , URL: https://www.haio.ir/app/uploads/2021/12/Neural-Networks-from-Scratch-in-Python-by-Harrison-Kinsley-Daniel-Kukiela-z-lib.org_.pdf (date of access: 14.06.2023)
11. Effective PyCharm. Learn the PyCharm IDE with a Hands-on Approach, Michael Kennedy Matt Harrison, 2019 (181 c.)
12. Pedro Kroger, Modern Python Development With PyCharm, 2015. URL: https://raw.githubusercontent.com/kroger/books/master/PyCharmBook.pdf
13. Dipanjan Sarkar, Raghav Bali and Tushar Sharma, Practical Machine Learning with Python. A Problem-Solver's Guide to Building Real-World Intelligent Systems. *Raghav Bali, Bangalore, Karnataka, India Bangalore, Karnataka, India* , 2018, URL: https://library.kre.dp.ua/Books/2-4%20kurs/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%2B%20%D0%BC%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F/Python/practical-machine-learning-python-problem-solvers.pdf (date of access: 14.06.2023)
14. Jaydip Sen, Machine Learning. Algorithms, Models and Applications, *United Kingdom, IntechOpen*, 2021. URL: https://www.researchgate.net/publication/357632509_Machine_Learning_Algorithms_Models_and_Applications (date of access: 14.06.2023)
15. Документація OpenCV. URL: https://docs.opencv.org/ (date of access: 14.06.2023)

16. Документація Numpy. URL: https://numpy.org/doc/stable/ (date of access: 14.06.2023)
17. Документація Maps.me. URL: https://ru.maps.me/app/(date of access: 14.06.2023)
18. Документація Waze. URL: https://www.waze.com/ru/live-map/(date of access: 14.06.2023)
19. Документація PyCharm. URL: https://www.jetbrains.com/pycharm/ (date of access: 14.06.2023)
20. Документація Here WeGo. URL: https://wego.here.com/(date of access: 14.06.2023)
21. Офіційні документації Python. URL: https://docs.python.org/(date of access: 14.06.2023)

ДОДАТОК А

Текст програмного коду

GitHub