Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем
(повна назва)
Кафедра Комп’ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Розроблення програми побудови маршруту мобільного робота з використанням алгоритму D* lite
(тема)

Виконав:
здобувач 4 року навчання,
групи АКТСІ-22-1
Руслан БУРЯК
(власне ім’я, прізвище)

Спеціальність 151 Автоматизація та комп’ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва освітньої програми)

Керівник ст.викл. Дмитро ГУРІН
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАРБІ _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім’я, прізвище)

2026 р.

Я, Буряк Руслан Євгенович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

" 30 " червня 2026 р.  Руслан БУРЯК

Харківський національний університет радіоелектроніки

Факультет Автоматизації, комп'ютерно-інтегрованих технологій та систем
Кафедра Комп’ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 151 Автоматизація та комп’ютерно-інтегровані технології
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Системна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«27» квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
здобувачеві Буряку Руслану Євгеновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розроблення програми побудови маршруту мобільного робота з використанням алгоритму D* lite
затверджена наказом університету від 04 червня 2026 р. № 473 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 16.07.2026 р.
3. Вихідні дані до роботи
3.1 Карта середовища у вигляді двовимірної сітки;
3.2 Алгоритм побудови маршруту D* Lite.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ;
4.2 Аналіз сучасних мобільних роботів та алгоритмів побудови маршруту;
4.3 Розробка математичного та алгоритмічного забезпечення програми побудови маршруту мобільного робота;
4.4 Розробка програми побудови маршруту мобільного робота з використанням алгоритму D* lite;
4.5 Експериментальні дослідження та аналіз отриманих результатів;
4.6 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Графічний матеріал у вигляді презентації – 14 арк. ф. А4.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз сучасних мобільних роботів та алгоритмів побудови маршруту	27.04.26-05.05.26	виконано
2	Розробка математичного та алгоритмічного забезпечення програми побудови маршруту мобільного робота	06.05.26-26.05.26	виконано
3	Розробка програми побудови маршруту мобільного робота з використанням алгоритму D* lite	27.05.26-24.06.26	виконано
4	Оформлення пояснювальної записки	25.06.26-03.07.26	виконано
5	Подання роботи на нормоконтроль	04.07.26-05.07.26	виконано
6	Подання роботи на перевірку Інтернет-сервісом Unicheck	06.07.26-07.07.26	виконано
7	Подання роботи на рецензію	07.07.26-10.07.26	виконано
8	Подання роботи на підпис зав. кафедри	11.07.26-13.07.26	виконано
9	Подання кваліфікаційної роботи в ЕК	14.07.26	виконано

Дата видачі завдання 27.04.2026 р.

Здобувач _____ Руслан БУРЯК
(підпис) (власне ім'я, прізвище)

Керівник роботи _____ ст.викл. Дмитро ГУРІН
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 109 с., 2 табл., 12 рис., 3 дод., 16 джерел.

МОБІЛЬНИЙ РОБОТ, АЛГОРИТМ D* LITE, ПОБУДОВА МАРШРУТУ, ПЛАНУВАННЯ ТРАЄКТОРІЇ, НАВІГАЦІЯ МОБІЛЬНОГО РОБОТА, ДИНАМІЧНЕ СЕРЕДОВИЩЕ, ПЕРЕПЛАНУВАННЯ МАРШРУТУ, ОБХІД ДИНАМІЧНИХ ПЕРЕШКОД.

Метою кваліфікаційної роботи є розроблення програмного забезпечення для побудови оптимального маршруту мобільного робота в динамічному середовищі з використанням алгоритму D* Lite, що забезпечує ефективну адаптацію до змін навколишнього середовища в реальному часі.

Об’єктом розробки є процес навігації мобільного робота в умовах частково або повністю невідомого та динамічного середовища.

Предметом розробки є методи та алгоритми побудови маршруту мобільного робота, зокрема алгоритм D* Lite, а також програмні засоби його реалізації для забезпечення ефективного планування шляху.

У роботі проведено аналіз сучасних мобільних роботів та алгоритмів побудови маршруту, розроблено математичне та алгоритмічне забезпечення програми побудови маршруту мобільного робота. Реалізовано функції побудови маршруту та розроблено інтерфейс користувача. Розроблено інтерфейс виводів результатів розрахунків та проведено експериментальні дослідження та аналіз отриманих результатів.

Отримані результати можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме 9.5.

ABSTRACT

Explanatory note: 109 p., 2 tables, 12 figures, 3 appendices, 16 sources.

MOBILE ROBOT, D* LITE ALGORITHM, ROUTE CONSTRUCTION, TRAJECTORY PLANNING, MOBILE ROBOT NAVIGATION, DYNAMIC ENVIRONMENT, ROUTE RE-PLANNING, DYNAMIC OBSTACLES AVOIDANCE.

The purpose of the qualification work is to develop software for constructing the optimal route of a mobile robot in a dynamic environment using the D* Lite algorithm, which ensures effective adaptation to environmental changes in real time.

The object of development is the process of mobile robot navigation in a partially or completely unknown and dynamic environment.

The subject of development is methods and algorithms for building a mobile robot route, in particular the D* Lite algorithm, as well as software tools for its implementation to ensure effective path planning.

The paper analyzes modern mobile robots and route planning algorithms, develops mathematical and algorithmic support for the mobile robot route planning program. Route planning functions are implemented and a user interface is developed. An interface for outputting calculation results is developed, and experimental studies and analysis of the results are conducted.

The results obtained can be attributed to Sustainable Development Goal 9 "Industry, Innovation and Infrastructure", namely 9.5.

ЗМІСТ

Перелік скорочень 9
Вступ 10
1 Аналіз сучасних мобільних роботів та алгоритмів побудови маршруту 12
1.1 Аналіз сучасних мобільних роботів та їх призначення 12
1.2 Аналіз сучасних методів керування мобільними роботами 13
1.3 Аналіз алгоритмів побудови маршрутів мобільного та їх порівняння 15
1.4 Постановка задач 17
2 Розробка математичного та алгоритмічного забезпечення програми побудови маршруту мобільного робота 19
2.1 Розробка математичного забезпечення побудови маршруту мобільного робота на базі алгоритму D* lite 19
2.2 Розробка загального алгоритму роботи програми 30
2.3 Розробка алгоритму обходу статичних перешкод 33
2.4 Розробка алгоритму обходу динамічних перешкод 37
2.5 Розрахунок вагової довжини маршруту 40
3 Розробка програми побудови маршруту мобільного робота з використанням алгоритму D* lite 44
3.1 Обґрунтування вибору мови та середовища розробки 44
3.2 Реалізація функцій побудови маршруту 45
3.3 Розробка інтерфейсу користувача 48
3.4 Розробка інтерфейсу виводів результатів розрахунків 54
3.5 Експериментальні дослідження та аналіз отриманих результатів 58
3.6 Охорона праці 64
Висновки 67
Перелік джерел посилання 69
Додаток А Код програми 72
Додаток Б Апробація результатів кваліфікаційної роботи 99

Додаток В Демонстраційний матеріал 108

ПЕРЕЛІК СКОРОЧЕНЬ

ПІД-регулятор – пропорційно-інтегрально-диференціальний регулятор;

HMI – Human–Machine Interface (людино-машинний інтерфейс);

LED – Light-Emitting Diode (світлодіод).

ВСТУП

Сучасний етап розвитку робототехніки характеризується активним впровадженням мобільних роботів у різні сфери діяльності людини, зокрема в промисловість, логістику, медицину, сільське господарство та військову галузь. Однією з ключових задач функціонування таких систем є ефективна навігація в умовах змінного середовища, що передбачає побудову оптимального маршруту руху з урахуванням наявних та нових перешкод.

Особливої актуальності набувають алгоритми планування маршруту, здатні працювати в режимі реального часу та адаптуватися до динамічних змін середовища. Серед таких алгоритмів важливе місце займає D* Lite, який забезпечує ефективне інкрементальне перепланування шляху без необхідності повного перерахунку маршруту, що дозволяє значно підвищити швидкодію та надійність систем автономної навігації мобільних роботів.

Метою кваліфікаційної роботи є розроблення програмного забезпечення для побудови оптимального маршруту мобільного робота в динамічному середовищі з використанням алгоритму D* Lite, що забезпечує ефективну адаптацію до змін навколишнього середовища в реальному часі.

Об’єктом розробки є процес навігації мобільного робота в умовах частково або повністю невідомого та динамічного середовища.

Предметом розробки є методи та алгоритми побудови маршруту мобільного робота, зокрема алгоритм D* Lite, а також програмні засоби його реалізації для забезпечення ефективного планування шляху.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних мобільних роботів та визначити їх основні функціональні можливості і сфери застосування;

– провести аналіз сучасних методів керування мобільними роботами;

– провести аналіз існуючих алгоритмів побудови маршрутів та здійснити їх порівняння;

– обґрунтувати вибір алгоритму D* Lite для розв’язання задачі планування маршруту;

– розробити математичне забезпечення побудови маршруту мобільного робота;

– розробити загальний алгоритм роботи програми;

– розробити алгоритми обходу статичних та динамічних перешкод;

– виконати необхідні розрахунки для забезпечення коректної роботи алгоритму;

– обґрунтувати вибір мови програмування та середовища розробки;

– розробити програму побудови маршруту мобільного робота з використанням алгоритму D* Lite;

– розробити інтерфейс користувача та засоби виведення результатів;

– провести експериментальні дослідження та виконати аналіз отриманих результатів.

Кваліфікаційна робота виконана згідно ДСТУ 3008–15 [1], методичних вказівок [2], також, отримала апробацію на науковій конференції [3], а результати можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме 9.5.

1 АНАЛІЗ СУЧАСНИХ МОБІЛЬНИХ РОБОТІВ ТА АЛГОРИТМІВ ПОБУДОВИ МАРШРУТУ

1.1 Аналіз сучасних мобільних роботів та їх призначення

Сучасні мобільні роботи є складними системами, що здатні автономно або напівавтономно переміщатися в просторі та виконувати поставлені завдання без безпосередньої участі людини. Вони активно розвиваються завдяки прогресу в галузі сенсорики, штучного інтелекту, комп'ютерного зору та алгоритмів навігації. Основною особливістю таких систем є можливість взаємодії з навколишнім середовищем у режимі реального часу.

За конструктивними особливостями мобільні роботи можуть бути наземними, повітряними, підводними або гібридними. Найбільш поширеними є наземні мобільні роботи, які використовуються у складських системах, логістиці, виробництві та сервісних задачах. Вони можуть мати колісну, гусеничну або крокуючу кінематику, що визначає їх прохідність та сферу застосування [4].

Одним із ключових напрямків використання мобільних роботів є автоматизація логістичних процесів. У складських комплексах такі роботи виконують транспортування вантажів, сортування товарів та оптимізацію внутрішніх переміщень, що дозволяє значно зменшити витрати часу та людських ресурсів, підвищуючи ефективність роботи підприємств.

У промисловості мобільні роботи застосовуються для виконання небезпечних, монотонних або високоточних операцій. Вони можуть працювати в умовах підвищеної температури, забруднення або радіації, де присутність людини є небажаною або небезпечною, це значно підвищує рівень безпеки виробничих процесів [5].

Окрему категорію становлять сервісні мобільні роботи, які використовуються у побуті та сфері обслуговування. До них належать роботи-

пилососи, роботи для доставки товарів, а також медичні роботи, що можуть транспортувати медикаменти або допомагати в лікарнях. Їх головною метою є підвищення комфорту та автоматизація рутинних задач.

У військовій та рятувальній сферах мобільні роботи застосовуються для розвідки, пошуково-рятувальних операцій та розмінування територій. Вони дозволяють виконувати завдання в умовах підвищеного ризику для життя людини, забезпечуючи дистанційний контроль та високу точність виконання операцій.

Сучасні мобільні роботи є універсальними системами, що знаходять застосування в різних галузях діяльності. Їх розвиток безпосередньо пов’язаний із вдосконаленням алгоритмів навігації, систем керування та методів обробки даних, що дозволяє розширювати їх функціональні можливості та підвищувати автономність роботи [6].

1.2 Аналіз сучасних методів керування мобільними роботами

Сучасні методи керування мобільними роботами базуються на поєднанні класичних підходів теорії автоматичного керування та сучасних алгоритмів штучного інтелекту. Їх основною метою є забезпечення стабільного, точного та адаптивного руху робота в умовах невизначеного або динамічного середовища. Вибір методу керування значною мірою визначає ефективність виконання задач навігації, уникнення перешкод та досягнення цільових точок.

Одним із базових підходів є класичне керування на основі математичних моделей, яке використовує ПІД-регулятори. Вони широко застосовуються для стабілізації руху та підтримання заданих параметрів швидкості й положення. Основною перевагою таких систем є простота реалізації та низькі обчислювальні витрати, однак їх ефективність знижується в умовах складного або невизначеного середовища [7].

Методи реактивного керування базуються на миттєвій обробці даних сенсорів і прийнятті рішень у реальному часі. Вони не потребують повної карти

середовища, що робить їх ефективними для уникнення раптових перешкод. Проте такі методи можуть призводити до неоптимальних траєкторій та зациклення в складних конфігураціях простору. Планувальні методи керування, зокрема ті, що використовують алгоритми A*, D* та D* Lite, забезпечують побудову оптимального або наближеного до оптимального маршруту на основі відомої карти середовища. Вони характеризуються високою якістю траєкторії, але потребують більшої обчислювальної потужності та оновлення плану при зміні умов [8].

Інтелектуальні методи керування, що базуються на нейронних мережах та нечіткій логіці, дозволяють роботам адаптуватися до невизначеності та навчатися на основі попереднього досвіду. Вони забезпечують високу гнучкість, але потребують значних обсягів даних для навчання та складні у реалізації. Гібридні системи керування поєднують переваги декількох підходів, забезпечуючи як глобальне планування маршруту, так і локальне уникнення перешкод. Такий підхід є найбільш практичним для сучасних автономних систем, оскільки дозволяє досягати балансу між точністю, швидкодією та адаптивністю [9]. Порівняльна характеристика методів керування мобільними роботами представлено в таблиці 1.1.

Таблиця 1.1 – Порівняльна методів керування мобільними роботами

Метод керування	Складність реалізації	Швидкодія	Якість траєкторії	Адаптивність	Вимоги до середовища
1	2	3	4	5	6
ПІД-регул.	Низька	Висока	Середня	Низька	Відомі параметри
Реактивне	Низька–середня	Дуже висока	Низька–середня	Висока	Не потребує карти
Планувал.	Середня–висока	Середня	Висока	Середня	Потрібна карта

Проодовження таблиці 1.1

1	2	3	4	5	6
Інтелект.	Висока	Середня	Висока	Дуже висока	Дані для навчання
Гібридне	Висока	Висока	Висока	Висока	Частково відома карта

Сучасні методи керування мобільними роботами формують багаторівневу систему підходів, кожен з яких має свої переваги та недоліки. На практиці найбільш ефективними є гібридні рішення, які поєднують глобальне планування та локальне реагування на зміни середовища [10].

1.3 Аналіз алгоритмів побудови маршрутів мобільного та їх порівняння

Побудова маршруту мобільного робота є однією з ключових задач автономної навігації, що полягає у визначенні оптимального шляху від початкової до цільової точки з урахуванням перешкод та обмежень середовища. Сучасні алгоритми планування маршрутів умовно поділяються на класичні графові, евристичні, інкрементальні та алгоритми на основі штучного інтелекту. Кожен із цих підходів має свої особливості, переваги та обмеження.

До класичних алгоритмів належить алгоритм Дейкстри, який забезпечує знаходження найкоротшого шляху у зваженому графі. Його головною перевагою є гарантована оптимальність результату, однак він має високу обчислювальну складність, що робить його менш ефективним для великих просторів. Покращенням є алгоритм A*, який використовує евристичну функцію для прискорення пошуку та значно зменшує кількість перевірених вузлів [11].

Алгоритм A* є одним із найпоширеніших у задачах навігації роботів, оскільки поєднує оптимальність і відносно високу швидкодію. Однак його недоліком є необхідність повторного повного перерахунку маршруту у разі змін у середовищі, що знижує ефективність у динамічних умовах.

Для роботи в динамічних середовищах використовуються інкрементальні алгоритми, такі як D* та D* Lite. Вони дозволяють оновлювати вже знайдений маршрут без повного перерахунку, що суттєво підвищує ефективність роботи системи. Зокрема, алгоритм D* Lite є спрощеною та більш ефективною реалізацією D*, яка забезпечує швидке перепланування шляху при появі нових перешкод [12].

Також застосовуються алгоритми на основі потенціальних полів, де цільова точка розглядається як «притягальна сила», а перешкоди – як «відштовхувальні». Такі методи є простими у реалізації та добре підходять для локального планування, але можуть призводити до зациклення робота або попадання у локальні мінімуми.

Окрему групу становлять ймовірнісні алгоритми, такі як RRT (Rapidly-exploring Random Tree) та його модифікації. Вони ефективні для роботи у складних просторах високої розмірності та дозволяють знаходити рішення навіть у випадках, коли класичні методи малоефективні. Проте отримані траєкторії часто потребують додаткового згладжування [13]. Наведемо порівняльну характеристику основних алгоритмів побудови маршрутів, представлену в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика основних алгоритмів побудови маршрутів

Алгоритм	Оптим. маршруту	Швидкод.	Робота в динам. середов.	Обчисл. складн.	Особливості
1	2	3	4	5	6
Дейкстри	Висока (гарантована)	Низька	Ні	Висока	Повний перебір графа

Продовження таблиці 1.2

1	2	3	4	5	6
A*	Висока	Висока	Обмежена	Середня	Використання евристики
D* Lite	Висока	Висока	Так	Середня	Інкрементальн. перепланув.
Потенц. поля	Середня	Дуже висока	Частково	Низька	Ризик локальних мінімумів
RRT	Низька–середня	Середня	Так	Середня–висока	Стохастичний підхід

Проведений аналіз показує, що класичні алгоритми ефективні для статичних середовищ, тоді як інкрементальні та стохастичні підходи краще підходять для динамічних умов. З урахуванням вимог до роботи мобільного робота в змінному середовищі, доцільним є використання алгоритму D* Lite як оптимального компромісу між швидкодією, адаптивністю та якістю маршруту [14].

1.4 Постановка задач

Метою даної кваліфікаційної роботи є розроблення програмного забезпечення для побудови маршруту мобільного робота в умовах статичного та динамічного середовища з використанням алгоритму D* Lite. Система повинна забезпечувати можливість автоматичного планування траєкторії руху, а також оперативного її оновлення у випадку появи нових перешкод або зміни конфігурації середовища.

Об’єктом дослідження є процес навігації мобільного робота у дискретизованому просторі з наявністю перешкод, а предметом – методи та

алгоритми побудови та корекції маршруту, зокрема алгоритм D* Lite, який дозволяє виконувати інкрементальне перепланування шляху.

Для досягнення поставленої мети необхідно розробити програмну систему, яка буде моделювати середовище руху мобільного робота у вигляді сіткової або графової структури, де кожна клітина або вузол представляє можливий стан простору. Система повинна враховувати як статичні, так і динамічні перешкоди, що впливають на побудову маршруту.

Основною вимогою до алгоритму є забезпечення знаходження оптимального або наближеного до оптимального шляху від початкової до цільової точки з можливістю швидкого оновлення маршруту при зміні середовища. Також важливо забезпечити ефективність обчислень, що дозволить використовувати систему в режимі реального часу.

Програмна реалізація повинна включати модуль формування карти середовища, модуль побудови маршруту на основі алгоритму D* Lite, а також модуль обробки змін у середовищі та повторного розрахунку траєкторії. Окремо необхідно реалізувати інтерфейс користувача для задання початкових умов, візуалізації процесу пошуку шляху та відображення результатів роботи алгоритму.

Ключова задача полягає у створенні комплексної програмної системи, що дозволяє моделювати роботу мобільного робота в динамічному середовищі, виконувати побудову оптимального маршруту та забезпечувати його адаптивне оновлення з використанням алгоритму D* Lite.

2 РОЗРОБКА МАТЕМАТИЧНОГО ТА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ПРОГРАМИ ПОБУДОВИ МАРШРУТУ МОБІЛЬНОГО РОБОТА

2.1 Розробка математичного забезпечення побудови маршруту мобільного робота на базі алгоритму D* lite

Розроблення математичного забезпечення побудови маршруту мобільного робота на базі алгоритму D* Lite є необхідним для формалізації процесів сприйняття середовища, оцінювання допустимих переходів між станами та оперативного перепланування траєкторії в умовах появи статичних і динамічних перешкод. Саме математичний опис дозволяє представити логіку роботи алгоритму у вигляді строго визначених функцій, критеріїв і правил оновлення, що забезпечує однозначність програмної реалізації та можливість подальшого аналізу ефективності маршрутизації. Крім того, наявність таких моделей створює основу для дослідження часових, просторових і обчислювальних характеристик процесу навігації мобільного робота в дискретному робочому середовищі.

Для формального подання робочого середовища мобільного робота у вигляді двовимірної сітки, у вузлах якої виконується пошук маршруту розробимо модель дискретного представлення карти середовища, що дозволить використовувати в програмі для генерації карти, розміщення старту, цілі, статичних і динамічних перешкод, представлену формулою (2.1):

$$\begin{gathered} M = \{(x,y) | x \in \{0,1,\dots,W-1\}, y \in \{0,1,\dots,H-1\}\} \\ s_{start} = (x_s, y_s),\ s_{goal} = (x_g, y_g), \\ \mathcal{O}_{start} \subset \mathcal{M},\ \mathcal{O}_{dyn}(k) \subset \mathcal{M}, \\ \mathcal{O}(k) = \mathcal{O}_{start} \cup \mathcal{O}_{dyn}(k), \end{gathered} \tag{2.1}$$

де M – множина всіх клітин карти;

W – ширина карти в клітинах;

H – висота карти в клітинах;

s_{start} – стартова клітина робота;

s_{goal} – цільова клітина;

$\mathcal{O}_{start}$ – множина статичних перешкод;

$\mathcal{O}_{dyn}(k)$ – множина динамічних перешкод на кроці k;

$\mathcal{O}(k)$ – повна множина недоступних клітин на кроці k.

Для переходу від сіткової карти до графової структури, на якій безпосередньо працює алгоритм D* Lite розробимо модель графа станів, що дає можливість представити кожну клітину карти розглядається як вершина графа, а допустимі переходи між сусідніми клітинами – як ребра, представимо у вигляді формули (2.2):

$$\begin{gathered} G = (V, E), \\ V = \mathcal{M}, \\ E = \{(u, v) | u, v \in V, v \in Nbr(u)\}, \\ Nrb(x, y) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}, \end{gathered} \tag{2.2}$$

де G – граф робочого простору;

V – множина вершин графа;

E – множина ребер графа;

u, v – поточна та сусідня вершини;

$Nbr(u)$ – множина сусідів вершини u за 4-зв’язною схемою.

Визначення допустимості руху між сусідніми клітинами та для врахування перешкод при плануванні маршруту необхідно врахувати за допомогою моделі вартості переходу між станами, де вартість переходу дорівнює одиниці для вільних клітин і нескінченності для переходів через перешкоди, представлено у вигляді формули (2.3):

$$c_k(u,v) = \begin{cases} 1, u \notin \mathcal{O}(k), v \notin \mathcal{O}(k), v \notin Nbr(u) \\ \infty, \text{інакше} \end{cases} \quad (2.3)$$

де $c_k(u,v)$ – вартість переходу з вершини u у вершину v на кроці k;

$\mathcal{O}(k)$ – множина зайнятих клітин;

∞ – недопустимий перехід;

1 – базова вартість переходу між сусідніми вільними клітинами.

Евристична функція наближення до цілі дає можливість провести оцінювання відстані між поточним станом і ціллю та для формування пріоритету вершин у черзі алгоритму, що дає можливість реалізувати у програмі мангеттенську відстань, що є природною для чотирьохнапрямленого руху по сітці та представлено у вигляді формули (2.4):

$$h(u,v) = |x_u - x_v| + |y_u - y_v|, \quad (2.4)$$

де $h(u,v)$ – евристична оцінка відстані між вершинами u та v;

x_u, y_u – координати вершини u;

x_v, y_v – координати вершини v;

Для опису двох основних оцінок D* Lite, на основі яких визначається узгодженість вершини та обчислюється оптимальний маршрут представим функція $g(s)$ та $rhs(s)$. Значення $g(s)$ описує поточну відому вартість шляху, а $rhs(s)$ – одно-крокову прогнозовану вартість досягнення цілі через найкращого сусіда, як представлено у вигляді формули (2.5):

$$g(s) \in [0, \infty],$$
$$rhs(s) = \begin{cases} 0, s = s_{goal} \\ \min\limits_{\acute{s} \in Succ(s)} (c(s,\acute{s}) + g(\acute{s})), s \neq s_{goal}, \end{cases} \quad (2.5)$$

де $g(s)$ – поточна оцінка вартості шляху від вершини s до цілі;

$rhs(s)$ – однокрокова узгоджена оцінка для вершини s;

s_{goal} – множина наступників вершини s;

$c(s, \acute{s})$ – вартість переходу.

Умова узгодженості вершини необхідна для визначення, чи потребує вершина оновлення в процесі перепланування, якщо оцінка $g(s)$ та $rhs(s)$ різні, то вершина є неузгодженою і повинна бути оброблена алгоритмом, як представлено у вигляді формули (2.6):

$$\begin{gathered} g(s) = rhs(s) \Rightarrow s \text{ узгоджена}, \\ g(s) \neq rhs(s) \Rightarrow s \text{ неузгоджена}, \end{gathered} \tag{2.6}$$

де $g(s)$ – поточна оцінка вартості;

$rhs(s)$ – локально узгоджена оцінка;

s – довільна вершина графа.

Модель ключа вершини у черзі пріоритетів, потрібна для впорядкування вершин у відкритому списку OPEN, щоб алгоритм спочатку обробляв найважливіші вершини. У D* Lite кожна вершина має лексикографічний ключ із двох компонент, який враховує поточну вартість, евристику та накопичений зсув при русі робота, як представлено у вигляді формули (2.7):

$$\begin{gathered} Key(s) = (k_1(s), k_2(s)), \\ k_1(s) = \min\left(g(s), rhs(s) + h\left(s_{start,} s\right) + k_m, \right. \\ k_2(s) = \min\left(g(s), rhs(s)\right), \end{gathered} \tag{2.7}$$

де $Key(s)$ – ключ вершини s у черзі пріоритетів;

$k_1(s)$ – перша компонента ключа;

$k_2(s)$ – друга компонента ключа;

$h\left(s_{start,} s\right)$ – евристична відстань від поточного старту до вершини s;

k_m – накопичена поправка ключа при зміні положення робота;

$g(s), rhs(s)$ – оціночні функції алгоритму.

Опишемо початкові умови алгоритму, ця модель потрібна для ініціалізації алгоритму перед початком планування. У програмі це вирішує задачу для всіх вершин, крім цілі, початково встановлюється нескінченна вартість, а для цілі значення rhs дорівнює нулю:

$$\begin{gathered} g(s) = \infty, \quad \forall s \in V, \\ rhs(s) = \infty, \ \forall s \in V, s \neq s_{goal}, \\ rsh(s_{goal}) = 0, \\ OPEN = \{s_{goal}\}, \end{gathered} \tag{2.8}$$

де $g(s)$ – початкова оцінка вартості;

$rhs(s)$ – початкова локальна оцінка;

V – множина всіх вершин;

$OPEN$ – черга пріоритетів;

s_{goal} – цільова вершина.

Для обновлення вершин графа, розробимо модель яка дає можливість локального перерахунку оцінок після зміни карти або вартості ребер, ця операція застосовується до змінених клітин і до їх сусідів, коли динамічні перешкоди переміщуються та обчислюється за формулою (2.9):

$$\begin{gathered} rhs(u) = \begin{cases} 0, u = s_{goal} \\ \min\limits_{ś \in Succ(u)} \left(c(u, ś) + g(ś)\right), u \neq s_{goal}, \end{cases} \\ g(u) \neq rhs(u) \Longrightarrow u \in Open, \end{gathered} \tag{2.9}$$

де u – вершина, що оновлюється;

$Succ(u)$ – множина сусідніх вершин, у які можна перейти;

$c(u, ś)$ – вартість переходу;

$g(u)$ – поточна оцінка;

$rhs(u)$ – оновлена локальна оцінка;

$Open$ – черга вершин, що підлягають обробці.

Для визначення, чи завершено процес пошуку або перепланування маршруту необхідно розробити та математично описати процедуру умов завершення, тобто алгоритм D* lite продовжує роботу, поки ключ найкращої вершини у черзі менший за ключ стартової вершини або поки стартова вершина не стала узгодженою, що можна описати як формула (2.10):

$$while(TopKey(Open) < Key(s_{start})) \vee (rhs(s_{start}) \neq g(s_{start})), \qquad (2.10)$$

де $TopKey(Open)$ – найменший ключ у черзі пріоритетів;

$Key(s_{start})$ – ключ поточної стартової вершини;

$rhs(s_{start})$ – локальна оцінка старту;

$g(s_{start})$ – поточна оцінка старту.

Ведемо правіло обновлення значення $g(u)$, ця модель потрібна для корекції вартості вершини в залежності від того, чи є вона завищеною чи заниженою відносно $rhs(u)$. У концепцій алгоритму D* lite, це дозволяє коректно перебудовувати маршрут після змін у середовищі та може бути представлено як (2.11):

$$\begin{gathered} \text{якщо } g(u) > rhs(u), \text{то } g(u) \leftarrow rhs(u), \\ \text{інакше } g(u) \leftarrow \infty, \end{gathered} \qquad (2.11)$$

де u – поточна вершина, що виймається з черги;

$g(u)$ – поточна оцінка вартості;

$rhs(u)$ – локально узгоджена оцінка;

∞ – ознака того, що стара оцінка скидається для подальшого перерахунку.

Для поширення змін вартості на сусідні вершини після перерахунку поточної вершини необхідно розробити оновлення множини попередників, це реалізується у програми за рахунок зміни клітини або її стану оновлюються сама клітина та всі її сусіди та можна описати як (2.12):

$$\begin{gathered} Pred(u) = Nbr(u), \\ \forall p \in Pred(u): UpdateVertex(p), \end{gathered} \tag{2.12}$$

де $Pred(u)$ – множина попередників вершини u;

$Nbr(u)$ – множина сусідів вершини u;

p – окрема сусідня вершина;

$UpdateVertex(p)$ – операція локального оновлення вершини.

Ведемо модель накопичення поправки k_m, ця модель потрібна для врахування зміни поточного положення робота без повного перерахунку евристики для всіх вершин та завдяки цій поправці алгоритм D* Lite ефективно виконує перепланування, коли робот рухається по карті, як описано в формулі (2.13):

$$\begin{gathered} k_m \leftarrow k_m + h(s_{last}, s_{start}^{new}), \\ s_{last} \leftarrow s_{start}^{new}, \end{gathered} \tag{2.13}$$

де k_m – накопичена поправка ключа;

s_{last} – попередня позиція робота;

s_{start}^{new} – нова поточна позиція робота після руху;

$h(\cdot)$ – евристична функція відстані.

Для фактичного переміщення робота після завершення чергового етапу планування, розробимо модель вибору наступного кроку, тобто реалізуємо можливість роботу переходить у того сусіда, для якого сума вартості переходу та оцінки g є мінімальною, як показано в (2.14):

$$\begin{gathered} s_{next} = arg \min_{s \in Succ(s_{robor})} (c(s_{robot}, \acute{s}) + g(\acute{s})), \\ s_{robot}(k+1) = s_{next}, \end{gathered} \tag{2.14}$$

де s_{next} – наступна клітина маршруту;

s_{robot} – поточна позиція робота;

$s_{robot}(k+1)$ – позиція робота на наступному кроці;

$Succ(s_{robor})$ – множина сусідів поточного положення;

$c(s_{robot}, \acute{s})$ – вартість переходу;

$g(\acute{s})$ – оцінка вартості до цілі через сусідню вершину.

Модель зміни динамічних перешкод потрібна для опису часової зміни конфігурації середовища, яка змушує алгоритм перепланувати маршрут у реальному часі, та може бути представлена у спрощеній реалізації динамічна перешкода змінює своє положення на сусідню клітину відповідно до свого напрямку руху або відбивається від меж і зайнятих клітин, як представлено формулою (2.15):

$$\begin{gathered} o_i(k) = (x_i(k), y_i(k)), \\ o_i(k+1) = o_i(k) + (\Delta x_i(k), \Delta y_i(k)), \\ \mathcal{O}_{dyn}(k) = \{o_i(k)\}_{i=1}^{N_{dyn}}, \end{gathered} \tag{2.15}$$

де $o_i(k)$ – положення i-ї динамічної перешкоди на кроці k;

$x_i(k), y_i(k)$ – координати перешкоди;

$\Delta x_i(k), \Delta y_i(k)$ – зміщення по координатах за один такт;

N_{dyn} – кількість динамічних перешкод;

$\mathcal{O}_{dyn}(k)$ – множина всіх динамічних перешкод на кроці k.

Для визначення тих клітин, для яких слід виконати перепланування після переміщення динамічних перешкод, пропонується використовувати симетричну різницю між попередньою та новою множинами положень динамічних перешкод, як представлено у формулі (2.16):

$$\Delta\mathcal{O}(k) = \mathcal{O}_{dyn}(k-1)\Delta\mathcal{O}_{dyn}(k), \tag{2.16}$$

де $\Delta\mathcal{O}(k)$ – множина клітин, стан яких змінився на кроці k;

$\mathcal{O}_{dyn}(k-1)$ – положення динамічних перешкод на попередньому кроці;

$\mathcal{O}_{dyn}(k)$ – положення на поточному кроці.

Модель довжини побудованого маршруту потрібна для кількісного оцінювання маршруту, який згенерував алгоритм та для подальшого порівняння з іншими експериментами показано в (2.17):

$$L_{path} = \sum_{i=0}^{N_{p-2}} d(s_i, s_{i+1}). \tag{2.17}$$

Для 4-зв'язної сітки:

$$d(s_i, s_{i+1}) = 1, \tag{2.18}$$

тоді:

$$L_{path} = N_p - 1, \tag{2.19}$$

де L_{path} – довжина маршруту;

N_p – кількість вершин у маршруті;

s_i – i-та вершина маршруту;

$d(s_i, s_{i+1})$ – локальна відстань між сусідніми вершинами.

Відповідно для оцінювання обчислювальної ефективності алгоритму D* Lite в процесі моделювання розробимо модель часу планування. У програмі що буде розроблятися пропонується час планування вимірюється для кожного циклу, як показано у формулі (2.20)

$$\begin{aligned} T_{plan}(k) &= t_{end}(k) - t_{start}(k), \\ \bar{T}_{plan} &= \frac{1}{N}\sum_{k=1}^{N} T_{plan}(k), \\ T_{plan}^{max} &= \max_{1 \le k \le N} T_{plan}(k), \end{aligned} \tag{2.20}$$

де $T_{plan}(k)$ – час планування на кроці k;

$t_{start}(k)$ – момент початку планування;

$t_{end}(k)$ – момент завершення;

$\bar{T}_{plan}$ – середній час планування;

T_{plan}^{max} – максимальний час планування;

N – кількість циклів планування.

Для розрахунку відстані до цілі, що потрібна для аналізу процесу зближення робота з цільовою точкою в ході моделювання, пропонується наступна модель (2.21):

$$D_{goal}(k) = \left|x_r(k) - x_g\right| + |y_r(k) - y_g|, \tag{2.21}$$

де $D_{goal}(k)$ – відстань від робота до цілі на кроці k;

$x_r(k), y_r(k)$ – координати робота на кроці k;

x_g, y_g – координати цілі.

Модель критерію успішності досягнення цілі потрібний для формальної перевірки завершення моделювання та фіксації результату експерименту, у програмі будемо важать успішним, якщо поточна позиція робота збігається з ціллю, як показано у (2.22):

$$J_{goal} = \begin{cases} 1, s_{robot} = s_{goal} \\ 0, s_{robot} \neq s_{goal} \end{cases}, \tag{2.22}$$

де J_{goal} – бінарний критерій досягнення цілі;

s_{robot} – поточна позиція робота;

s_{goal} – цільова позиція.

Для оцінювання адаптивності алгоритму до змін динамічного середовища розробимо модель кількості перепланувань, цей показник збільшується щоразу, коли зміна перешкод вимагає повторного оновлення маршруту, як представлено в (2.23):

$$N_{replan} = \sum_{k=1}^{N} \delta_k,$$
$$\delta_k = \begin{cases} 1, \Delta\mathcal{O}(k) \neq \emptyset \\ 0, \Delta\mathcal{O}(k) = \emptyset' \end{cases} \quad (2.23)$$

де N_{replan} – загальна кількість перепланувань;

δ_k – індикатор перепланування на кроці k;

$\Delta\mathcal{O}(k)$ – множина змінених клітин на кроці k;

N – загальна кількість кроків моделювання.

Ведемо інтегральну модель функціонування алгоритму D* lite в програмі, це потрібно для цілісного опису циклу роботи D* Lite у створеній HMI-системі. Вона показує, що на кожному кроці відбуваються оновлення середовища, корекція карти перешкод, перепланування, вибір наступного стану й накопичення метрик, як показано в (2.24):

$$S(k) = \{\mathcal{M}, \mathcal{O}(k), s_{robot}(k), s_{goal}, g_k(\cdot), rhs_k(\cdot), Open_k\},$$
$$S(k+1) = \Phi(S(k), \Delta\mathcal{O}(k)), \quad (2.24)$$

де оператор Ф включає:

$$\Phi: \text{оновлення перешод} \rightarrow UpdateVertex \rightarrow$$
$$ComputeShortestPath \rightarrow Mave, \quad (2.25)$$

де $S(k)$ – стан системи на кроці k;

$\mathcal{O}(k)$ – карта перешкод;

$s_{robot}(k)$ – положення робота;

$g_k(\cdot)$ – множина значень функції g на кроці k;

$rhs_k(\cdot)$ – множина значень функції rhs;

$Open_k$ – стан черги пріоритетів;

Ф – оператор переходу системи до наступного стану.

У результаті розроблено математичне забезпечення алгоритму D* Lite, яке описує всі основні етапи побудови та перепланування маршруту мобільного робота в динамічному середовищі. Запропоновані математичні моделі охоплюють дискретне подання середовища, граф станів, оцінювання вартості переходів, евристичну функцію, механізми оновлення вершин, перепланування маршруту, моделювання динамічних перешкод, а також показники оцінювання ефективності роботи алгоритму. Розроблене математичне забезпечення стало основою для створення алгоритмічного та програмного забезпечення, забезпечивши коректну, адаптивну та ефективну реалізацію навігації мобільного робота в умовах зміни конфігурації середовища.

2.2 Розробка загального алгоритму роботи програми

Розробка загального алгоритму роботи програми є необхідною для формалізації послідовності виконання обчислювальних і керуючих процедур, що забезпечують коректне функціонування системи побудови маршруту мобільного робота. Наявність чітко визначеного алгоритму дозволяє узгодити взаємодію між модулями введення даних, генерації середовища, реалізації алгоритму D* Lite та візуалізації результатів у режимі реального часу. Крім того, алгоритмічний опис забезпечує можливість аналізу логіки роботи програми, оптимізації її продуктивності та виявлення потенційних помилок на етапі проєктування, це особливо важливо в умовах динамічного середовища, де необхідне оперативне перепланування маршруту та адаптація до змін. Загальний вигляд алгоритму роботи програми представлено на рисунку 2.1. Відповідно розробленому загальному алгоритму програми побудови маршруту мобільного робота на базі D* Lite опишемо призначення кожного його етапу:

– "введення вихідних даних" – на цьому етапі здійснюється введення користувачем основних параметрів моделювання, таких як розміри карти, координати старту та цілі, а також кількість статичних і динамічних перешкод. Ці дані визначають структуру середовища та початкові умови задачі

маршрутизації. Введені параметри перевіряються на коректність і допустимість для подальшої обробки. Отримані значення передаються до наступного етапу ініціалізації;

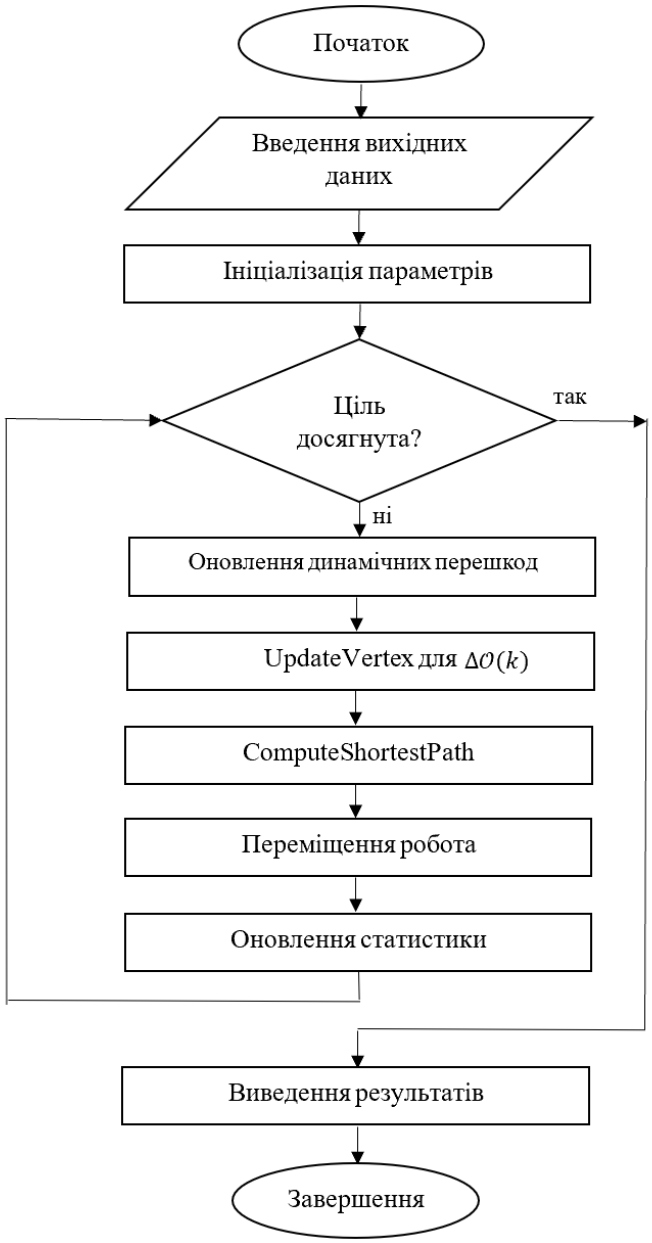


Рисунок 2.1 – Загальний алгоритм програми побудови маршруту мобільного робота на базі D* Lite

– "ініціалізація параметрів" – на цьому етапі формується внутрішнє представлення середовища у вигляді сітки або графа, а також ініціалізуються всі необхідні змінні алгоритму D* Lite. Задаються початкові значення функцій $g(s)$ та $rhs(s)$, визначається стартова та цільова вершини. Ініціалізується черга

пріоритетів OPEN та встановлюється початковий ключ для цільової вершини. Таким чином створюється база для подальшого виконання алгоритму;

– "ціль досягнута?" – даний блок виконує перевірку поточного стану системи на досягнення роботом цільової позиції. Порівнюються координати робота та цілі, що дозволяє визначити завершення процесу маршрутизації. У випадку позитивного результату алгоритм переходить до завершальних процедур. Якщо ціль не досягнута, процес продовжується з урахуванням змін у середовищі;

– "оновлення динамічних перешкод" – на цьому етапі виконується моделювання руху динамічних перешкод у середовищі. Координати перешкод змінюються відповідно до заданих правил руху або випадкових процесів. Формується нова конфігурація карти перешкод, яка може впливати на допустимість поточного маршруту. Результатом є оновлена множина зайнятих клітин середовища;

– "UpdateVertex для $\Delta\mathcal{O}(k)$ " – даний блок відповідає за локальне оновлення вершин графа, для яких змінився стан через переміщення перешкод. Виконується перерахунок значень $rhs(s)$ для змінених клітин та їх сусідів. У випадку неузгодженості значень $g(s)$ і $rhs(s)$ вершини додаються до черги пріоритетів. Це дозволяє алгоритму ефективно реагувати на локальні зміни без повного перепланування;

– "ComputeShortestPath" – на цьому етапі реалізується основний цикл алгоритму D* Lite для обчислення або оновлення найкоротшого маршруту. Виконується обробка вершин з черги OPEN відповідно до їх ключів та корекція значень $g(s)$ і $rhs(s)$. Алгоритм працює до досягнення узгодженості стартової вершини або поки не буде знайдено оптимальний шлях. Результатом є оновлений маршрут від поточної позиції робота до цілі;

– "переміщення робота" – на цьому етапі здійснюється вибір наступної вершини маршруту та фактичне переміщення робота у відповідну клітину. Вибір виконується на основі мінімізації вартості переходу з урахуванням значень $g(s)$.

Оновлюється поточна позиція робота та накопичується пройдений шлях. Це забезпечує поступове просування робота до цілі;

– "оновлення статистики" – даний блок відповідає за збір і оновлення метрик роботи алгоритму в процесі моделювання. Фіксуються такі параметри, як час планування, довжина шляху, відстань до цілі та кількість перепланувань. Ці дані використовуються для подальшого аналізу ефективності алгоритму. Накопичена статистика передається у модуль візуалізації результатів;

– "виведення результатів" – на цьому етапі здійснюється представлення отриманих результатів користувачу у графічній та числовій формі. Виводяться побудований маршрут, траєкторія руху робота та графіки ключових параметрів. Аналізуються отримані метрики для оцінки якості маршрутизації, що дозволяє зробити висновки щодо ефективності роботи алгоритму D* Lite в заданих умовах.

Розроблений алгоритм забезпечує ефективну побудову маршруту мобільного робота в умовах динамічного середовища завдяки інкрементальному механізму перепланування, що дозволяє уникнути повного перерахунку шляху при зміні перешкод. Він характеризується високою обчислювальною ефективністю та швидкою адаптацією до змін у середовищі, що є критично важливим для роботи в режимі реального часу. Алгоритм інтегрує як моделі середовища, так і механізми збору статистики, що дозволяє проводити глибокий аналіз якості маршрутизації. Крім того, його структура є універсальною та може бути розширена для складніших задач, включаючи багатороботні системи та інтеграцію з інтелектуальними методами керування.

2.3 Розробка алгоритму обходу статичних перешкод

Необхідність розробки алгоритму обходу статичних перешкод обумовлена тим, що реальні робочі середовища мобільних роботів майже завжди містять об’єкти, які обмежують або повністю блокують можливість прямолінійного руху до цілі. Без формалізованого підходу до врахування таких перешкод неможливо

гарантувати побудову коректного та безпечного маршруту. Алгоритм обходу дозволяє забезпечити визначення допустимих траєкторій, які не перетинають заборонені області простору, що є критично важливим для запобігання зіткненням. Крім того, він забезпечує оптимізацію довжини шляху з урахуванням геометрії перешкод та обмежень руху робота. У контексті дискретного подання середовища така задача потребує точного визначення правил переходу між станами та оцінювання їх вартості. Наявність ефективного алгоритму обходу також створює основу для подальшого розширення системи до більш складних сценаріїв, включаючи динамічні середовища. Алгоритм обходу статичних перешкод представлено на рисунку 2.2.

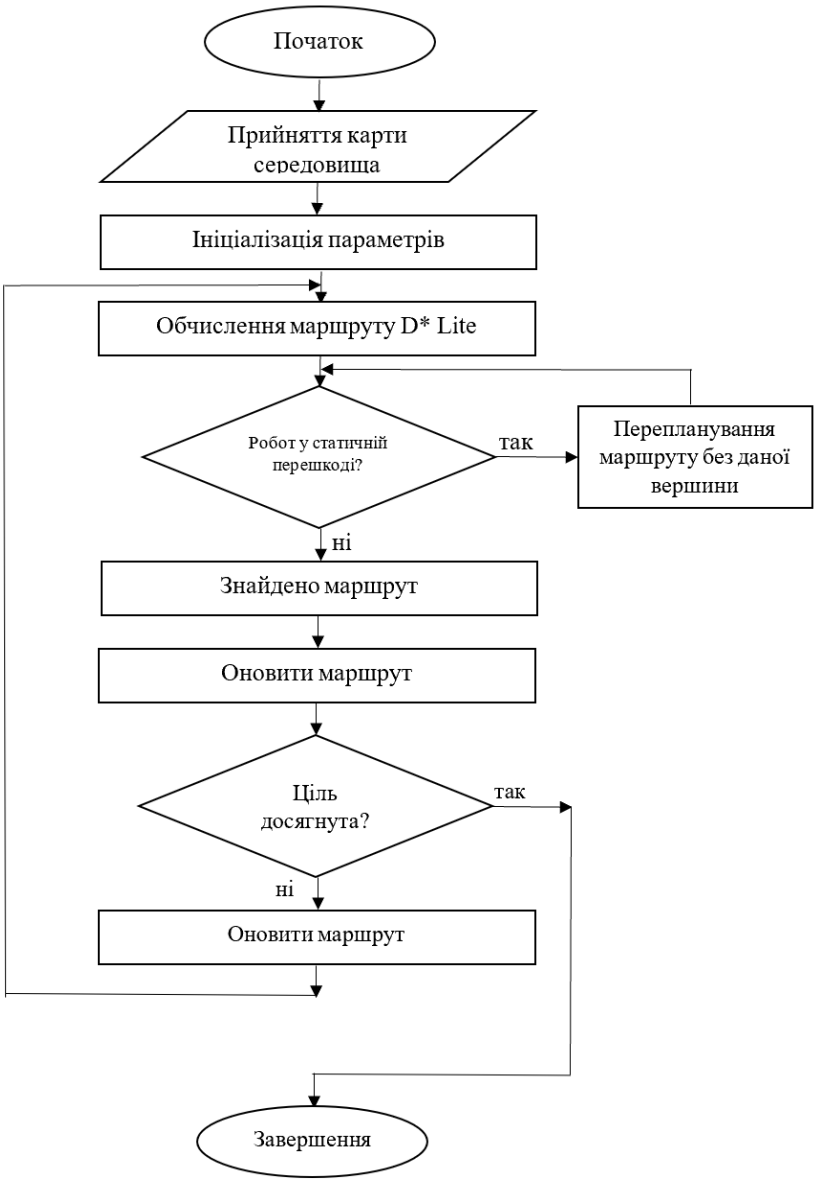


Рисунок 2.2 – Алгоритм обходу статичних перешкод

Опис призначення блоків алгоритму обходу статичних перешкод (рис. 2.2) представлено нижче:

– "прийняття карти середовища" – на цьому етапі відбувається отримання та формування дискретної моделі робочого простору, яка містить інформацію про вільні клітини та розташування статичних перешкод. Карта задає геометрію середовища, в якому буде виконуватись планування маршруту. Вона може бути згенерована або введена користувачем у рамках HMI-системи. Отримані дані використовуються як база для побудови графа станів;

– "ініціалізація параметрів" – цей блок відповідає за підготовку внутрішніх змінних алгоритму D* Lite та початкових умов задачі. Встановлюються координати старту і цілі, ініціалізуються значення функцій $g(s)$ та $rhs(s)$, а також формується початковий стан черги пріоритетів. Виконується прив’язка карти до графової структури для подальших обчислень. Це забезпечує готовність алгоритму до побудови маршруту;

– "обчислення маршруту D* Lite" – на даному етапі виконується первинний запуск алгоритму для знаходження найкоротшого шляху від старту до цілі з урахуванням статичних перешкод. Алгоритм обробляє вершини графа, мінімізуючи функцію вартості переходів і використовуючи евристичну оцінку. В результаті формується оптимальний маршрут або визначається його відсутність. Отриманий шлях передається до наступного етапу перевірки;

– "робот у статичній перешкоді?" – цей логічний блок перевіряє, чи поточна позиція робота або обрана наступна вершина маршруту співпадає з клітиною, зайнятою статичною перешкодою. Така ситуація може виникнути при некоректній ініціалізації або зміні середовища. Якщо умова виконується, необхідно виконати перепланування маршруту з виключенням проблемної вершини. У протилежному випадку алгоритм продовжує виконання;

– "перепланування маршруту без даної вершини" – у цьому блоці виконується повторний запуск алгоритму D* Lite з урахуванням виключення недопустимої вершини або області. Змінюється множина доступних станів графа, що дозволяє знайти альтернативний маршрут. Виконується оновлення

значень функцій $g(s)$ і $rhs(s)$ для нової конфігурації середовища. Це забезпечує адаптацію маршруту до обмежень;

– "знайдено маршрут" – на цьому етапі підтверджується факт існування допустимого шляху від старту до цілі. Виконується перевірка результату роботи алгоритму на наявність послідовності вершин, що задовольняє умовам задачі. У разі позитивного результату маршрут вважається коректним і може бути використаний для руху робота. Далі відбувається його актуалізація;

– "оновити маршрут" – цей блок відповідає за оновлення поточного маршруту з урахуванням нових обчислень або змін у середовищі. Формується актуальна послідовність вершин, яка буде використовуватись для переміщення робота. Виконується підготовка даних для наступного кроку руху. Це дозволяє підтримувати відповідність маршруту поточному стану системи;

– "ціль досягнута?" – даний логічний блок перевіряє, чи співпадає поточна позиція робота з цільовою точкою. Якщо умова виконується, процес обходу перешкод і побудови маршруту завершується. У випадку негативного результату алгоритм переходить до наступної ітерації оновлення маршруту. Це забезпечує циклічний характер роботи до досягнення цілі;

– "оновити маршрут" – повторне виконання цього блоку пов’язане з необхідністю адаптації маршруту після кожного кроку руху робота. Алгоритм враховує поточне положення та оновлює траєкторію для подальшого просування. Це дозволяє уникати перешкод і підтримувати оптимальність шляху. Процес повторюється до моменту досягнення цілі або завершення алгоритму.

Розроблений алгоритм обходу статичних перешкод забезпечує надійне формування безпечного маршруту за рахунок коректного врахування геометрії середовища та обмежень руху мобільного робота. Його інтеграція з алгоритмом D* Lite дозволяє ефективно виконувати перепланування маршруту без значних обчислювальних витрат, що підвищує швидкодію системи. Алгоритм демонструє високу стійкість до складних конфігурацій перешкод і здатність знаходити альтернативні шляхи у разі блокування основного маршруту. Крім

того, він забезпечує основу для подальшого розширення до більш складних сценаріїв навігації та інтеграції з інтелектуальними методами керування.

2.4 Розробка алгоритму обходу динамічних перешкод

Потреба у створенні алгоритму обходу динамічних перешкод зумовлена тим, що середовище функціонування мобільних роботів постійно змінюється і містить об’єкти з непередбачуваною траєкторією руху. У таких умовах заздалегідь сформований маршрут не може залишатися актуальним протягом усього часу виконання задачі. Відсутність механізму адаптації до змін призводить до зниження безпеки руху та підвищує ризик виникнення зіткнень. Використання спеціалізованого алгоритму дозволяє оперативно враховувати зміну конфігурації середовища та коригувати траєкторію в реальному часі, що забезпечує безперервність руху робота навіть за наявності рухомих перешкод і складних умов навігації. Крім того, такий підхід створює передумови для впровадження більш складних методів прогнозування поведінки об’єктів у середовищі. Приклад розробленного алгоритму обходу динамічних перешкод приведено на рисунку 2.3.

Опис призначення блоків алгоритму обходу динамічних перешкод (рис. 2.3) представлено нижче:

– "прийняття карти середовища" – на цьому етапі формується або зчитується дискретна модель середовища, яка включає інформацію про вільні області та розташування перешкод. Карта визначає простір, у якому здійснюється навігація мобільного робота. Вона є основою для побудови графа станів та подальших обчислень алгоритму. Отримані дані використовуються для ініціалізації системи та планування маршруту;

– "ініціалізація параметрів" – цей блок відповідає за встановлення початкових умов функціонування алгоритму D* Lite. Ініціалізуються значення функцій g(s) та rhs(s), задаються координати старту і цілі, а також формується початковий стан черги пріоритетів. Визначаються основні змінні, необхідні для

обчислення маршруту. Це забезпечує коректний запуск алгоритму в заданому середовищі;

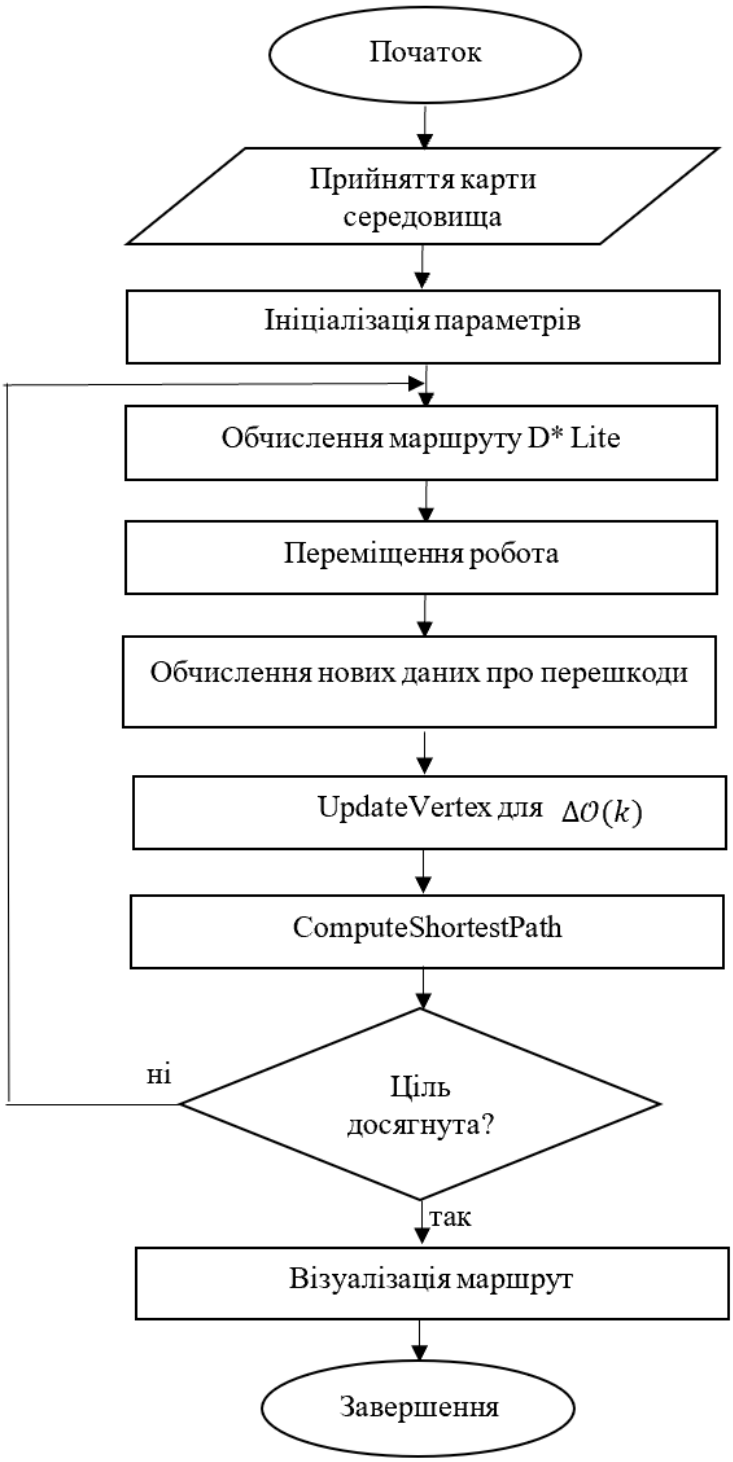


Рисунок 2.3 – Алгоритм обходу динамічних перешкод

– "обчислення маршруту D* Lite" – на цьому етапі виконується пошук початкового оптимального маршруту від стартової позиції до цільової з

урахуванням поточної конфігурації середовища. Алгоритм обробляє вершини графа, використовуючи евристичну функцію та оцінки вартості переходів. Визначається послідовність клітин, яка забезпечує мінімальну довжину шляху. Результатом є базовий маршрут для подальшого руху робота;

– "переміщення робота" – у цьому блоці здійснюється фактичне оновлення положення робота відповідно до знайденого маршруту. Вибирається наступна вершина шляху, яка має мінімальну вартість переходу. Координати робота змінюються, що дозволяє моделювати його рух у середовищі. Це забезпечує поступове наближення до цільової точки;

– "обчислення нових даних про перешкоди" – на цьому етапі відбувається оновлення інформації про динамічні перешкоди, які можуть змінювати своє положення у часі. Виконується аналіз нової конфігурації середовища та визначаються клітини, що стали зайнятими або звільненими. Це дозволяє враховувати зміну умов руху в реальному часі. Отримані дані використовуються для подальшого перепланування маршруту;

– "UpdateVertex для ΔO(k)" – цей блок реалізує локальне оновлення вершин графа, які були змінені внаслідок переміщення динамічних перешкод. Перераховуються значення rhs(s) для відповідних вершин і їх сусідів. У разі виникнення неузгодженості значень g(s) та rhs(s) вершини додаються до черги пріоритетів. Це забезпечує ефективне врахування змін без повного перерахунку маршруту;

– "ComputeShortestPath" – на цьому етапі виконується оновлення маршруту з урахуванням нових даних про середовище. Алгоритм повторно обробляє необхідні вершини та коригує значення функцій g(s) і rhs(s). Виконується інкрементальне перепланування, що дозволяє швидко адаптувати маршрут. Результатом є актуалізований шлях до цілі;

– "ціль досягнута?" – даний блок перевіряє, чи співпадає поточне положення робота з координатами цільової точки. У разі позитивного результату алгоритм переходить до завершального етапу візуалізації результатів. Якщо ціль

не досягнута, процес повторюється з урахуванням нових даних середовища. Це забезпечує циклічність роботи алгоритму;

– "візуалізація маршруту" – на цьому етапі здійснюється графічне представлення отриманого маршруту та траєкторії руху робота. Відображаються результати роботи алгоритму у зручному для користувача вигляді. Аналізуються ключові параметри маршруту, такі як довжина та форма траєкторії. Це дозволяє оцінити ефективність алгоритму в заданих умовах.

Розроблений алгоритм обходу динамічних перешкод забезпечує оперативну адаптацію маршруту мобільного робота до змін середовища за рахунок інкрементального перепланування, що дозволяє уникнути повного перерахунку шляху. Він підвищує безпеку руху, своєчасно враховуючи появу та переміщення перешкод, що мінімізує ризик зіткнень. Алгоритм демонструє високу обчислювальну ефективність і стабільність роботи в режимі реального часу навіть у складних умовах навігації. Крім того, його структура є гнучкою та придатною для інтеграції з методами прогнозування поведінки об’єктів і системами інтелектуального керування.

2.5 Розрахунок вагової довжини маршруту

Розрахунок вагової довжини маршруту є необхідним для більш точного врахування реальних умов руху мобільного робота, оскільки не всі ділянки середовища є однаково безпечними або енергетично вигідними. Такий підхід дозволяє інтегрувати додаткові фактори, зокрема близькість до перешкод, нерівномірність покриття або підвищені ризики, у процес прийняття рішень алгоритмом D* Lite. У результаті маршрут оптимізується не лише за геометричною довжиною, але й за узагальненим критерієм вартості, що підвищує надійність і безпеку навігації, що особливо важливо в складних або динамічних середовищах, де прямолінійний найкоротший шлях може бути небажаним або небезпечним.

Роздивимося задачу коли різні клітини або переходи мають різну вартість, наприклад через близькість до перешкод, небезпечні зони або енергетичні витрати. У такому випадку алгоритм D* Lite мінімізує не тільки геометричну довжину, а й узагальнену вартість маршруту, як показано у (2.26):

$$J = \sum_{i=0}^{N-2} c(s_i, s_{i+1}), \tag{2.26}$$

де:

$$\mathrm{c}(s_i, s_{i+1}) = w_i \cdot d(s_i, s_{i+1}), \tag{2.27}$$

де J – сумарна вагова довжина або вартість маршруту;

$c(s_i, s_{i+1})$ – вартість переходу між сусідніми вершинами;

w_i – ваговий коефіцієнт для переходу або зони руху;

$d(s_i, s_{i+1})$ – геометрична довжина окремого переходу.

Приведемо приклад розрахунку, нехай алгоритм D* Lite побудував маршрут на сітковій карті у вигляді послідовності клітин $s_0 = (0,0), s_1 = (1,0), s_2 = (2,0), s_3 = (2,1),\ s_4 = (2,2), s_5 = (3,2), s_6 = (4,2)$. У цьому випадку кількість вершин маршруту дорівнює $N = 7$, тоді для 4-зв'язної карти довжина маршруту в клітинах становить:

$$L = N - 1 = 7 - 1 = 6. \tag{2.28}$$

Перевіримо це через покрокову суму:

$$L = d(s_0, s_1) + d(s_1, s_2) + d(s_2, s_3) + d(s_3, s_4) + d(s_4, s_5) + d(s_5, s_6).$$

Оскільки всі переходи виконуються між сусідніми клітинами, то:

$$d(s_i, s_{i+1}) = 1,$$

тому:

$$L = 1 + 1 + 1 + 1 + 1 + 1 = 6.$$

Нехай розмір однієї клітини карти дорівнює:

$$a = 0{,}25 \text{ м}$$

Тоді фізична довжина маршруту становить:

$$L_m = L \cdot a = 6 \cdot 0.25 = 1.5\text{м}$$

Тепер нехай у процесі руху через появу динамічної перешкоди робот був змушений перепланувати маршрут і фактично пройшов 8 клітин замість 6, тоді:

$$L_{real} = 8.$$

Абсолютне збільшення довжини маршруту:

$$\Delta L = L_{real} - L_{plan} = 8 - 6 = 2.$$

Відносне збільшення довжини маршруту:

$$\delta_L = \frac{8 - 6}{6} \cdot 100\% = 33{,}3\ \%$$

Якщо додатково врахувати вагову вартість переходів, наприклад перші 4 кроки мають вагу $w = 1$, а останні 2 кроки проходять поблизу перешкоди з вагою $w = 1.5$, то сумарна вартість маршруту становитиме

$$J = 1 + 1 + 1 + 1 + 1{,}5 + 1{,}5 = 7.$$

Отже, для наведеного прикладу початкова довжина маршруту за алгоритмом D* Lite дорівнює 6 клітинам або 1,5 м, фактична довжина після перепланування може становити 8 клітин, а відносне збільшення маршруту в такому випадку дорівнює 33,3 %.

3 РОЗРОБКА ПРОГРАМИ ПОБУДОВИ МАРШРУТУ МОБІЛЬНОГО РОБОТА З ВИКОРИСТАННЯМ АЛГОРИТМУ D* LITE

3.1 Обґрунтування вибору мови та середовища розробки

Для розробки програми побудови маршруту мобільного робота з використанням алгоритму D* Lite доцільно обрати мову Python і середовище розробки PyCharm, оскільки саме така комбінація забезпечує оптимальне поєднання простоти реалізації, наочності коду та зручності відлагодження. У порівнянні з мовами C і C++, Python має значно простіший синтаксис, що дозволяє швидше реалізовувати складні алгоритми пошуку та перепланування маршруту без надмірного ускладнення програмного коду. Хоча C++ забезпечує вищу швидкодію, його використання потребує більших витрат часу на розробку, керування пам’яттю та супровід програмного забезпечення. У порівнянні з Java, Python є більш гнучким для науково-дослідних задач, математичного моделювання та швидкого створення прототипів, що є важливим при дослідженні алгоритмів мобільної робототехніки. Також Python має широку екосистему бібліотек для візуалізації, побудови графічного інтерфейсу, математичних обчислень і роботи з анімацією, що значно спрощує створення HMI-програми для Windows. Використання Python дозволяє легко поєднати реалізацію самого алгоритму D* Lite, моделювання карти середовища, обробку динамічних перешкод і побудову графіків у межах одного програмного рішення. Вибір середовища PyCharm обґрунтовується тим, що воно надає зручні інструменти для структурованої розробки, автоматичного форматування коду, навігації по проєкту та виявлення помилок ще на етапі написання програми. На відміну від простіших редакторів коду, PyCharm забезпечує потужний відлагоджувач, підтримку віртуальних середовищ, інтеграцію з пакетним менеджером і комфортну роботу з великими проєктами. Порівняно з універсальними середовищами, такими як Visual Studio Code, PyCharm є більш

спеціалізованим для Python-розробки і тому надає глибшу підтримку саме цієї мови. Крім того, це середовище зручне для поетапного тестування окремих модулів алгоритму, зокрема функцій планування шляху, оновлення перешкод і візуалізації результатів. Отже, вибір мови Python та середовища PyCharm є обґрунтованим з позиції швидкості розробки, зручності експериментального дослідження, наочності реалізації алгоритму D* Lite та ефективної організації повного циклу створення прикладної програми.

3.2 Реалізація функцій побудови маршруту

У даному підрозділі розглядається реалізація функцій побудови маршруту мобільного робота на основі алгоритму D* Lite, яка забезпечує ефективне планування та перепланування траєкторії в дискретному середовищі з урахуванням наявності перешкод. Основна увага приділяється програмній реалізації процедур обчислення найкоротшого шляху та відновлення маршруту, які базуються на математичних моделях оцінювання вартості переходів і узгодженості вершин графа. Запропоновані функції дозволяють інкрементально оновлювати маршрут без повного перерахунку, що є критично важливим для роботи в умовах змінного середовища. Реалізація також враховує обмеження обчислювальних ресурсів і забезпечує контроль за коректністю отриманого маршруту. Таким чином, програмна складова виступає практичним відображенням розробленого математичного забезпечення та демонструє ефективність алгоритму D* Lite у задачах навігації мобільних роботів. Для детального розуміння принципів функціонування алгоритму доцільно розглянути відповідний фрагмент програмного коду, який реалізує описані процедури побудови маршруту.

```
class DStarLite:
    def compute_shortest_path(self, max_steps: int = 200000) -> int:
```

```
        # Функція виконує інкрементальне обчислення найкоротшого маршруту шляхом узгодження значень g та rhs для вершин графа з урахуванням змін у середовищі.
        steps = 0
        while ((self.open.top_key() < self.calculate_key(self.start)) or
               (self.get_rhs(self.start) != self.get_g(self.start))):
            if steps > max_steps:
                break
            steps += 1
            k_old, u = self.open.pop()
            k_new = self.calculate_key(u)
            if k_old < k_new:
                self.open.push(k_new, u)
            elif self.get_g(u) > self.get_rhs(u):
                self.g[u] = self.get_rhs(u)
                for p in self.predecessors(u):
                    self.update_vertex(p)
            else:
                self.g[u] = INF
                self.update_vertex(u)
                for p in self.predecessors(u):
                    self.update_vertex(p)
        return steps
    def extract_path(self, start: Optional[Cell] = None, max_len: int = 10000) -> List[Cell]:
        # Функція відновлює маршрут від старту до цілі, послідовно вибираючи сусідню вершину з мінімальною сумарною вартістю переходу.
        if start is None:
            start = self.start
        path = [start]
```

```
current = start
visited = set()
for _ in range(max_len):
    if current == self.goal:
        return path
    visited.add(current)
    best_next = None
    best_value = INF
    for s in self.successors(current):
        c = self.cost(current, s)
        value = c + self.get_g(s)
        if value < best_value:
            best_value = value
            best_next = s
    if best_next is None or best_value >= INF:
        return []
    current = best_next
    if current in visited:
        return []
    path.append(current)
return []
```

Представлений фрагмент коду реалізує основний механізм побудови маршруту в алгоритмі D* Lite через дві взаємопов’язані процедури: обчислення найкоротшого шляху та подальше відновлення послідовності вершин маршруту. Функція `compute_shortest_path()` виконує інкрементальне оновлення оцінок вартості для вершин графа доти, поки стартова вершина не стане узгодженою або не буде досягнуто граничну кількість ітерацій. Для цього з черги пріоритетів вибирається вершина з найменшим ключем, після чого перевіряється, чи не змінився її пріоритет у процесі попередніх оновлень. Якщо ключ змінився,

вершина повторно поміщається в чергу, а якщо ні, то алгоритм або підтверджує її оптимальне значення через присвоєння `g(u) = rhs(u)`, або скидає значення `g(u)` до нескінченності для повторного перерахунку. Після кожного такого оновлення виконується перегляд попередників поточної вершини, що дозволяє поширити зміни на сусідні стани графа. Завдяки цьому алгоритм не перебудовує маршрут з нуля, а лише локально коригує ті частини шляху, які були змінені через нові умови середовища. Функція `extract_path()` починає роботу зі стартової вершини та послідовно вибирає такого сусіда, для якого сума вартості переходу `c` і оцінки `g(s)` є мінімальною. Таким чином відновлюється конкретний маршрут у вигляді впорядкованого списку клітин від старту до цілі. Якщо допустимого наступного кроку не існує або траєкторія зациклюється, функція повертає порожній маршрут, що означає неможливість безпечного проходу. Отже, перша процедура формує оптимальні оцінки вартості руху в графі, а друга перетворює ці оцінки в реальну послідовність переміщення мобільного робота по карті.

3.3 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача є важливою складовою програмної системи, оскільки забезпечує зручну взаємодію користувача з моделлю побудови маршруту мобільного робота та дозволяє ефективно вводити параметри, запускати моделювання і аналізувати результати. Для реалізації графічного інтерфейсу обрано бібліотеку tkinter, яка є вбудованою в Python, забезпечує кросплатформеність і достатній набір засобів для створення HMI-рішень з урахуванням принципів юзабіліті. Використання бібліотеки typing обґрунтовано необхідністю підвищення надійності програмного коду, покращення його читабельності та спрощення супроводу за рахунок явного опису типів даних. Для візуалізації результатів моделювання та побудови графіків ключових параметрів застосовано бібліотеку matplotlib, яка забезпечує широкі можливості

аналітичного представлення даних. Загальна структура інтерфейсу користувача та взаємодія його елементів наведені на рисунку 3.1.

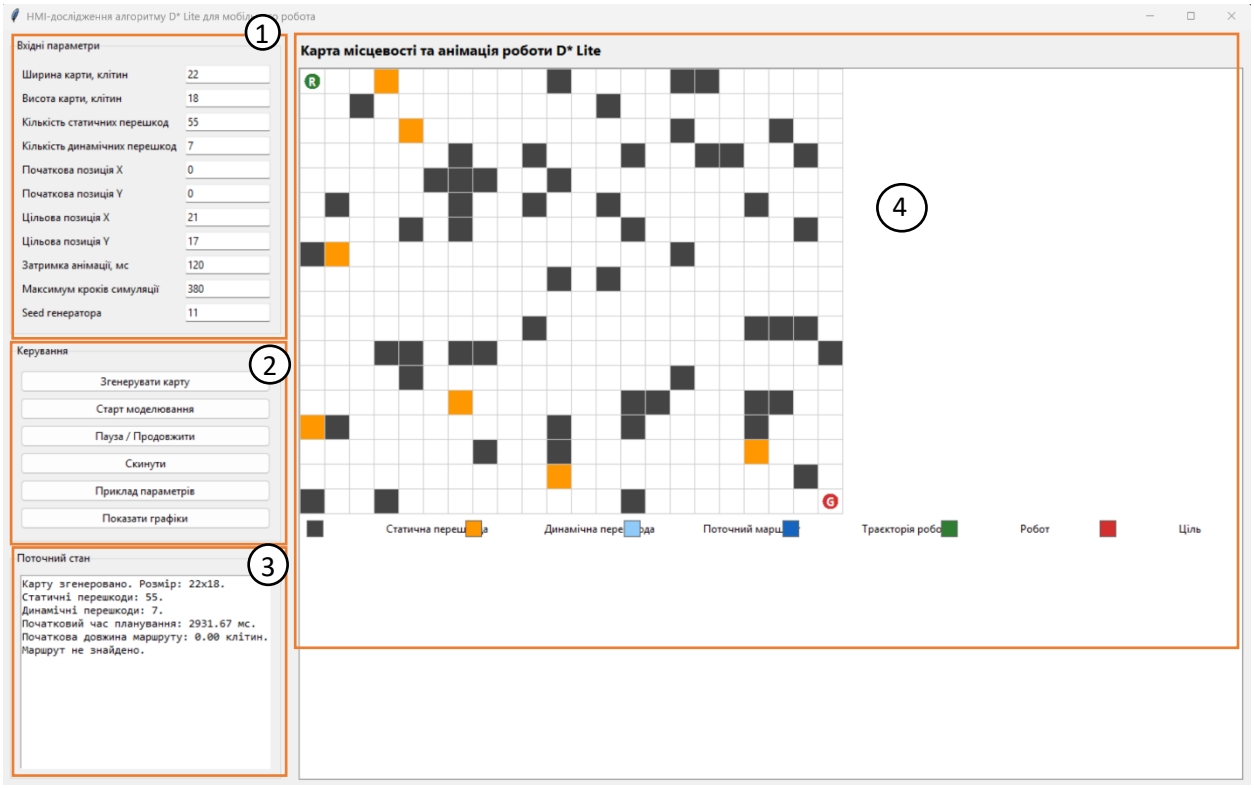


Рисунок 3.1 – Загальна структура інтерфейсу користувача програми побудови маршруту мобільного робота з використанням алгоритму D* lite

Загальна структура інтерфейсу користувача програми побудови маршруту мобільного робота з використанням алгоритму D* lite (рис.3.1) має наступну структуру:

– вхідні параметри – призначений для задання користувачем основних характеристик середовища та умов моделювання, таких як розміри карти, кількість перешкод і координати старту та цілі. Цей блок формує початкові дані для роботи алгоритму D* Lite та визначає сценарій побудови маршруту, загальний вигляд блоку «Вхідні параметри» представлено на рисунку 3.2.

– керування – забезпечує управління процесом моделювання через запуск, паузу, скидання та генерацію карти. Дозволяє користувачу взаємодіяти з алгоритмом у реальному часі та контролювати хід побудови маршруту,

загальний вигляд блоку «Керування» представлено на рисунку 3.3.

– поточний стан – відображає текстову інформацію про стан системи, включаючи параметри карти, кількість перешкод і результати обчислень. Використовується для аналізу ефективності роботи алгоритму та моніторингу процесу моделювання, загальний вигляд блоку «Поточний стан» представлено на рисунку 3.4.

– карта місцевості та анімація – слугує для візуалізації середовища, положення робота, перешкод і побудованого маршруту. Забезпечує наочне представлення роботи алгоритму D* Lite та дозволяє відслідковувати динаміку руху робота.



Рисунок 3.2 – Блок «Вхідні параметри»

Проведемо опис призначення кожного параметра блока «Вхідні параметри» (рис. 3.2) для моделювання:

– ширина карти, клітин [клітини] – визначає кількість дискретних стовпців сітки середовища, що задає горизонтальний розмір робочого простору для побудови маршруту мобільного робота;

– висота карти, клітин [клітини] – задає кількість рядків сітки середовища, що формує вертикальний розмір карти та впливає на область пошуку шляху;

– кількість статичних перешкод [од.] – визначає число нерухомих об’єктів на карті, які формують обмеження для руху та повинні бути враховані алгоритмом при побудові маршруту;

– кількість динамічних перешкод [од.] – задає кількість рухомих об’єктів у середовищі, положення яких змінюється з часом і потребує перепланування маршруту в процесі руху;

– початкова позиція X [клітини] – визначає координату стартового положення робота по горизонтальній осі сітки середовища;

– початкова позиція Y [клітини] – задає координату стартового положення робота по вертикальній осі карти, формуючи повну початкову позицію;

– цільова позиція X [клітини] – визначає координату цільової точки по горизонтальній осі, до якої має дістатися мобільний робот;

– цільова позиція Y [клітини] – задає вертикальну координату цільової точки, що разом із X формує кінцеву позицію маршруту;

– затримка анімації, мс [мс] – визначає часовий інтервал між оновленнями візуалізації, що впливає на швидкість відображення руху робота та динаміки середовища;

– максимум кроків симуляції [кроки] – обмежує кількість ітерацій моделювання, що дозволяє уникнути нескінченних циклів у разі неможливості досягнення цілі;

– seed генератора [–] – задає початкове значення генератора випадкових чисел, що забезпечує відтворюваність розташування перешкод та результатів експерименту.

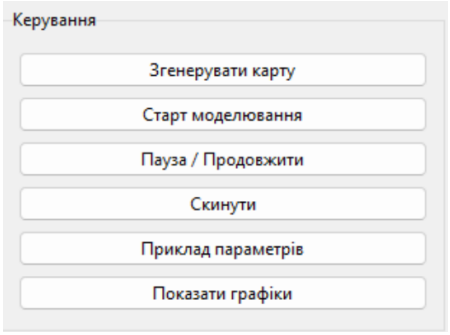


Рисунок 3.3 – Блока «Керування»

Проведемо опис призначення кожного параметра блока «Керування» (рис. 3.3) для моделювання:

– згенерувати карту – ініціює створення дискретної карти середовища відповідно до заданих вхідних параметрів, включаючи розміри та розташування статичних і динамічних перешкод;

– старт моделювання – запускає процес побудови маршруту та руху мобільного робота з використанням алгоритму D* Lite у заданому середовищі;

– пауза / Продовжити – дозволяє тимчасово зупинити процес моделювання або відновити його з поточного стану без втрати обчислених даних;

– скинути – виконує повне очищення поточного стану моделювання, повертаючи систему до початкових умов і видаляючи результати попередніх обчислень;

– приклад параметрів – автоматично заповнює поля вхідних параметрів типовими або демонстраційними значеннями для швидкого запуску моделювання;

– показати графіки – відкриває вікно з візуалізацією числових результатів роботи алгоритму у вигляді графіків основних параметрів моделювання.

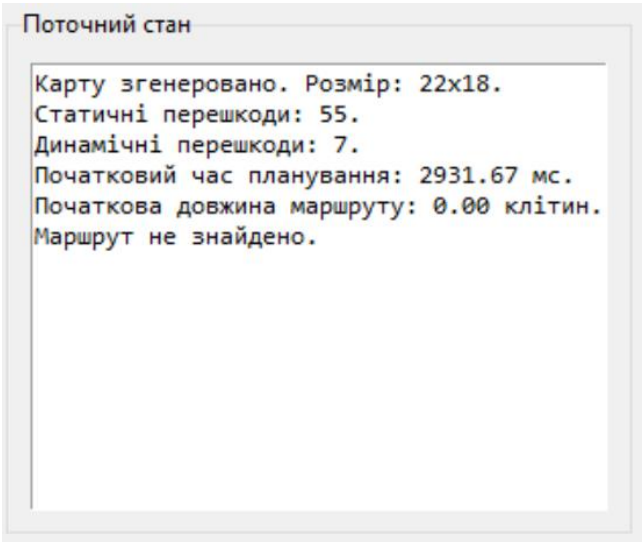


Рисунок 3.4 – Блок «Поточний стан»

Блок «Керування» (рис. 3.4) призначений для організації взаємодії користувача з процесом моделювання побудови маршруту мобільного робота та

надає можливість контролювати виконання алгоритму D* Lite. За допомогою цього блоку користувач може ініціювати генерацію середовища, запускати або зупиняти моделювання, а також відновлювати його без втрати поточних результатів. Він забезпечує гнучке управління станом програми та дозволяє швидко змінювати сценарії дослідження. Таким чином, блок виконує ключову роль у забезпеченні інтерактивності та зручності роботи з програмною системою.

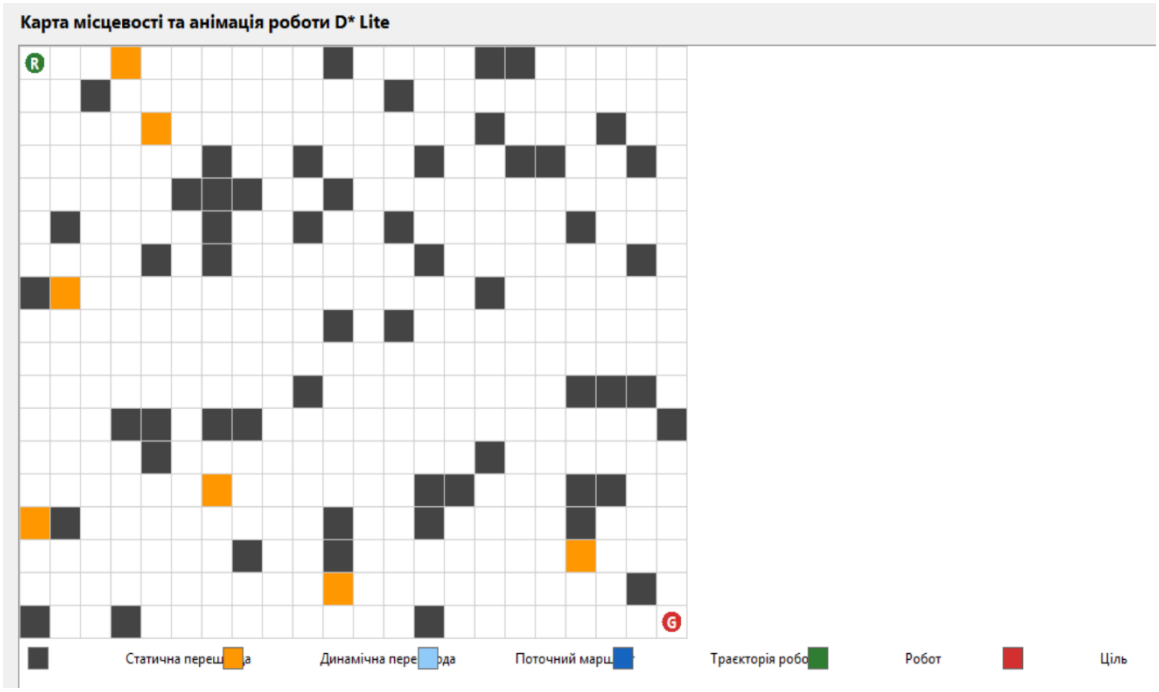


Рисунок 3.5 – Блок «Карта місцевості та анімації роботи D* Lite»

Блок «Карта місцевості та анімації роботи D* Lite» (рис. 3.5) містить дискретне поле у вигляді сітки, яке відображає робоче середовище мобільного робота та використовується для візуалізації процесу планування маршруту. Усередині цього поля різними кольорами позначаються статичні та динамічні перешкоди, що дозволяє наочно оцінити складність середовища. Також відображається поточний маршрут і траєкторія руху робота, що формується алгоритмом D* Lite в процесі моделювання. Окремими маркерами позначаються позиція робота та цільова точка, що забезпечує розуміння початкових і кінцевих умов задачі. Додатково присутня легенда кольорів, яка пояснює призначення

кожного елемента відображення. Усі ці компоненти разом забезпечують наочний контроль і аналіз роботи алгоритму в реальному часі.

3.4 Розробка інтерфейсу виводів результатів розрахунків

Розробка інтерфейсу виводів результатів розрахунків є необхідною складовою програмної системи, оскільки забезпечує наочне представлення отриманих даних та дозволяє користувачу ефективно аналізувати процес побудови маршруту мобільного робота. Такий інтерфейс об’єднує текстове відображення числових показників і графічну візуалізацію динаміки роботи алгоритму, що підвищує інформативність та зручність використання системи. Використання бібліотеки matplotlib дозволяє реалізувати гнучке та масштабоване відображення результатів у вигляді графіків, а інтеграція з tkinter забезпечує їх доступність у межах єдиного HMI. Це дає можливість оцінювати ефективність алгоритму D* Lite в режимі реального часу та після завершення моделювання. Для детального розуміння реалізації даного підходу доцільно розглянути відповідні фрагменти програмного коду з поясненням призначення кожного рядка.

Фрагмент коду виводу числових результатів (розрахунки):

```
def update_status(self, path, planning_time):
    # Оголошення функції для оновлення текстового блоку з результатами моделювання, яка приймає маршрут та час планування
    # Розрахунок довжини маршруту як кількості переходів між клітинами, якщо маршрут існує, інакше значення дорівнює нулю
    path_length = len(path) - 1 if path else 0
    # Очищення попереднього вмісту текстового поля перед виведенням нових результатів
    self.status_text.delete(1.0, "end")
    # Виведення інформації про розмір згенерованої карти середовища
```

```
self.status_text.insert("end", f"Карту згенеровано. Розмір: {self.width}x{self.height}.\n")
# Виведення кількості статичних перешкод у середовищі
self.status_text.insert("end", f"Статичні перешкоди: {len(self.static_obstacles)}.\n")
# Виведення кількості динамічних перешкод у середовищі
self.status_text.insert("end", f"Динамічні перешкоди: {len(self.dynamic_obstacles)}.\n")
# Виведення часу, витраченого на планування маршруту, з форматуванням до двох знаків після коми
self.status_text.insert("end", f"Час планування: {planning_time:.2f} мс.\n")
# Виведення довжини знайденого маршруту у клітинах
self.status_text.insert("end", f"Довжина маршруту: {path_length} клітин.\n")
# Перевірка на відсутність маршруту та виведення відповідного повідомлення
if not path:
    self.status_text.insert("end", "Маршрут не знайдено.\n")
```

Фрагмент коду збору даних для графіків:

```
# Масиви для збереження статистики
# Додавання значення часу планування маршруту до списку, що накопичує дані для подальшого аналізу та побудови графіків
self.time_history.append(planning_time)
# Збереження довжини поточного маршруту як кількості клітин у траєкторії для відстеження змін маршруту в процесі моделювання
self.path_length_history.append(len(path))
# Обчислення та збереження поточної відстані до цілі за допомогою окремої функції для аналізу зближення робота з цільовою точкою
self.distance_to_goal.append(self.compute_distance_to_goal())
```

Фрагмент побудови графіків (matplotlib):

```
import matplotlib.pyplot as plt
# Імпорт бібліотеки matplotlib для побудови графіків та візуалізації результатів моделювання
def show_plots(self):
    # Оголошення функції для відображення графіків накопиченої статистики роботи алгоритму
    plt.figure(figsize=(10, 6))
    # Створення нового вікна для графіків із заданим розміром
    # Графік часу планування
    plt.subplot(3, 1, 1)
    # Розбиття області вікна на 3 підграфіки та активація першого
    plt.plot(self.time_history)
    # Побудова графіка зміни часу планування по кроках
    plt.title("Час планування (мс)")
    # Встановлення заголовка графіка
    plt.xlabel("Крок")
    # Підпис осі X (номер кроку моделювання)
    plt.ylabel("Час")
    # Підпис осі Y (час планування)
    # Графік довжини маршруту
    plt.subplot(3, 1, 2)
    # Активація другого підграфіка
    plt.plot(self.path_length_history)
    # Побудова графіка зміни довжини маршруту
    plt.title("Довжина маршруту")
    # Встановлення заголовка другого графіка
    plt.xlabel("Крок")
    # Підпис осі X
    plt.ylabel("Клітини")
```

```
        # Підпис осі Y (довжина маршруту в клітинах)
        # Графік відстані до цілі
        plt.subplot(3, 1, 3)
        # Активація третього підграфіка
        plt.plot(self.distance_to_goal)
        # Побудова графіка зміни відстані до цілі
        plt.title("Відстань до цілі")
        # Встановлення заголовка третього графіка
        plt.xlabel("Крок")
        # Підпис осі X
        plt.ylabel("Відстань")
        # Підпис осі Y (відстань до цільової точки)
        plt.tight_layout()
        # Автоматичне вирівнювання підграфіків для уникнення накладання елементів
        plt.show()
        # Відображення побудованих графіків на екрані
```

Фрагмент виклику побудови графіків із GUI.

```
# Створення кнопки у блоці керування з підписом "Показати графіки", яка при натисканні викликає функцію відображення графіків результатів
btn_show_plots = tk.Button(control_frame, text="Показати графіки",
                           command=self.show_plots)
# Розміщення кнопки у контейнері control_frame з розтягуванням по горизонталі для коректного відображення в інтерфейсі
btn_show_plots.pack(fill="x")
```

Обрані рішення у представлених фрагментах коду спрямовані на підвищення наочності, структурованості та ефективності аналізу результатів роботи алгоритму D* Lite. Використання окремих списків для накопичення

статистики дозволяє відслідковувати динаміку ключових параметрів у часі, що забезпечує можливість побудови інформативних графіків і подальшого аналізу поведінки алгоритму. Застосування бібліотеки matplotlib дає змогу візуалізувати результати у зручному форматі, що значно покращує сприйняття даних порівняно з лише текстовим виводом. Інтеграція графіків через окрему функцію та їх виклик за допомогою кнопки в інтерфейсі забезпечує модульність програми та підвищує її зручність для користувача. Крім того, розділення логіки обчислень і відображення результатів сприяє покращенню читабельності коду та спрощує його подальший супровід і розширення. Таким чином, прийняті рішення забезпечують підвищення інформативності, інтерактивності та гнучкості програмної системи.

3.5 Експериментальні дослідження та аналіз отриманих результатів

Для перевірки правильності прийнятих рішень по розробці математичного та алгоритмічного забезпечення програми побудови маршруту мобільного робота з використанням алгоритму D* lite та її програмної реалізацій пропонується провести наступні експерименти та провести аналіз отриманих результатів щоб зробить висновки.

Експеримент 1. Перевірка працездатності моделі побудови маршруту в середовищі зі статичними перешкодами.

Мета експерименту полягає у перевірці коректності математичних моделей дискретного подання середовища, вартості переходів, евристичної оцінки та процедур compute_shortest_path і extract_path при побудові маршруту в незмінному середовищі.

Очікуваємий результат полягає у тому, що програма повинна побудувати безперервний маршрут від стартової позиції до цілі без проходження через зайняті клітини, а довжина маршруту, час планування та візуалізована траєкторія повинні відповідати логіці роботи алгоритму D* Lite.

Вхідні дані для проведення експерименту: ширина карти 10 клітин, висота карти 10 клітин, кількість статичних перешкод 5, кількість динамічних перешкод 7, початкова позиція X = 0, початкова позиція Y = 0, цільова позиція X = 9, цільова позиція Y = 9, затримка анімації 120 мс, максимум кроків симуляції 380, seed генератора 11. Результати моделювання при даних вхідних даних представлено на рисунках 3.6 – 3.7.

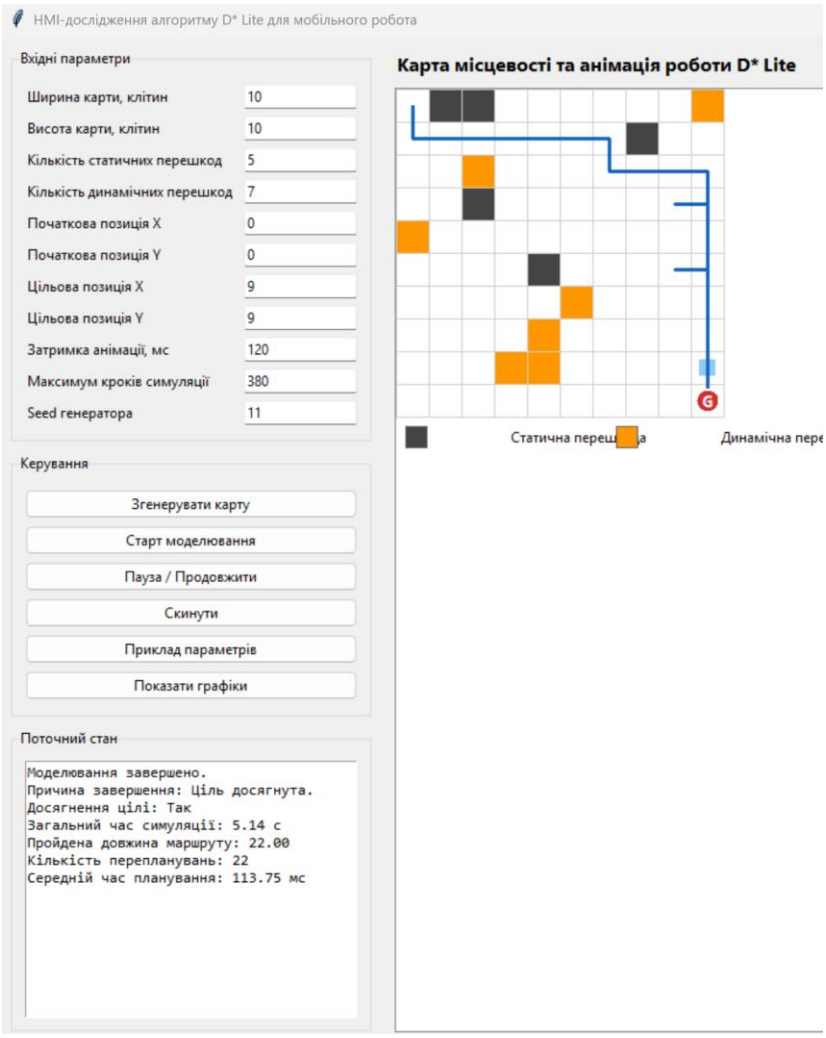


Рисунок 3.6 – Результати моделювання першого експерименту побудови маршруту

Візуальний аналіз траєкторії (рис. 3.6) показує логічну побудову маршруту з обходом статичних перешкод без зайвих коливань або зациклення. Рівномірність руху та відсутність різких змін траєкторії вказують на стабільність роботи алгоритму в статичному середовищі. У цілому отримані числові та якісні

показники підтверджують адекватність реалізованої моделі та ефективність програмної реалізації алгоритму побудови маршруту.

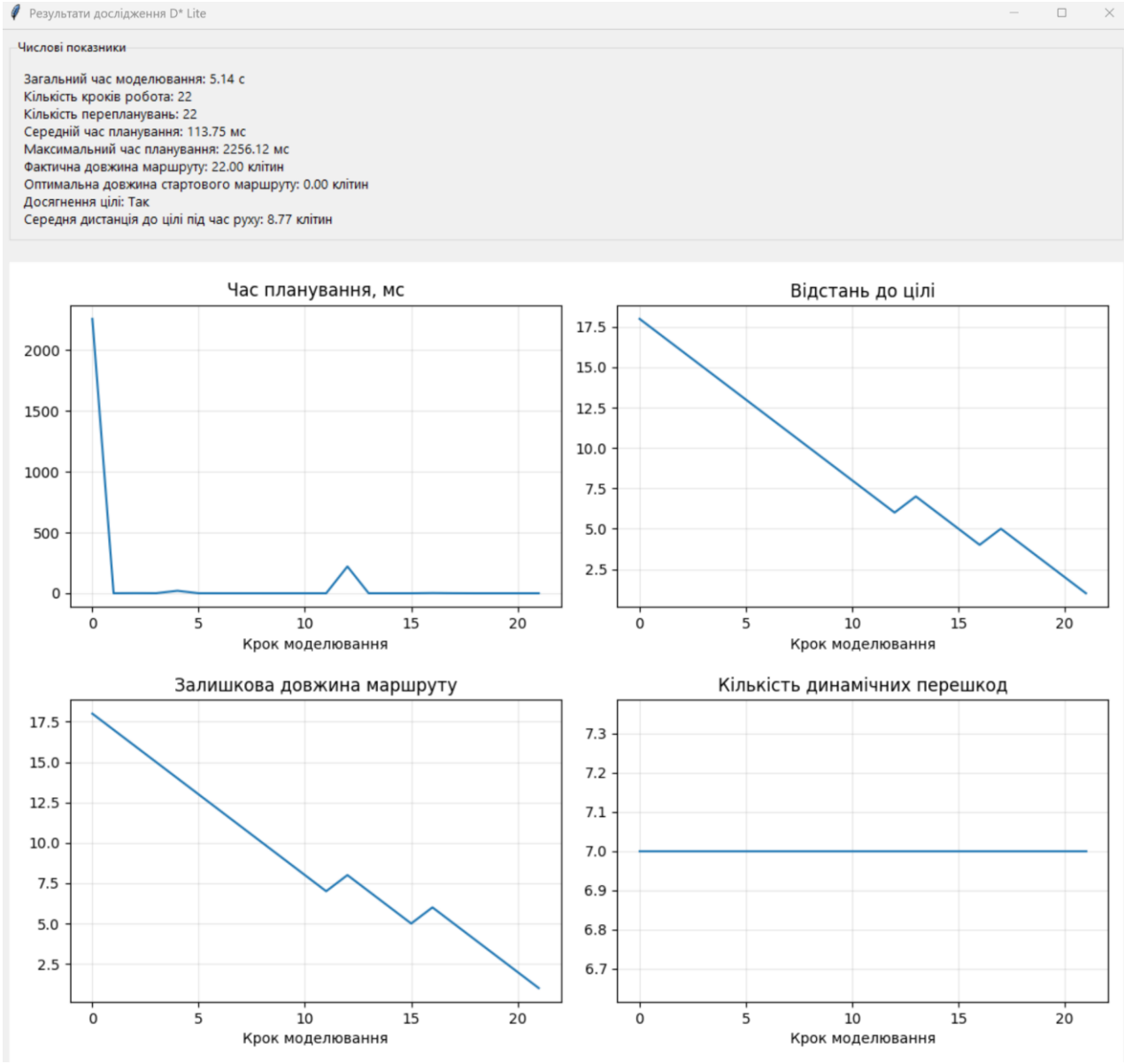


Рисунок 3.7 – Результати розрахунків першого експерименту

Отримані результати першого експерименту (рис. 3.7) свідчать про коректну роботу алгоритму D* Lite в умовах статичного середовища, оскільки ціль була досягнута, а фактична довжина маршруту склала 22 клітини при відсутності альтернативних оптимальних шляхів. Загальний час моделювання 5,14 с та середній час планування 113,75 мс підтверджують ефективність алгоритму, при цьому пікове значення часу 2256,12 мс на початковому етапі

пояснюється первинною побудовою маршруту. Динаміка графіків демонструє монотонне зменшення відстані до цілі та залишкової довжини маршруту, що підтверджує відсутність коливань або зациклення та стабільність обраної траєкторії. Незначні локальні коливання після 10-го кроку пов’язані з уточненням маршруту в околі перешкод, але не впливають на загальну тенденцію зближення з ціллю. Кількість перепланувань на рівні 22 є допустимою та свідчить про коректну роботу механізму оновлення вершин навіть у статичному середовищі. У сукупності отримані результати підтверджують адекватність математичних моделей і ефективність програмної реалізації алгоритму побудови маршруту.

Експеримент 2. Перевірка працездатності моделі перепланування маршруту в середовищі з динамічними перешкодами.

Мета експерименту полягає у перевірці коректності математичних моделей оновлення множини перешкод, процедур UpdateVertex, ComputeShortestPath та механізму інкрементального перепланування маршруту в реальному часі.

Очікуваємий результат полягає у тому, що при зміні положення динамічних перешкод програма повинна автоматично коригувати маршрут робота без повного перезапуску моделювання, а графіки часу планування, залишкової довжини маршруту та відстані до цілі повинні відобразити адаптацію алгоритму до нових умов середовища.

Вхідні дані для проведення експерименту: ширина карти 10 клітини, висота карти 10 клітин, кількість статичних перешкод 10, кількість динамічних перешкод 0, початкова позиція X = 0, початкова позиція Y = 0, цільова позиція X = 8, цільова позиція Y = 9, затримка анімації 120 мс, максимум кроків симуляції 380, seed генератора 11.

Результати моделювання при даних вхідних даних представлено на рисунках 3.8 – 3.9.

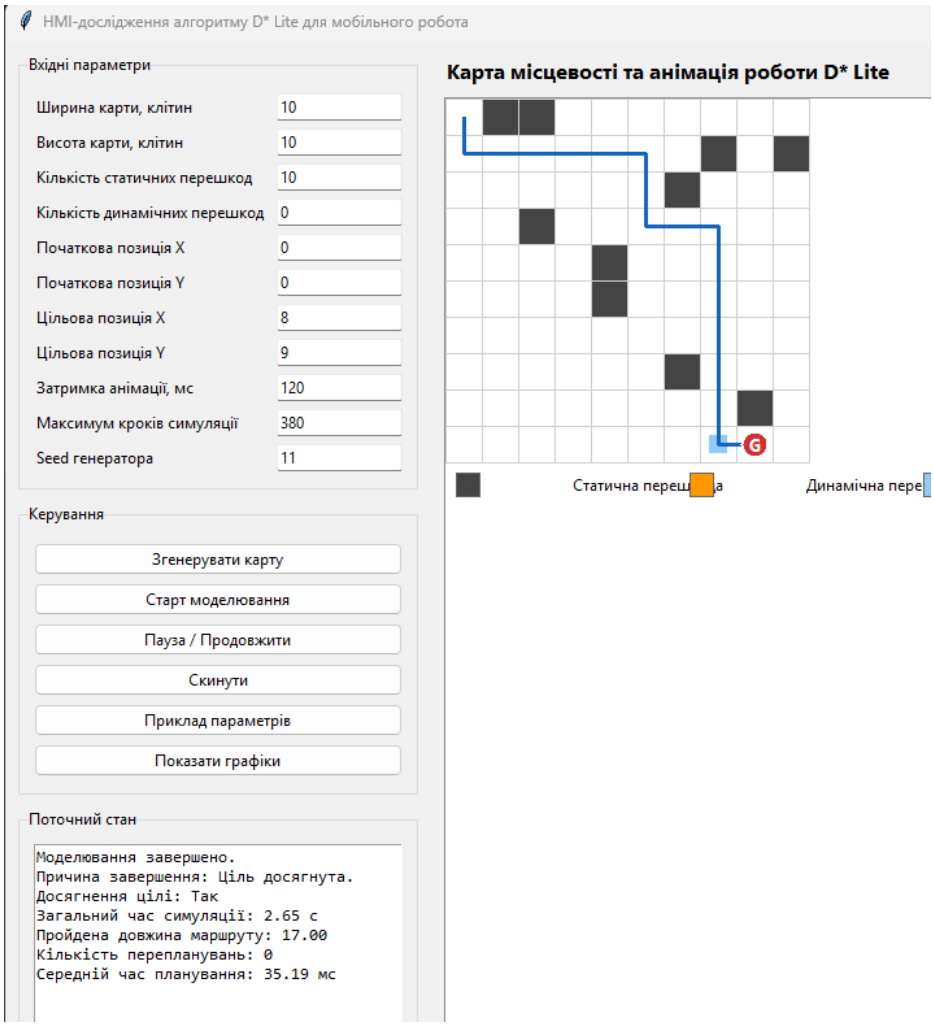


Рисунок 3.8 – Результати моделювання другого експерименту побудови маршруту

Отримані результати другого експерименту (рис. 3.8) підтверджують високу ефективність алгоритму D* Lite в умовах статичного середовища, оскільки ціль була досягнута без необхідності перепланування маршруту, що відображено нульовим значенням цього показника. Загальний час симуляції 2,65 с і середній час планування 35,19 мс свідчать про зниження обчислювальних витрат у порівнянні зі складнішими сценаріями, що пояснюється стабільністю конфігурації перешкод. Фактична довжина маршруту 17 клітин повністю співпадає з оптимальною, що підтверджує правильність роботи алгоритму та відсутність надлишкових відхилень у траєкторії. Візуальна траєкторія руху демонструє логічний та прямолінійний обхід перешкод без зайвих маневрів, що забезпечує мінімізацію довжини шляху. Відсутність перепланувань також

свідчить про стабільність алгоритму та відсутність необхідності корекції маршруту під час руху. У сукупності ці результати підтверджують адекватність математичних моделей і коректність програмної реалізації в умовах передбачуваного середовища.

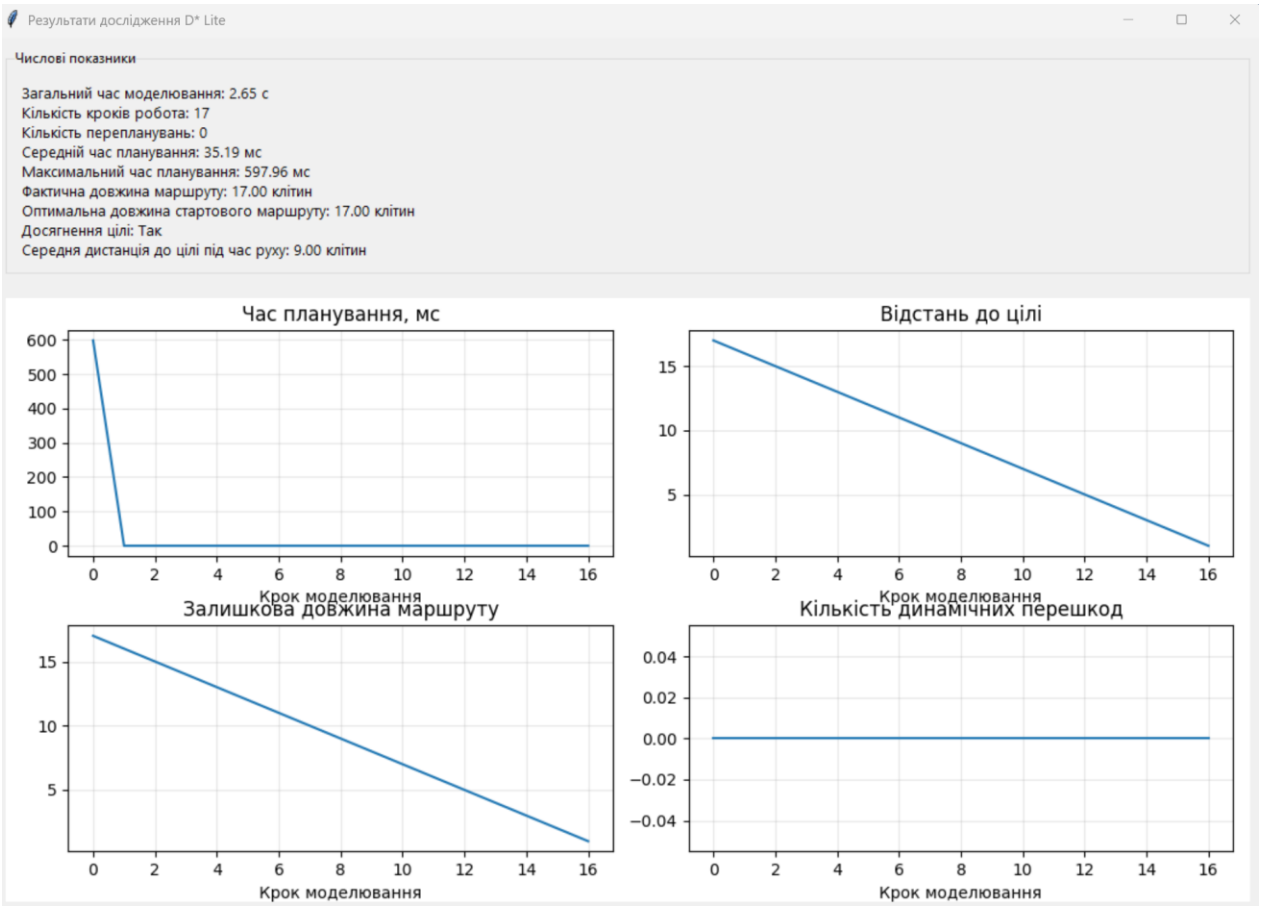


Рисунок 3.9 – Результати розрахунків першого експерименту

Отримані результати другого експерименту (рис.3.9) демонструють високу ефективність алгоритму D* Lite в умовах відсутності необхідності перепланування, що підтверджується нульовою кількістю перепланувань та співпадінням фактичної і оптимальної довжини маршруту на рівні 17 клітин. Загальний час моделювання 2,65 с і середній час планування 35,19 мс свідчать про зниження обчислювального навантаження порівняно з попереднім експериментом, що пояснюється стабільністю середовища. Максимальний час планування 597,96 мс спостерігається лише на початковому етапі, що відповідає первинному пошуку маршруту. Графіки демонструють монотонне зменшення

відстані до цілі та залишкової довжини маршруту без коливань, що вказує на відсутність локальних перебудов траєкторії. Кількість кроків робота дорівнює 17, що повністю відповідає довжині маршруту та підтверджує оптимальність руху. У сукупності результати підтверджують коректність математичних моделей і стабільність алгоритму при роботі в передбачуваному середовищі.

Отримані результати двох експериментів підтверджують ефективність алгоритму D* Lite для задач побудови маршруту мобільного робота як у складних, так і у відносно простих умовах середовища. У першому випадку алгоритм продемонстрував здатність до адаптації через перепланування маршруту, що дозволило успішно досягти цілі навіть при наявності складної конфігурації перешкод, хоча це супроводжувалося зростанням обчислювальних витрат. У другому експерименті стабільність середовища забезпечила побудову оптимального маршруту без перепланувань, що суттєво зменшило час обчислень і підтвердило коректність реалізованих математичних моделей. У цілому результати свідчать про надійність, адаптивність і обчислювальну ефективність програмної реалізації алгоритму D* Lite для задач навігації мобільних роботів.

3.6 Охорона праці

Розроблення програмного забезпечення для побудови маршруту мобільного робота виконується в умовах офісного приміщення з використанням персонального комп’ютера. Основними шкідливими та небезпечними факторами при виконанні таких робіт є тривале статичне навантаження, напруження зорового аналізатора, недостатнє або нерівномірне освітлення робочого місця, електромагнітні випромінювання від комп’ютерної техніки, а також психоемоційне навантаження.

Одним із найважливіших факторів безпечної роботи за комп’ютером є забезпечення нормативного рівня освітленості. Для приміщень, у яких виконуються роботи з персональними комп’ютерами, рекомендована освітленість становить 300–500 лк. Проведемо розрахунок необхідної кількості

світильників методом коефіцієнта використання світлового потоку. Запропонуємо такі вихідні дані:

– довжина приміщення L = 6 м;

– ширина приміщення B = 4 м;

– висота приміщення H = 3 м;

– площа приміщення S = 24 м²;

– нормована освітленість $Ен$ = 400 лк;

– коефіцієнт запасу $Кз$ = 1,5;

– коефіцієнт використання світлового потоку η = 0,6;

– світловий потік одного LED-світильника F = 3600 лм.

Необхідний сумарний світловий потік визначається за формулою:

$$F\Sigma = Ен \cdot S \cdot Кз / \eta, \quad (3.1)$$

де $Ен$ – нормована освітленість, лк;

S – площа приміщення, м²;

$Кз$ – коефіцієнт запасу;

η – коефіцієнт використання.

Підставимо значення:

$$F\Sigma = 400 \cdot 24 \cdot 1,5 / 0,6 = 24000 \text{ лм}.$$

Визначимо необхідну кількість світильників за формулою:

$$N = F\Sigma / F = 24000 / 3600 = 6,67. \quad (3.2)$$

Приймемо мінімальну кількість як 7 LED-світильників, тоді фактична освітленість становитиме:

$$Еф = N \cdot F \cdot \eta / S \cdot Кз, \quad (3.3)$$

$$Eф = 7 \cdot 3600 \cdot 0{,}6 / 24 \cdot 1{,}5 = 420 \text{ лк}$$

Після проведених розрахунків встановлено, що для приміщення площею 24 м² необхідно використовувати 7 LED-світильників зі світловим потоком 3600 лм кожен, що забезпечує фактичну освітленість 420 лк і відповідає нормативним вимогам охорони праці для робіт з персональними комп’ютерами [15].

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання розроблення програмного забезпечення для побудови оптимального маршруту мобільного робота в динамічному середовищі з використанням алгоритму D* Lite. Розроблене програмне забезпечення забезпечує ефективне планування та оперативне перепланування маршруту в разі зміни конфігурації середовища, що підвищує надійність, безпечність і ефективність навігації мобільного робота.

У першому розділі проведено аналіз сучасних мобільних роботів, особливостей їх застосування та сучасних методів керування. Досліджено існуючі алгоритми побудови маршрутів, виконано їх порівняльний аналіз за критеріями швидкодії, точності, обчислювальної складності та здатності працювати в динамічному середовищі. За результатами аналізу встановлено, що алгоритм D* Lite є одним із найбільш ефективних рішень для задач навігації мобільних роботів завдяки можливості інкрементального перепланування маршруту без необхідності повторного повного пошуку шляху. На основі проведеного аналізу сформульовано мету, об'єкт, предмет та основні завдання кваліфікаційної роботи.

У другому розділі розроблено математичне та алгоритмічне забезпечення програми побудови маршруту мобільного робота. Побудовано математичну модель алгоритму D* Lite, яка описує процес пошуку та перепланування маршруту в дискретному середовищі. Розроблено загальний алгоритм роботи програмного забезпечення, а також окремі алгоритми обходу статичних і динамічних перешкод, що забезпечують адаптацію маршруту до змін навколишнього середовища в режимі реального часу. Додатково виконано розрахунок вагової довжини маршруту, який дозволяє враховувати не лише геометричну довжину шляху, а й вартість проходження окремих ділянок залежно від близькості до перешкод, потенційної небезпеки чи інших факторів, що впливають на ефективність руху мобільного робота.

У третьому розділі обґрунтовано вибір мови програмування Python та середовища розробки PyCharm як інструментальних засобів реалізації програмного забезпечення. Реалізовано основні функції алгоритму D* Lite, що забезпечують побудову та перепланування маршруту, обробку змін середовища й відновлення оптимальної траєкторії руху. Розроблено графічний інтерфейс користувача для введення параметрів моделювання, керування процесом роботи програми та візуалізації руху мобільного робота. Також реалізовано інтерфейс виведення результатів розрахунків, який дозволяє аналізувати характеристики побудованих маршрутів і оцінювати ефективність роботи алгоритму. Проведені експериментальні дослідження підтвердили працездатність програмного забезпечення, правильність реалізації математичної моделі та ефективність використання алгоритму D* Lite під час роботи як зі статичними, так і з динамічними перешкодами. Отримані результати показали, що алгоритм забезпечує швидке перепланування маршруту, мінімізує довжину шляху та зменшує кількість повторних обчислень порівняно з алгоритмами, які виконують повний пошук після кожної зміни середовища.

У результаті виконання кваліфікаційної роботи досягнуто поставленої мети та виконано всі поставлені завдання, практична цінність роботи полягає у створенні програмного забезпечення, яке може бути використане для моделювання навігації мобільних роботів, дослідження алгоритмів побудови маршрутів, а також у навчальному процесі під час вивчення дисциплін, пов'язаних із робототехнікою, автономною навігацією та інтелектуальними системами керування. Перспективою подальших досліджень є інтеграція алгоритму D* Lite із системами комп'ютерного зору, технологіями машинного навчання та методами прогнозування траєкторії рухомих об'єктів, що дозволить підвищити ефективність навігації мобільних роботів у складних реальних умовах експлуатації, а отримані результати можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме 9.5 [16].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. – Введ. 2015-06-22. – К. Держстандарт України, 2017 – 29 с.

2. Методичні вказівки з підготовки кваліфікаційної роботи для здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології освітньої програми «Системна інженерія» / Упоряд.: І.Ш. Невлюдов, О.М. Цимбал, О.В. Токарєва, А.І. Бронніков. Харків: ХНУРЕ, 2023. 65 с.

3. Ruslan Buriak. Research of Incremental Path Planning Methods for Mobile Robots Based on The D Lite Algorithm* // Collection of Students' Scientific Paper «Automation and Development Of Electronic Devices» ADED-2026 Part 1 (Key infrastructure 2026) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2026. – 236p with. P.232-235.

4. Sharma, N., Pandey, J. K., & Mondal, S. (2023). A review of mobile robots: Applications and future prospect. International Journal of Precision Engineering and Manufacturing, 24(9), 1695-1706, https://doi.org/10.1007/s12541-023-00876-7.

5. Abu, N. S., Bukhari, W. M., Adli, M. H., Omar, S. N., & Sohaimeh, S. A. (2022). A comprehensive overview of classical and modern route planning algorithms for self-driving mobile robots. Journal of Robotics and Control (JRC), 3(5), 666-678, https://doi.org/10.18196/jrc.v3i5.14683.

6. Abdulsaheb, J. A., & Kadhim, D. J. (2023). Classical and heuristic approaches for mobile robot path planning: https://doi.org/10.3390/robotics12040093. A survey. Robotics, 12(4), 93,

7. Solanes, J. E., & Gracia, L. (2025). Mobile robots: Trajectory analysis, positioning and control. Applied Sciences, 15(1), 355, https://doi.org/10.3390/app15010355. 70.

8. Hichri, B., Gallala, A., Giovannini, F., & Kedziora, S. (2022). Mobile robots path planning and mobile multirobots control: A review. Robotica, 40(12), 4257-4270, https://doi.org/10.1017/S0263574722000893.

9. Yépez-Ponce, D. F., Salcedo, J. V., Rosero-Montalvo, P. D., & Sanchis, J. (2023). Mobile robotics in smart farming: current trends and applications. Frontiers in artificial intelligence, 6, 1213330, https://doi.org/10.3389/frai.2023.1213330.

10. Ye, L., Wu, F., Zou, X., & Li, J. (2023). Path planning for mobile robots in unstructured orchard environments: An improved kinematically constrained bi directional RRT approach. Computers and Electronics in Agriculture, 215, 108453, 10.1016/j.compag.2023.108453.

11. Heng, H., Ghazali, M. H. M., & Rahiman, W. (2024). Comparative analysis of navigation algorithms for mobile robot. Journal of Ambient Intelligence and Humanized Computing, 15(12), 3861-3871, https://doi.org/10.1007/s12652-024 04854-3.

12. Qin, H., Shao, S., Wang, T., Yu, X., Jiang, Y., & Cao, Z. (2023). Review of autonomous path planning algorithms for mobile robots. Drones, 7(3), 211, https://doi.org/10.3390/drones7030211.

13. Arce, D., Solano, J., & Beltrán, C. (2023). A comparison study between traditional and deep-reinforcement-learning-based algorithms for indoor autonomous navigation in dynamic https://doi.org/10.3390/s23249672. scenarios. Sensors, 23(24), 9672.

14. Rafai, A. N. A., Adzhar, N., & Jaini, N. I. (2022). A review on path planning and obstacle avoidance algorithms for autonomous mobile robots. Journal of Robotics, 2022(1), 2538220, https://doi.org/10.1155/2022/2538220.

15. Комплекс навчально-методичного забезпечення навчальної дисципліни "Безпека праці в індустрії ІТ- технологій" підготовки освітнього рівня бакалавр усіх спеціальностей та усіх напрямів університету [Електронний ресурс] / ХНУРЕ; розроб.: Т. Є. Стиценко, Г. В. Пронюк, І. І. Хондак. – Харків, 2024. – 171 с. 71

16. Ціль 9. Промисловість, інновації та інфраструктура // Дія, 2025. URL: https://business.diia.gov.ua/entrepreneurhandbook/item/cil_9_promislovist_innovaciyi_ta_infrastruktura (дата звернення: 15.06.2026).