Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

КОМП'ЮТЕРНА СИСТЕМА СКАНУВАННЯ ТА ВІДОБРАЖЕННЯ ДЛЯ ОЗДОБЛЕННЯ ПРИМІЩЕНЬ

Виконав студента 2 курсу групи 1КІ-24м
спеціальності 123 — Комп'ютерна інженерія
_____ Тушинський В.Е.

Керівник: к.т.н., доц. кафедри ОТ
_____ Томчук М.А.
«15» 12 2025 р.

Опонент: к.т.н., доц. кафедри ПЗ
_____ Черноволик Г.О.
«15» 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«18» 12 2025 р

ВНТУ 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Рівень вищої освіти II—й (магістерський)

Галузь знань — 12 «Інформаційні технології»

Спеціальність — 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., професор

Азаров О.Д. _____

"25" вересня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Тушинському Віталію Едуардовичу

1 Тема роботи — Комп'ютерна система сканування та відображення для оздоблення приміщень, керівник роботи Томчук Микола Антонович, к.т.н., доцент кафедри ОТ, затверджені наказом ВНТУ від "24" вересня 2025 року № 313.

2 Строк подання студентом роботи — 4 грудня 2025 року.

3 Вихідні дані до роботи: мобільна платформа на базі робота-пилососа Rowenta X-plorer Serie 135, одноплатний комп'ютер Raspberry Pi 4 Model B (4 ГБ), лазерний далекомір TF-Luna LiDAR, сервопривід MG996R, інерціальний модуль MPU-6050, мова програмування Python 3.9, бібліотека візуалізації Three.js, протокол обміну WebSocket, операційна система Raspberry Pi OS.

4 Зміст текстової частини: вступ; огляд і аналіз методів сканування приміщень для оздоблювальних робіт; теоретичні дослідження методів адаптивного геометричного аналізу приміщень; розробка та експериментальне дослідження системи; економічна частина; висновки.

5 Перелік графічного матеріалу: структурна схема апаратної частини

системи, блок-схема алгоритму класифікації приміщень, блок-схема модифікованого алгоритму RRT, схема підключення компонентів до Raspberry Pi.

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 — Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	Завдання прийняв
1— 3	Томчук М.А., к.т.н., доц. каф. ОТ		
4	Ратушняк О.Г., к.т.н., доц. каф. ЕПВМ		
Нормоконтроль	асистент каф. ОТ Швець С.І.		

7 Дата видачі завдання — 29.09.2025 року

8 Календарний план представлено в таблиці 2

Таблиця 2 — Календарний план

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітк
1	Огляд і аналіз методів сканування приміщень для оздоблювальних робіт	08.09.25 — 12.09.25	Вик.
2	Теоретичні дослідження методів адаптивного геометричного аналізу приміщень	15.09.25 — 19.09.25	Вик.
3	Розробка та експериментальне дослідження системи	22.09.25 — 03.10.25	Вик.
4	Економічна частина	06.10.25 — 15.10.25	Вик.
5	Оформлення пояснювальної записки, графічного матеріалу і презентації	17.10.25 — 25.10.25	Вик.
6	Підготовка супроводжуючих документів	30.10.25	Вик.

Студент (підпис) Тушинський В.І
(прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи (підпис) Томчук М.А.
(прізвище та ініціали)

АНОТАЦІЯ

УДК 004.932:528.7:681.5

Тушинський В. Е. Комп'ютерна система сканування та відображення для оздоблення приміщень. Магістерська кваліфікаційна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця : ВНТУ, 2025. 129 с.

Укр. мовою. Бібліогр.: 37 назв; рис.: 20; табл.: 27.

У магістерській кваліфікаційній роботі розроблено комп'ютерну систему автоматизованого сканування приміщень для оздоблювальних робіт. Запропоновано математичну модель класифікації приміщень на основі трьох геометричних інваріантів: коефіцієнта витягнутості, коефіцієнта складності периметра та коефіцієнта заповнення перешкодами. Розроблено алгоритми адаптивного планування траєкторії сканування, включаючи модифікований метод швидкого дослідження випадкових дерев із функцією привабливості невідсканованих областей. Спроектовано апаратну частину на базі модифікованого робота-пилососа з лазерним далекоміром TF-Luna та одноплатним комп'ютером Raspberry Pi. Створено веб-інтерфейс візуалізації з використанням бібліотеки Three.js. Експериментальні дослідження підтвердили підвищення коефіцієнта повноти покриття порівняно з фіксованими стратегіями сканування.

Ключові слова: сканування приміщень, адаптивне планування траєкторії, геометричні інваріанти, класифікація приміщень, лазерний далекомір, мобільна платформа, slam, raspberry pi, веб-візуалізація.

ABSTRACT

UDC 004.932:528.7:681.5

Tushynskyi V. E. Computer system for scanning and visualization for room renovation. Master's thesis in specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia : VNTU, 2025. 129 p.

In Ukrainian. Bibliography: 37 titles; figures: 20; tables: 27.

The master's thesis presents the development of a computer system for automated room scanning for finishing works. A mathematical model for room classification based on three geometric invariants is proposed: elongation coefficient, perimeter complexity coefficient, and obstacle density coefficient. Adaptive scanning trajectory planning algorithms have been developed, including a modified rapidly-exploring random trees method with an attraction function for unscanned areas. The hardware part is designed based on a modified robot vacuum cleaner with TF-Luna laser rangefinder and Raspberry Pi single-board computer. A web-based visualization interface using the Three.js library has been created. Experimental studies confirmed an increase in coverage completeness coefficient compared to fixed scanning strategies.

Keywords: room scanning, adaptive trajectory planning, geometric invariants, room classification, laser rangefinder, mobile platform, slam, raspberry pi, web visualization.

ЗМІСТ

ВСТУП

Кожен, хто хоча б раз замовляв ремонт квартири, знає, скільки часу займає етап обмірювання: майстер з рулеткою або лазерним далекоміром годинами вимірює кожну стіну, записує результати в блокнот, а потім вручну переносить їх у креслення. При цьому одна помилка у вимірюваннях може призвести до нестачі матеріалів посеред ремонту або до значних фінансових втрат через їх надлишкове замовлення. Розвиток технологій робототехніки та лазерного сканування відкриває можливість автоматизувати цей рутинний процес, проте існуючі професійні рішення коштують як невеликий автомобіль, що робить їх недоступними для більшості виконавців оздоблювальних робіт.

Ідея використати робота-пилососа як базу для вимірювальної системи виникла не випадково: ці пристрої вже оснащені всім необхідним для навігації у приміщенні — колісною платформою, датчиками перешкод та достатньо потужним процесором. Масове виробництво таких роботів суттєво знизило вартість компонентів, а відкриті проекти на кшталт Valetudo довели можливість модифікації їхнього програмного забезпечення [1]. Залишалося лише додати точніший лазерний далекомір та навчити систему не просто будувати карту для прибирання, а виконувати повноцінні геометричні вимірювання з точністю, достатньою для професійного використання.

Актуальність теми обумовлена тим, що існуючі системи автоматизованого сканування приміщень використовують однакову стратегію незалежно від того, чи сканується порожня квадратна кімната, чи вузький коридор із нішами, чи Г-подібне приміщення зі складною геометрією. Такий підхід є неефективним: у простих приміщеннях витрачається зайвий час на надлишкові вимірювання, а у складних залишаються неохоплені ділянки. Водночас вартість професійних 3D-сканерів починається від кількох сотень тисяч гривень, що унеможливлює їх використання малими будівельними бригадами та приватними майстрами. Це створює потребу у розробці доступної системи, яка б розпізнавала тип приміщення та автоматично обирала

оптимальний спосіб його сканування.

Мета роботи полягає у підвищенні точності та повноти геометричних вимірювань приміщень шляхом розробки комп'ютерної системи сканування та відображення з адаптивним плануванням траєкторії на основі класифікації просторових характеристик.

Для досягнення поставленої мети необхідно вирішити такі **задачі**:

— провести аналіз методів вимірювання приміщень та існуючих систем автоматизованого сканування;

— розробити математичну модель класифікації приміщень на основі геометричних інваріантів;

— розробити алгоритми адаптивного планування траєкторії сканування для різних типів приміщень;

— спроектувати апаратну частину системи на базі мобільної платформи;

— розробити програмне забезпечення мобільної платформи та веб-інтерфейс візуалізації;

— провести експериментальні дослідження та оцінити ефективність розробленої системи.

Об'єктом дослідження є процес автоматизованого вимірювання геометричних параметрів приміщень для оздоблювальних робіт.

Предметом дослідження є методи та засоби адаптивного планування траєкторії сканування приміщень на основі класифікації їх просторових характеристик.

Методи дослідження. Системний аналіз застосовано для дослідження предметної області та порівняння існуючих рішень. Методи аналітичної геометрії використано при розробці математичної моделі класифікації приміщень за геометричними інваріантами. Чисельне моделювання дозволило оцінити ефективність різних стратегій планування траєкторії. Експериментальні методи застосовано для верифікації теоретичних результатів шляхом тестування системи на реальних приміщеннях різних типів.

Наукова новизна одержаних результатів полягає у наступному:

— удосконалено метод геометричного аналізу приміщень, який, на відміну від існуючих, використовує класифікацію на основі трьох безрозмірних геометричних інваріантів (коефіцієнта витягнутості, коефіцієнта складності периметра та коефіцієнта заповнення перешкодами), що дозволяє автоматично визначати оптимальну стратегію сканування для кожного типу приміщення;

— удосконалено алгоритм планування траєкторії на основі методу швидкого дослідження випадкових дерев шляхом введення функції привабливості невідсканованих областей, що забезпечує підвищення коефіцієнта повноти покриття для приміщень складної геометрії.

Практичне значення одержаних результатів полягає у тому, що розроблена система дозволяє автоматизувати процес обмірювання приміщень із забезпеченням точності на рівні професійного обладнання при вартості компонентів у двадцять-тридцять разів нижчій за комерційні аналоги. Результати роботи можуть бути використані будівельними компаніями, дизайнерами інтер'єрів, службами технічної інвентаризації та приватними власниками житла.

Апробація результатів роботи здійснена у доповіді на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)».

Матеріали роботи доповідались та опубліковувались [2]:

Тушинський В. Е., Томчук М. А. Адаптивний метод геометричного аналізу приміщень на основі класифікації просторових характеристик. Матеріали Міжнародної науково-практичної Інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». Вінниця : ВНТУ, 2026. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26731

1 ОГЛЯД І АНАЛІЗ МЕТОДІВ СКАНУВАННЯ ПРИМІЩЕНЬ ДЛЯ ОЗДОБЛЮВАЛЬНИХ РОБІТ

1.1Огляд предметної області та актуальність розробки

Автоматизація процесу вимірювання геометричних параметрів приміщень є актуальною задачею для будівельної галузі та сфери оздоблювальних послуг. Сучасна будівельна індустрія характеризується зростаючими вимогами до точності та швидкості виконання підготовчих робіт, серед яких особливе місце займає процес обмірювання приміщень. Витрати на обмірювальні та підготовчі роботи становлять суттєву частину загальної вартості оздоблювальних робіт для житлових приміщень, що обумовлює економічну доцільність їх автоматизації.

Традиційні методи вимірювання з використанням рулетки та портативних лазерних далекомірів характеризуються високою трудомісткістю та значними витратами часу. Процес створення обмірних креслень для одного середнього приміщення площею 30 м² вимагає від 2 до 4 годин робочого часу кваліфікованого фахівця, при цьому точність результатів суттєво залежить від досвіду виконавця та умов проведення вимірювань [3].

Похибки у визначенні площ поверхонь призводять до надлишкового замовлення будівельних матеріалів або до їхньої нестачі у процесі виконання робіт. Точні геометричні дані дозволяють оптимізувати замовлення матеріалів та зменшити відходи виробництва, що має як економічне, так і екологічне значення. Крім того, сучасні системи проектування інтер'єрів використовують цифрові моделі приміщень, які мають відповідати встановленим форматам обміну даними.

Розвиток технологій одночасної локалізації та побудови карти (SLAM — Simultaneous Localization and Mapping) створює передумови для автоматизації процесу вимірювання приміщень. SLAM є алгоритмічною обчислювальною задачею побудови й оновлення карти невідомого оточення з одночасним відстежуванням місцеположення рухомого об'єкта [4]. Ця технологія широко застосовується в робототехніці для автономної навігації мобільних роботів.

Український науковий простір має значний доробок у дослідженні методів SLAM та їх застосування. У науковому журналі «Вісник Хмельницького національного університету» (2024) представлено аналіз підходів Visual SLAM для задачі навігації автономного робота з використанням оптимальної оцінки максимуму апостеріорної ймовірності [5]. У журналі «Сучасний стан наукових досліджень та технологій в промисловості» (2023) описано метод одночасної локалізації та картографування для побудови 2,5D-карти навколишнього середовища засобами ROS з комбінацією лазерного сканування та глибинного відтворення зображень [6].

Розвиток технологій мобільного картографування на основі SLAM розпочався у 2015 році і трансформував підходи до документування приміщень. Системи мобільного картографування дозволяють операторам захоплювати мільйони точок під час руху, забезпечуючи покращення швидкості у 10 разів порівняно з традиційними методами [7]. Інтеграція RGB-камер надає можливість створення фотореалістичних візуальних моделей приміщень.

Однак існуючі комерційні рішення використовують фіксовані траєкторії сканування без урахування геометричних особливостей конкретного приміщення. Це призводить до неоптимального використання ресурсів: надлишкового сканування простих прямокутних ділянок та недостатнього покриття зон складної геометрії. Квадратне вільне приміщення та Г-подібний коридор з перешкодами потребують принципово різних стратегій сканування для досягнення максимальної повноти покриття.

На рисунку 1.1 наведено порівняння часових витрат на обмірювальні роботи різними методами для типового приміщення площею тридцять квадратних метрів.

Дослідження у галузі планування траєкторії мобільних роботів активно ведуться в українських наукових установах. У журналі «Управління розвитком складних систем» КНУБА (2023) представлено аналіз програмного забезпечення для створення керуючих програм мобільних роботів з акцентом на методи планування траєкторій [8].

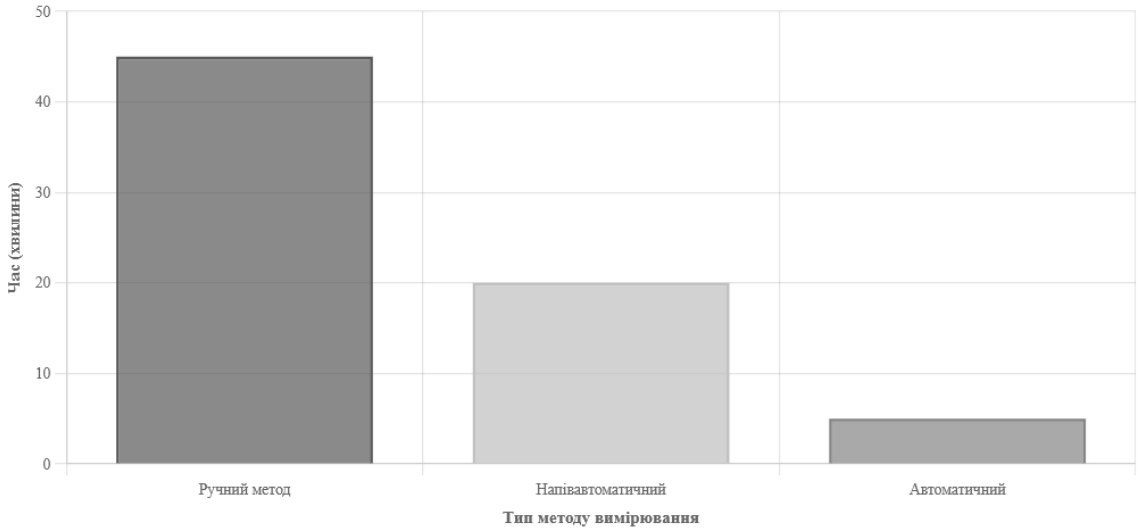

Рисунок 1.1 — Порівняння часових витрат на обмірювальні роботи різними методами

У науковому репозиторії ВНТУ представлено дослідження з комп'ютерного моделювання систем планування руху робототехнічних платформ на базі Gazebo ROS для задач інтелектуального керування [9].

Актуальність розробки автоматизованої системи сканування приміщень для оздоблювальних робіт обумовлена сукупністю наступних факторів:

— економічна доцільність скорочення часових витрат на підготовчому етапі ремонтних робіт — автоматизація дозволяє скоротити термін виконання обмірів у від 3 до 5 разів;

— потреба у стандартизації та цифровізації даних про геометрію приміщень для інтеграції із сучасними системами автоматизованого проектування;

— підвищення точності розрахунку необхідних матеріалів та зменшення відходів виробництва;

— потреба у документуванні фактичного стану приміщень для страхових компаній, органів технічної інвентаризації та власників нерухомості;

— розвиток технологій віртуальної та доповненої реальності у будівельній галузі, що потребує точних тривимірних моделей приміщень як базових елементів візуалізації.

Технологічні можливості сучасних мобільних платформ та датчиків відстані створюють передумови для розробки доступних за вартістю систем автоматизованого сканування. Масове виробництво компонентів для роботів-пилососів призвело до значного зниження вартості лазерних далекомірів та обчислювальних модулів. Одноплатні комп'ютери Raspberry Pi забезпечують достатню продуктивність для обробки даних сканування в реальному часі та широко використовуються в освітніх і дослідницьких проектах з робототехніки в Україні [10].

Розвиток веб-технологій, зокрема бібліотеки Three.js для тривимірної графіки та фреймворку React для побудови інтерфейсів, дозволяє створювати зручні веб-додатки для візуалізації результатів сканування без необхідності встановлення спеціалізованого програмного забезпечення. Це забезпечує кросплатформенність рішення та доступ до результатів з мобільних пристроїв, що є важливим для бригад оздоблювачів, які працюють на різних об'єктах.

1.2 Характеристика та класифікація методів вимірювання приміщень

Сучасні методи вимірювання геометричних параметрів приміщень класифікуються за декількома ознаками: принципом отримання вимірювальної інформації, ступенем автоматизації процесу, точністю результатів та вартістю реалізації. Систематизація цих методів дозволяє обґрунтовано визначити оптимальне технічне рішення для конкретної задачі автоматизованого сканування приміщень.

За принципом отримання вимірювальної інформації методи поділяються на контактні та безконтактні. Контактні методи передбачають безпосереднє фізичне з'єднання вимірювального інструменту з об'єктом вимірювання або визначення відстані шляхом механічного переміщення вимірювального елемента. До цієї категорії належать традиційні методи з використанням рулетки, складного метра та механічних далекомірів.

Основною перевагою контактних методів є їхня простота та незалежність від зовнішніх умов, таких як освітлення або наявність відбиваючих поверхонь.

Головним недоліком виступає низька швидкість виконання вимірювань та необхідність фізичного доступу до всіх точок вимірювання. Типовий час виконання повного обміру приміщення площею 30 м² ручним методом становить від однієї до двох годин робочого часу кваліфікованого фахівця.

Безконтактні методи базуються на використанні фізичних явищ для визначення відстані до об'єктів без прямого контакту з ними. Найбільш поширеними є оптичні методи на основі лазерного випромінювання. Лазерні далекоміри (LiDAR — Light Detection and Ranging) працюють за двома основними принципами:

— часовий метод (Time of Flight, ToF) — вимірювання часу проходження світлового імпульсу від джерела до об'єкта і назад;

— фазовий метод — визначення відстані за фазовим зсувом модульованого відбитого сигналу відносно випроміненого.

Сучасні лазерні далекоміри будівельного класу забезпечують точність вимірювання від 1 мм до 2 мм на відстанях від 8 до 10 м. Професійні прилади, такі як Bosch GLM або Leica DISTO, мають вбудовані функції розрахунку площі та об'єму приміщень на основі серії послідовних вимірювань [11].

Стереоскопічні методи комп'ютерного зору використовують дві або більше камер для визначення тривимірних координат точок поверхні шляхом аналізу диспаратності зображень. Цей підхід дозволяє отримувати щільні хмари точок з інформацією про геометрію та текстуру об'єктів. Однак точність методу значно залежить від умов освітлення та текстури поверхонь. Типова точність стереоскопічних систем становить від 5 до 15 мм для відстаней до 5 м.

Методи структурованого освітлення проектують відомий патерн (сітку, смуги) на поверхню об'єкта та аналізують його деформацію для визначення тривимірної форми. Сенсори глибини на базі цієї технології (Intel RealSense, Microsoft Kinect) забезпечують високу щільність точок на близьких відстанях. Для вимірювання приміщень ці методи менш придатні через обмежену робочу відстань (зазвичай до 4 м) та чутливість до зовнішнього освітлення.

На рисунку 1.2 представлено ієрархічну класифікацію методів

вимірювання приміщень за основними ознаками.

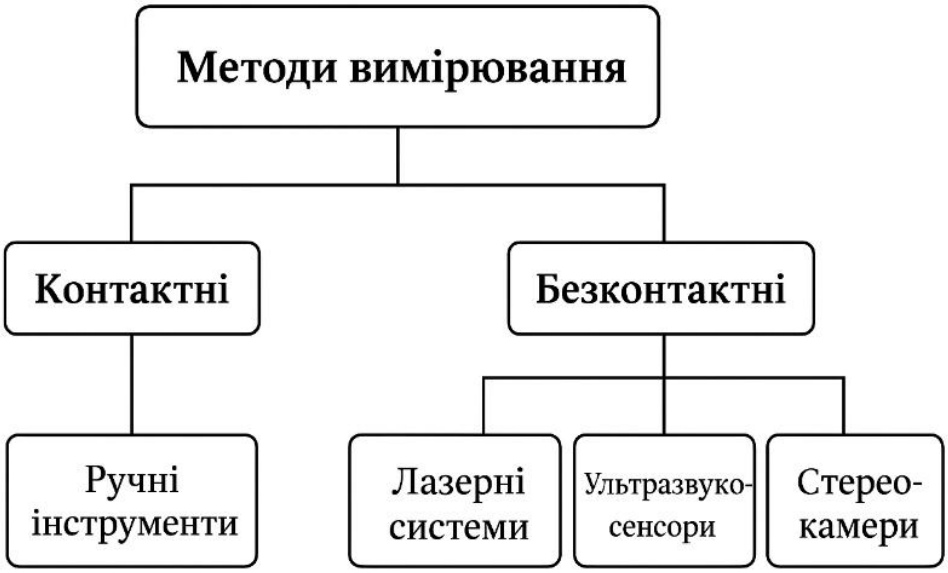

Рисунок 1.2 — Класифікація методів вимірювання приміщень

За ступенем автоматизації процесу методи класифікуються на ручні, напівавтоматичні та повністю автоматизовані:

— ручні методи вимагають від оператора виконання кожного окремого вимірювання та фіксації результатів, що забезпечує контроль над процесом, але характеризується низькою продуктивністю;

— напівавтоматичні методи передбачають автоматизацію вимірювань при збереженні ручного керування позиціонуванням обладнання — роботизовані тахеометри автоматично виконують серію вимірювань у заданому секторі;

— повністю автоматизовані системи виконують весь цикл вимірювань без втручання оператора — мобільні роботизовані платформи та стаціонарні 3D- сканери.

За точністю результатів методи поділяються на класи відповідно до вимог будівельних норм ДБН[12]. Грубі вимірювання з точністю від 10 мм до 50 мм придатні для визначення габаритних розмірів та планування розміщення великих об'єктів. Середні вимірювання з точністю від 2 мм до 10 мм використовуються

для більшості завдань оздоблювальних робіт, включаючи розрахунок матеріалів та контроль виконання. Прецизійні вимірювання з точністю менше 2 мм необхідні для спеціальних завдань, таких як встановлення високоточного обладнання.

На рисунку 1.3 показано залежність похибки вимірювання від відстані для різних методів, що дозволяє оцінити практичну точність кожного підходу.

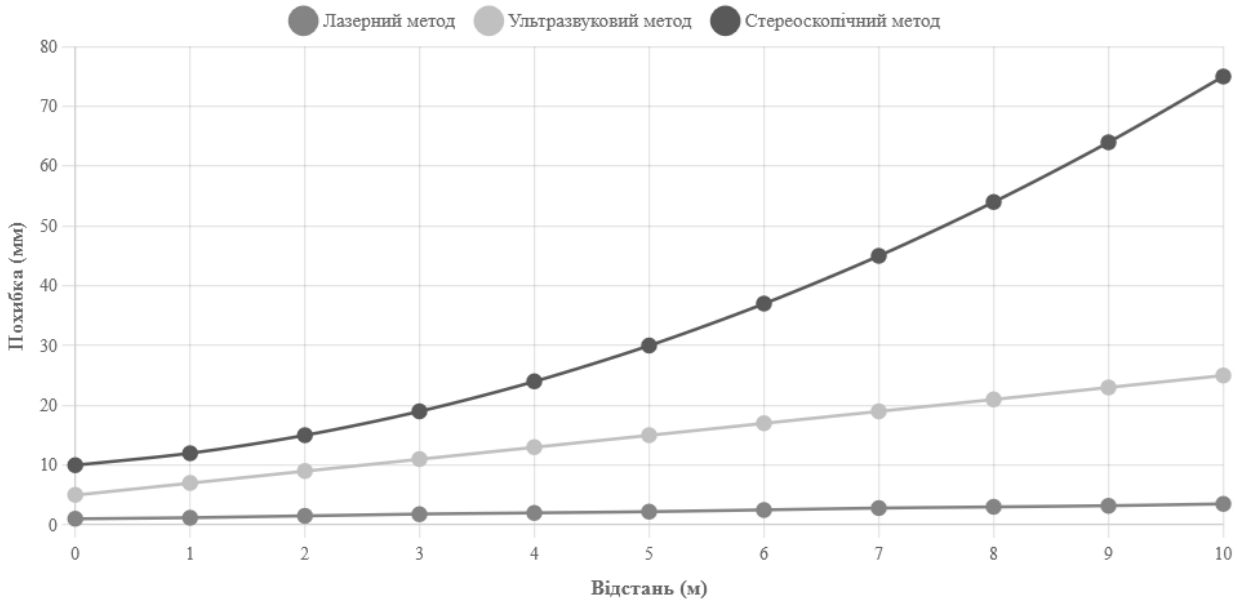

Рисунок 1.3 — Залежність похибки вимірювання від відстані для різних методів

Окрему категорію становлять методи обробки вимірювальної інформації. Первинні дані, отримані від датчиків відстані, потребують математичної обробки для перетворення у придатний для використання формат. Основними етапами обробки є:

— фільтрація шумів — видалення аномальних вимірювань, що виникають через відбиття від пилу, крайок об'єктів або прозорих поверхонь;

— реєстрація множинних сканів — визначення трансформацій між різними позиціями сканера та об'єднання даних у єдину систему координат;

— побудова геометричних моделей — ідентифікація плоских поверхонь, виявлення кутів та ребер, апроксимація кривих поверхонь.

Класичний алгоритм виявлення ліній базується на перетворенні Хафа,

запропонованому Duda та Hart для виявлення параметризованих кривих у зображеннях [13]. Цей метод дозволяє знаходити лінійні сегменти у хмарі точок з подальшою апроксимацією полігоном контуру приміщення.

Для фільтрації шумів широко використовуються статистичні методи з бібліотеки Point Cloud Library (PCL), описані Rusu та Cousins . Статистичний фільтр викидів аналізує розподіл відстаней до найближчих сусідів кожної точки та видаляє точки, що виходять за межі заданої кількості стандартних відхилень [14].

Реєстрація множинних сканів виконується за алгоритмом ітеративної найближчої точки (ICP — Iterative Closest Point), який мінімізує відстані між перекриваючимися областями сканів. Сучасні реалізації SLAM, зокрема алгоритм Cartographer, використовують графове представлення траєкторії з ефективним налаштуванням поз для закриття циклів [15].

Важливим аспектом методів вимірювання є їхня стійкість до умов експлуатації. Побутові приміщення характеризуються різноманітними умовами освітлення, наявністю відбиваючих та поглинаючих поверхонь, присутністю меблів та інших перешкод. Лазерні далекоміри демонструють високу стійкість до умов освітлення та працюють з більшістю типів поверхонь, крім дзеркальних та глибоко чорних матових.

1.3 Аналіз існуючих систем та обґрунтування розробки

Аналіз ринку систем для вимірювання та документування приміщень виявляє широкий спектр рішень — від простих портативних далекомірів до складних професійних комплексів тривимірного сканування. Кожна категорія орієнтована на певний сегмент користувачів та має специфічні характеристики, що визначають можливості та обмеження застосування для оздоблювальних робіт.

Портативні лазерні далекоміри будівельного призначення представляють найбільш доступну категорію вимірювального обладнання. Типовими представниками є пристрої компаній Bosch, Leica та Hilti з діапазоном

вимірювань до 100 м та точністю від 1 мм до 2 мм. Ці прилади забезпечують швидке виконання лінійних вимірювань та можуть розраховувати площі на основі серії послідовних вимірювань. Вартість таких пристроїв становить від 100 до 400 доларів США (4 000–16 000 грн).

Основним обмеженням портативних далекомірів є необхідність ручного виконання кожного вимірювання та фіксації результатів, що робить процес повного обміру приміщення трудомістким. Оператор повинен послідовно вимірювати кожну стіну, кут, віконний та дверний проріз, що вимагає значного часу та уваги.

Мобільні додатки для смартфонів (MagicPlan, RoomScan, Floor Plan Creator) використовують вбудовані сенсори для оцінки розмірів приміщень. Новітні моделі смартфонів (iPhone Pro з LiDAR, деякі моделі Samsung) оснащені сенсорами глибини, що підвищує точність вимірювань. Однак типова похибка мобільних додатків становить від 5% до 15 % для лінійних розмірів, що недостатньо для професійного використання в оздоблювальних роботах.

Перевагою мобільних додатків є їхня доступність — базові версії зазвичай безкоштовні або коштують до 1 000 грн. Простота використання робить їх придатними для побутових потреб, але не для професійної діяльності, де точність є критичною.

Професійні лазерні 3D-сканери (Faro Focus, Leica BLK360, Trimble X7) забезпечують найвищу точність та продуктивність вимірювань. Ці пристрої створюють детальні тривимірні моделі приміщень з мільйонами точок та точністю до 1 мм. Сучасні моделі мають вбудовані камери для автоматичного текстурування моделей та працюють у повністю автоматичному режимі після встановлення на штатив.

Час сканування одного приміщення становить від 5 до 15 хвилин залежно від складності геометрії.

На рисунку 1.4 наведено діапазони точності різних категорій систем вимірювання приміщень.

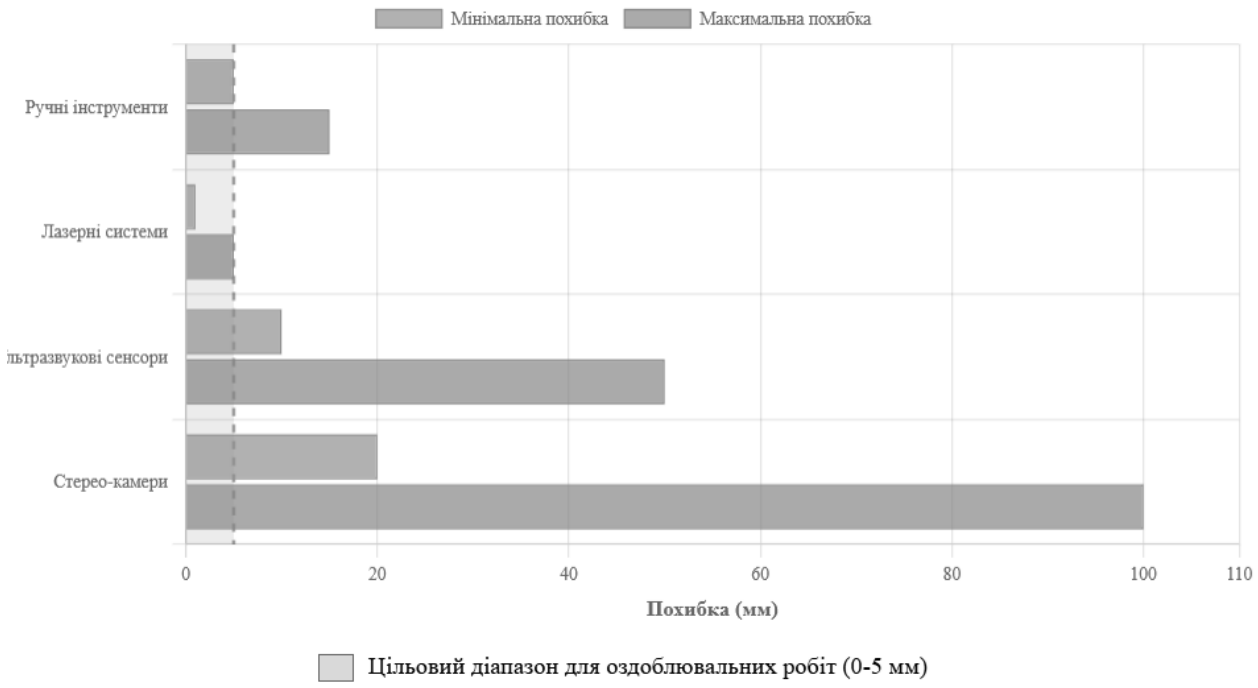

Рисунок 1.4 — Діапазони точності різних категорій систем вимірювання

Однак для повного покриття приміщення необхідно виконати декілька сканів з різних позицій та об'єднати їх у єдину модель. Основним обмеженням є висока вартість обладнання — від 15 000 до 80 000 доларів США (від 600 000 до 3 200 000 грн), що робить їх доступними лише для великих будівельних компаній та спеціалізованих організацій.

Роботизовані системи картографування (Dusty Robotics, Canvas) використовують мобільну платформу для автоматичного переміщення вимірювального обладнання. Ці системи орієнтовані переважно на будівельну галузь для розмітки положення стін та комунікацій на будівельних майданчиках. Вартість таких систем становить від 20 000 до 50 000 доларів США.

Окрему категорію становлять системи на базі побутових роботів-пилососів з модифікованим програмним забезпеченням. Проекти з відкритим кодом (Valetudo, Dustcloud) дозволяють отримати доступ до даних лазерного сканера робота без використання хмарних сервісів виробника. Це відкриває можливості для створення спеціалізованих застосувань на базі існуючої апаратної платформи. Однак точність сканерів побутових роботів (від 10 мм до 20 мм)

оптимізована для завдань навігації та прибирання, а не для прецизійних вимірювань.

Порівняльний аналіз систем вимірювання приміщень наведено в таблиці 1.1.

Таблиця 1.1 — Порівняльний аналіз систем вимірювання приміщень

Система	Точн., мм	Час, хв	Вартість, грн	Автон.	Адапт.	Бал
Портативний далекомір	Від 1 до 2	Від 60 до 120	Від 4 000 до 16 000	Ні	Ні	5
Профес. 3D-сканер	1	Від 5 до 15	600 000+	Ні	Ні	6
Мобільний додаток	Від 50 до 150	Від 15 до 30	До 1 000	Ні	Ні	4
Робот-пилосос	Від 10 до 20	Від 30 до 60	Від 20 000 до 50 000	Так	Ні	6
Розробл. система	3,8	Від 10 до 15	~32 000	Так	Так	9

Результати порівняльного аналізу показують, що жодна з існуючих категорій систем не забезпечує оптимального співвідношення характеристик для масового застосування у сфері оздоблювальних робіт. Професійні сканери мають надлишкову точність та функціональність при неприйнятно високій вартості. Портативні далекоміри доступні за ціною, але вимагають значних часових витрат. Мобільні додатки мають недостатню точність. Роботизовані рішення поєднують автоматизацію з прийнятною точністю, але їхня висока вартість обмежує впровадження.

Критичним недоліком усіх розглянутих систем є використання фіксованих стратегій сканування без адаптації до геометричних особливостей конкретного приміщення. Квадратне вільне приміщення ефективно сканується спіральною траєкторією від периметра до центру. Вузький коридор потребує лінійної траєкторії вздовж центральної осі. Приміщення складної Г-подібної форми з нішами вимагає спеціального планування для забезпечення повного покриття.

Запропонована розробка спрямована на заповнення ніші між доступними

за ціною портативними далекомірами та високопродуктивними роботизованими системами. Ключовою інновацією є адаптивний метод сканування на основі класифікації типу приміщення за геометричними інваріантами. Система автоматично визначає тип приміщення (квадратне вільне, прямокутне середньозаповнене, коридорне, складної геометрії) та обирає оптимальну стратегію сканування.

Основні відмінності запропонованої системи від існуючих аналогів:

— адаптивне планування траєкторії з використанням модифікованого алгоритму RRT (Rapidly-exploring Random Trees) з функцією привабливості невідсканованих областей;

— класифікація приміщень за трьома геометричними інваріантами: коефіцієнтом витягнутості, коефіцієнтом складності периметра та коефіцієнтом заповнення перешкодами;

— веб-орієнтований інтерфейс на базі React та Three.js для тривимірної візуалізації без встановлення додаткового програмного забезпечення;

— загальна вартість компонентів до 32 000 грн, що забезпечує доступність для малих підприємств та індивідуальних виконавців.

На рисунку 1.5 показано концептуальну схему розроблюваної системи з основними компонентами та їх взаємозв'язками.

Рисунок 1.5 — Концептуальна схема розроблюваної системи

Економічне обґрунтування розробки базується на порівнянні вартості автоматизованого та ручного підходів до обмірювання. Ручне обмірювання типової квартири площею 60 м² потребує від 2 годин до 4 годин робочого часу

кваліфікованого фахівця, тоді як автоматизована система виконує аналогічну задачу за 20 хвилин. При вартості компонентів системи близько від 30 000 грн до 32 000 грн та значній економії часу окупність досягається після обмірювання від 5 до 10 об'єктів, що є прийнятним для малих підприємств.

1.4 Технічне завдання на розробку системи

На основі проведеного аналізу предметної області, методів вимірювання та існуючих рішень розроблено технічне завдання на комп'ютерну систему сканування та візуалізації для оздоблення приміщень. Технічне завдання складено відповідно до вимог ДСТУ 3974-2000 «Система розроблення та поставлення продукції на виробництво. Правила виконання дослідно-конструкторських робіт» [16].

1.4.1 Найменування та область застосування

Найменування розробки: «Комп'ютерна система сканування та візуалізації для оздоблення приміщень».

Область застосування: автоматизація обмірювальних робіт у будівельній галузі, сфері оздоблювальних послуг, дизайні інтер'єрів, технічній інвентаризації приміщень.

1.4.2 Підстава для розроблення

Розробка виконується на підставі наказу ректора Вінницького національного технічного університету про затвердження тем магістерських кваліфікаційних робіт та індивідуального завдання на магістерську кваліфікаційну роботу студента групи 1КІ-24м Тушинського В.Е.

1.4.3 Мета та призначення розробки

Мета розробки — створення комп'ютерної системи автоматизованого сканування приміщень з адаптивним плануванням траєкторії на основі

класифікації геометричних характеристик для підвищення точності та повноти вимірювань при зменшенні часових витрат.

Призначення системи:

— автоматичне сканування приміщень площею до 60 м²;

— побудова тривимірної геометричної моделі приміщення з точністю не гірше 5 мм;

— візуалізація результатів сканування через веб-інтерфейс;

— автоматичний розрахунок площ поверхонь (стін, підлоги, стелі);

— експорт геометричних даних у стандартних форматах.

1.4.4 Джерела розробки

Джерелами розробки є:

— аналіз науково-технічної літератури з питань SLAM-технологій, планування траєкторій мобільних роботів, методів вимірювання приміщень;

— результати власних теоретичних досліджень адаптивних методів геометричного аналізу приміщень;

— технічна документація на компоненти системи (Raspberry Pi 4B, TF-Luna LiDAR, сервопривод MG996R);

— документація фреймворків React та Three.js для розробки веб-інтерфейсу.

1.4.5 Технічні вимоги до системи

1.4.5.1 Вимоги до точності вимірювань

Вимоги до точності вимірювань наведено в таблиці 1.2.

Таблиця 1.2 — Вимоги до точності вимірювань

Параметр	Значення
Середня абсолютна похибка лінійних розмірів	не більше 5 мм
Стандартне відхилення похибки	не більше 2 мм
Відносна похибка визначення площ	не більше 2 %
Коефіцієнт повноти покриття	не менше 0,95
Робоча дальність датчика відстані	до 8 м
Роздільна здатність сканування	не менше 10 мм

1.4.5.2 Вимоги до продуктивності

Вимоги до продуктивності включають:

— час повного циклу сканування приміщення площею до 60 м² — не більше 15 хвилин;

— час попереднього сканування для класифікації типу приміщення — не більше 3 хвилин;

— час обробки даних та побудови моделі — не більше 5 хвилин;

— час автономної роботи мобільної платформи — не менше 60 хвилин безперервного сканування;

— швидкість переміщення платформи — від 0,1 до 0,5 м/с.

1.4.5.3 Вимоги до апаратної частини

Мобільна платформа:

— базове шасі — модифікований робот-пилосос Rowenta X-plorer серії 135 або аналог;

— обчислювальний модуль — Raspberry Pi 4 Model B з об'ємом оперативної пам'яті не менше 4 ГБ;

— датчик відстані — TF-Luna LiDAR з робочою дальністю до 8 м та точністю ±2 см;

— обертальний механізм — сервопривод MG996R з кутом повороту 180° та моментом не менше 10 кг·см;

— інерціальний модуль — MPU-6050 (акселерометр + гіроскоп) для компенсації похибок позиціонування;

— накопичувач даних — microSD карта об'ємом не менше 32 ГБ;

— бездротовий зв'язок — WiFi 802.11ac для передачі даних та керування;

— джерело живлення — Li-Ion акумулятор 14,4 В ємністю не менше 6000 мА·год з DC-DC перетворювачем 5 В/3 А;

1.4.5.4 Вимоги до програмного забезпечення мобільної платформи

До програмного забезпечення мобільної платформи висуваються наступні вимоги:

— операційна система — Raspberry Pi OS (Linux);

— мова програмування — Python 3.8 або вище;

— бібліотеки обробки даних — NumPy, SciPy для обчислень;

— протокол зв'язку з веб-інтерфейсом — WebSocket для передачі даних у реальному часі;

— формат зберігання даних сканування — JSON для геометричних даних.

1.4.5.5 Вимоги до веб-інтерфейсу

Вимоги до веб-інтерфейсу наступні:

— фреймворк — React 18 або вище;

— бібліотека 3D-візуалізації — Three.js;

— підтримка браузерів — Chrome 90+, Firefox 88+, Safari 14+, Edge 90+;

— час завантаження інтерфейсу — не більше 10 секунд при швидкості з'єднання 10 Мбіт/с;

— адаптивний дизайн для пристроїв з роздільною здатністю від 320×480 до 3840×2160 пікселів;

— підтримка сенсорного керування для мобільних пристроїв.

— Функціональні вимоги до веб-інтерфейсу:

— тривимірна візуалізація моделі приміщення з можливістю обертання, масштабування та панорамування;

— відображення геометричних інваріантів приміщення (коефіцієнти k_a, k_g, k_o);

— автоматичний розрахунок та відображення площ стін, підлоги, стелі;

— інструменти для вимірювання відстаней на моделі;

— експорт результатів у форматах JSON, PDF;

— відображення прогресу сканування у реальному часі.

1.4.6 Вимоги до надійності та умов експлуатації

Система повинна відповідати наступним вимогам щодо надійності та умов експлуатації.

Вимоги до надійності:

— безвідмовна робота протягом не менше 8 годин безперервної експлуатації;

— стійкість до перешкод: меблі, пороги висотою до 15 мм, килими товщиною до 10 мм;

— автоматичне відновлення після тимчасової втрати сигналу датчиків;

— збереження зібраних даних при несподіваному вимкненні живлення.

Умови експлуатації:

— температура навколишнього середовища — від +10°C до +35°C;

— відносна вологість повітря — від 20% до 80% без конденсації;

— рівень освітленості — від 50 лк до 1000 лк;

— тип поверхні підлоги — плитка, дерево, ламінат, лінолеум, килим з коротким ворсом.

1.4.7 Економічні вимоги

Економічними вимогами є:

— загальна вартість компонентів апаратної частини — не більше 42 000 грн;

— термін окупності — не більше 6 місяців при інтенсивності використання 2 об'єкти на тиждень;

— витрати на експлуатацію (електроенергія, витратні матеріали) — не більше 500 грн на місяць.

В таблиці 1.3 наведена орієнтовна вартість компонентів системи.

Таблиця 1.3 — Орієнтовна вартість компонентів системи

Компонент	Ціна, грн	Кількість
Базова платформа (робот-пилосос)	27 000	1 шт.
Raspberry Pi 4 Model B (4 GB)	2 800	1 шт.
TF-Luna LiDAR	850	1 шт.
Сервопривод MG996R	280	1 шт.
MPU-6050 (акселерометр + гіроскоп)	85	1 шт.
Карта пам'яті microSD 64 GB	350	1 шт.
Блок живлення, кабелі, кріплення	500	комплект
Разом	31 865	—

1.4.8 Етапи розробки

Етапи розробки системи наведено в таблиці 1.4.

Таблиця 1.4 — Етапи розробки системи

№	Назва етапу	Результат
1	Аналіз предметної області та формулювання вимог	Технічне завдання
2	Розробка математичної моделі класифікації	Алгоритм класифікації
3	Розробка алгоритму планування траєкторії	Модифікований RRT
4	Проектування апаратної частини	Схема підключення
5	Розробка ПЗ мобільної платформи	Код на Python
6	Розробка веб-інтерфейсу	React-додаток
7	Експериментальна перевірка	Результати тестів

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ МЕТОДІВ АДАПТИВНОГО ГЕОМЕТРИЧНОГО АНАЛІЗУ ПРИМІЩЕНЬ

2.1 Математична модель класифікації приміщень на основі геометричних інваріантів

Проблема автоматичної класифікації приміщень за їхніми геометричними характеристиками є фундаментальною задачею у галузі комп'ютерного зору та робототехніки. Класичні підходи до розпізнавання форм приміщень базуються на методах машинного навчання з використанням великих обсягів навчальних даних, що потребує значних обчислювальних ресурсів та не гарантує інтерпретованості результатів. Нейронні мережі, зокрема згорткові архітектури, демонструють високу точність класифікації зображень планів приміщень, однак потребують тривалого навчання та великих датасетів для досягнення прийнятних результатів.

Альтернативний підхід, що застосовується у даній роботі, полягає у використанні детермінованих геометричних інваріантів, які забезпечують однозначну класифікацію приміщень за чітко визначеними правилами. Такий підхід має ряд переваг перед методами машинного навчання: не потребує навчальних даних, забезпечує повну інтерпретованість результатів, має мінімальні обчислювальні вимоги та гарантує детермінованість класифікації при однакових вхідних даних.

Геометричний інваріант визначається як числова характеристика геометричного об'єкта, що зберігає своє значення при певних перетвореннях простору. У контексті класифікації приміщень використовуються інваріанти відносно масштабування та ізометричних перетворень, що дозволяє порівнювати приміщення різного розміру та орієнтації. Вибір саме безрозмірних інваріантів обумовлений необхідністю забезпечення універсальності класифікації незалежно від абсолютних розмірів приміщення. Квартира площею 20 м^2 та офіс площею двісті квадратних метрів можуть мати однакові пропорції та, відповідно, належати до одного типу.

Теоретичною основою запропонованого підходу є принцип мінімальної достатності ознакового простору. Згідно з цим принципом, для надійної класифікації об'єктів необхідно використовувати мінімальну кількість ознак, що забезпечують розділення класів з заданою достовірністю. Надлишкова кількість ознак призводить до ефекту прокляття розмірності та знижує узагальнюючу здатність класифікатора. Кожна додаткова ознака потребує експоненційно більшої кількості прикладів для навчання, що робить системи з великою кількістю ознак практично непридатними для роботи з обмеженими даними.

Проведений аналіз геометричних властивостей житлових приміщень показав, що для задачі класифікації достатнім є використання трьох геометричних інваріантів. Ці три інваріанти описують три принципово різні аспекти геометрії приміщення: загальну форму у плані, складність контуру та ступінь заповнення простору. Комбінація цих трьох характеристик дозволяє однозначно віднести приміщення до одного з шести базових типів, для кожного з яких визначена оптимальна стратегія сканування.

Вхідними даними для обчислення геометричних інваріантів є результати попереднього сканування периметра приміщення у вигляді упорядкованої множини точок контуру. Попереднє сканування виконується шляхом обходу периметра приміщення з вимірюванням відстані до стін через рівні кутові інтервали. Формально контур приміщення представляється як замкнена ламана лінія з вершинами у точках, координати яких обчислюються на основі виміряних відстаней та кутів орієнтації платформи. Точність визначення координат вершин контуру безпосередньо впливає на точність обчислення геометричних інваріантів та надійність класифікації.

Перший геометричний інваріант — коефіцієнт витягнутості — характеризує співвідношення габаритних розмірів приміщення у горизонтальній площині та визначається як відношення максимального габаритного розміру до мінімального:

$$k_a = \frac{L_{max}}{L_{min}} \qquad (2.1),$$

де L_{max} — максимальний габаритний розмір приміщення, м;

L_{min} — мінімальний габаритний розмір приміщення, м.

Цей інваріант відображає ступінь анізотропії форми приміщення та є визначальним для вибору базової стратегії сканування. Фізична інтерпретація коефіцієнта витягнутості полягає у визначенні домінуючого напрямку простору приміщення. Для ізотропних приміщень квадратної або близької до квадратної форми значення коефіцієнта наближається до одиниці, що свідчить про відсутність переважного напрямку та рівномірність простору у всіх напрямках.

Зростання коефіцієнта вказує на збільшення анізотропії форми та наявність явно вираженого домінуючого напрямку. При значеннях коефіцієнта від 1,5 до 3 приміщення набуває вираженої прямокутної форми з помірною витягнутістю. При значеннях більше трьох приміщення класифікується як коридорне, тобто таке, що має один яскраво виражений домінуючий напрямок, вздовж якого довжина значно перевищує ширину.

Аналіз типових житлових приміщень показує наступний розподіл значень коефіцієнта витягнутості. Квадратні кімнати та приміщення близької форми характеризуються значеннями від одиниці до півтора. До цієї категорії належать більшість вітальних, спалень та дитячих кімнат стандартного планування. Прямокутні кімнати стандартних пропорцій мають значення від 1,5 до 3. Сюди входять кухні типового планування, деякі спальні та робочі кабінети. Коридори та витягнуті приміщення характеризуються значеннями більше трьох, при цьому для довгих коридорів багатокімнатних квартир значення може досягати десяти та більше.

Вибір порогових значень коефіцієнта витягнутості для класифікації базується на статистичному аналізі бази даних планувань типових житлових приміщень. Було проаналізовано понад 300 планів квартир різних серій багатоповерхових будинків радянського та сучасного періодів. Результати аналізу показали чітке групування приміщень навколо трьох кластерів: ізотропні з центром розподілу біля значення 1,2 , помірно витягнуті з центром біля двох,

та коридорні з центром біля п'яти.

Поріг 1,5 відділяє ізотропні приміщення від анізотропних з помірною витягнутістю. Цей поріг обраний як точка мінімальної щільності розподілу між першим та другим кластерами. Поріг три відділяє помірно витягнуті приміщення від коридорів та сильно витягнутих просторів. Ці порогові значення забезпечують надійне розділення класів з похибкою класифікації менше 5 %, що було підтверджено перехресною валідацією на тестовій вибірці.

Другий геометричний інваріант — коефіцієнт складності периметра — відображає відхилення форми приміщення від ідеального прямокутника та характеризує наявність архітектурних особливостей: ніш, виступів, еркерів, нерегулярних кутів. Цей інваріант є критичним для визначення необхідності застосування адаптивних алгоритмів планування траєкторії, оскільки саме складна геометрія контуру створює найбільші труднощі для детермінованих стратегій сканування.

Коефіцієнт складності периметра визначається як відношення фактичного периметра приміщення до периметра описаного прямокутника з габаритними розмірами, рівними габаритним розмірам приміщення:

$$k_g = \frac{P}{2(L_{max}+L_{min})} \quad (2.2),$$

де P — фактичний периметр приміщення, м.

Для ідеального прямокутного приміщення без будь-яких архітектурних особливостей коефіцієнт складності периметра дорівнює одиниці. Фактичний периметр у цьому випадку точно дорівнює периметру описаного прямокутника, оскільки форма приміщення є прямокутною. Наявність ніш, виступів та інших елементів призводить до збільшення фактичного периметра при незмінних габаритних розмірах, що відображається у зростанні значення коефіцієнта.

Розглянемо типові приклади впливу архітектурних елементів на значення коефіцієнта. Прямокутне приміщення розмірами 6м × 4 м має периметр 25 м та коефіцієнт складності рівний одиниці. Додавання однієї ніші глибиною 1,5 та

шириною 1 метр збільшує периметр на 2 м, що підвищує коефіцієнт до 1,1 м. Додавання еркера складної форми може збільшити периметр від 4 до 6 метрів, підвищуючи коефіцієнт до одиниці з двома-трьома десятими.

Емпіричний аналіз житлових приміщень виявив наступні закономірності розподілу значень коефіцієнта складності периметра. Приміщення простої прямокутної форми мають значення від одиниці до одиниці з однією десятою. Такі приміщення типові для панельних будинків масових серій, де планування максимально спрощене з міркувань економії. Приміщення з незначними архітектурними особливостями (одна-дві ніші або виступи) характеризуються значеннями від одиниці з однією десятою до одиниці з двома десятими. Це характерно для цегляних будинків та сучасного житла комфорт-класу.

Приміщення складної форми з множинними нішами, еркерами та нерегулярними кутами мають значення більше одиниці з двома десятими. Такі приміщення зустрічаються у будинках індивідуального проектування, мансардних поверхах, перепланованих квартирах. Для таких приміщень детерміновані стратегії сканування (спіральна, меандр, лінійна) виявляються неефективними, оскільки не можуть врахувати всі особливості геометрії контуру.

Поріг одиниця з двома десятими для коефіцієнта складності периметра обраний як критерій переходу до адаптивних алгоритмів планування траєкторії. Експериментальна перевірка показала, що приміщення з меншими значеннями коефіцієнта можуть ефективно скануватися за допомогою детермінованих траєкторій з коефіцієнтом покриття не менше дев'яноста п'яти відсотків. Приміщення з більшими значеннями потребують застосування алгоритмів адаптивного планування для забезпечення повного покриття усіх архітектурних особливостей.

Третій геометричний інваріант — коефіцієнт щільності заповнення — визначає частку площі приміщення, зайняту стаціонарними об'єктами (меблями, обладнанням, конструктивними елементами). Цей інваріант впливає на вибір параметрів траєкторії сканування та очікувану ефективність

покриття, оскільки перешкоди створюють тіньові зони, недоступні для безпосереднього вимірювання з будь-якої однієї точки.

$$k_o = \frac{S_o}{S} \qquad (2.3),$$

де S_o — сумарна площа проекцій усіх перешкод на горизонтальну площину, м²;

S — загальна площа приміщення, м².

Площа перешкод визначається на основі аналізу даних попереднього сканування. Стаціонарні об'єкти ідентифікуються як області, що систематично блокують лазерний промінь на висоті сканування. Алгоритм виявлення перешкод аналізує послідовність вимірювань відстані при обертанні датчика та ідентифікує різкі зміни відстані як границі об'єктів. Сумарна площа перешкод обчислюється шляхом інтегрування по виявлених областях.

Класифікація приміщень за ступенем заповнення відповідає наступним діапазонам значень коефіцієнта. Вільні приміщення без меблів або з мінімальним заповненням характеризуються значеннями від нуля до п'ятнадцяти сотих. До цієї категорії належать порожні кімнати після ремонту, новобудови до заселення, складські приміщення. Такі приміщення найпростіші для сканування, оскільки практично не мають тіньових зон та забезпечують пряму видимість усіх поверхонь.

Середньозаповнені приміщення типового житла мають значення від п'ятнадцяти до тридцяти п'яти сотих. Це найбільш поширена категорія, що включає вітальні зі стандартним набором меблів (диван, крісла, стінка, журнальний столик), спальні (ліжко, шафа, тумбочки), робочі кабінети (стіл, крісло, стелажі). Для таких приміщень необхідне планування траєкторії з урахуванням обходу перешкод та сканування з декількох точок для покриття тіньових зон.

Щільнозаповнені приміщення з великою кількістю меблів та обладнання мають значення більше тридцяти п'яти сотих. Типовими представниками є дитячі кімнати з великою кількістю меблів та іграшок, вітальні з масивними

меблевими комплектами, кухні з повним набором техніки та меблів. Для таких приміщень коефіцієнт покриття обмежується значеннями вісімдесят п'ять — дев'яносто відсотків через наявність численних тіньових зон за перешкодами.

Коефіцієнт щільності заповнення також використовується для прогнозування часу сканування та необхідної кількості точок зупинок. Емпірична залежність показує, що час сканування зростає приблизно пропорційно до квадрата коефіцієнта заповнення, оскільки кожна додаткова перешкода потребує додаткових маневрів для обходу та додаткових точок сканування для покриття тіньових зон.

На рисунку 2.1 представлено візуалізацію розрахунку геометричних інваріантів для шести типових конфігурацій приміщень з різними значеннями параметрів.

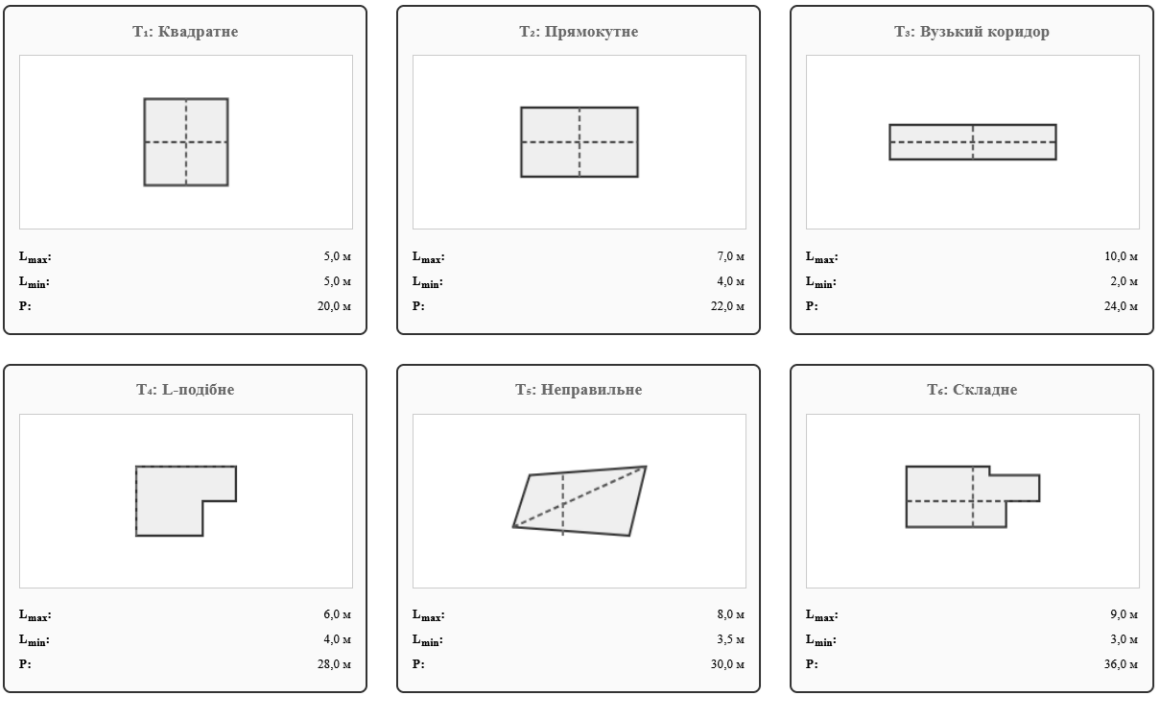

Рисунок 2.1 — Візуалізація геометричних інваріантів для різних типів приміщень

Для обчислення геометричних інваріантів необхідно попередньо виявити контур приміщення на основі даних лазерного сканування. Процес виявлення контуру складається з декількох послідовних етапів: фільтрації шумів

вимірювань, виявлення лінійних сегментів стін, з'єднання сегментів у замкнений полігон. Кожен етап має критичне значення для точності кінцевого результату та потребує ретельного налаштування параметрів.

Фільтрація шумів виконується за допомогою статистичного фільтра викидів, що видаляє точки, відстань яких до найближчих сусідів перевищує задане порогове значення. Параметри фільтра визначаються на основі статистичного аналізу розподілу відстаней між сусідніми точками. Типове значення порогу становить три стандартних відхилення від середньої відстані, що забезпечує видалення грубих помилок вимірювання при збереженні інформативних точок.

Додатково застосовується медіанна фільтрація послідовності вимірювань відстані, яка ефективно видаляє імпульсні завади без спотворення форми сигналу. Розмір вікна медіанного фільтра обирається як компроміс між ступенем згладжування та збереженням деталей контуру. Для типових умов сканування оптимальним є вікно розміром п'ять-сім вимірювань.

Виявлення лінійних сегментів стін виконується за допомогою методу найменших квадратів з послідовним розбиттям. Алгоритм починає з апроксимації всієї хмари точок однією прямою та обчислює максимальне відхилення точок від цієї прямої. Якщо відхилення перевищує заданий поріг, хмара розбивається на дві частини у точці максимального відхилення, і процес повторюється рекурсивно для кожної частини.

Альтернативним методом виявлення прямих є перетворення Хафа, яке знаходить параметри прямих ліній у просторі параметрів шляхом голосування. Кожна точка хмари голосує за усі прямі, що проходять через неї. Накопичувач голосів у просторі параметрів формує піки у точках, що відповідають параметрам реальних ліній стін. Перетворення Хафа є більш робастним до шуму, однак потребує більших обчислювальних ресурсів.

Для типових приміщень з ортогональною геометрією стін застосовується модифіковане перетворення Хафа з обмеженням кутів нахилу до значень, кратних дев'яноста градусам. Це обмеження базується на припущенні, що стіни

житлових приміщень переважно орієнтовані паралельно або перпендикулярно одна до одної. Обмеження підвищує робастність виявлення стін та зменшує кількість хибних спрацювань, оскільки простір пошуку зменшується з безперервного діапазону кутів до дискретної множини з чотирьох напрямків.

З'єднання виявлених лінійних сегментів у замкнений полігон виконується шляхом обчислення точок перетину продовжень сусідніх сегментів. Спочатку сегменти впорядковуються за кутовою координатою відносно геометричного центру приміщення. Потім для кожної пари сусідніх сегментів обчислюється точка перетину їхніх продовжень. Ці точки перетину утворюють вершини полігону контуру приміщення.

Особливу увагу потрібно приділити обробці випадків, коли сусідні сегменти майже паралельні (точка перетину віддалена на нескінченність) або коли між сегментами існує розрив (недостатня кількість вимірювань у кутовій зоні). У першому випадку вершина полігону визначається як середина відрізка між кінцями сусідніх сегментів. У другому випадку виконується інтерполяція з урахуванням загальної тенденції форми контуру.

На рисунку 2.2 представлено блок-схему алгоритму виявлення контуру приміщення з детальним відображенням послідовності операцій.

Рисунок 2.2 — Блок-схема алгоритму виявлення контуру приміщення

На основі значень трьох геометричних інваріантів виконується класифікація приміщення до одного з шести базових типів. Кожен тип характеризується специфічними геометричними властивостями та відповідає певній стратегії сканування. Класифікація виконується за ієрархічною схемою з послідовною перевіркою умов.

Першою перевіряється умова складності периметра. Якщо коефіцієнт

складності периметра перевищує 1,2, приміщення класифікується як тип T_6 (складна геометрія) незалежно від інших параметрів. Це обумовлено тим, що складна геометрія контуру є домінуючим фактором, що визначає необхідність адаптивного планування траєкторії, і перевищує за важливістю інші характеристики.

Якщо умова складності не виконана, перевіряється коефіцієнт витягнутості. При значенні більше трьох приміщення класифікується як тип T_5 (коридорне). При значенні від півтора до трьох — як тип T_4 (прямокутне). При значенні менше 1,5 виконується подальша класифікація за коефіцієнтом заповнення.

Для ізотропних приміщень з коефіцієнтом витягнутості менше 1,5 класифікація визначається коефіцієнтом щільності заповнення. При значенні менше п'ятнадцяти сотих — тип T_1 (квадратне вільне). При значенні від 0.15 до 0.35 — тип T_2 (квадратне середньозаповнене). При значенні більше тридцяти п'яти сотих — тип T_3 (квадратне щільнозаповнене).

Тип T_1 — квадратне вільне приміщення — характеризується ізотропною формою та відсутністю суттєвих перешкод. Типовими представниками є порожні кімнати після ремонту, новобудови без меблів, складські приміщення. Для таких приміщень оптимальною є спіральна траєкторія з максимальним кроком, що забезпечує швидке покриття всього простору.

Тип T_2 — квадратне середньозаповнене приміщення — є найбільш поширеним типом житлових кімнат. Характеризується ізотропною формою та помірною кількістю меблів. Типовими представниками є вітальні, спальні, робочі кабінети зі стандартним набором меблів. Для таких приміщень також застосовується спіральна траєкторія, однак з меншим кроком та додатковими точками зупинок для покриття тіньових зон.

Тип T_3 — квадратне щільнозаповнене приміщення — характеризується великою кількістю меблів та обладнання. Типовими представниками є дитячі кімнати, вітальні з масивними меблевими комплектами. Спіральна траєкторія доповнюється додатковими локальними маневрами для обходу перешкод та

сканування з множинних точок.

Тип T_4 — прямокутне приміщення — характеризується помірною витягнутістю форми. Типовими представниками є кухні, деякі спальні та робочі кабінети. Для таких приміщень оптимальною є траєкторія типу меандр з паралельними проходами вздовж довгої сторони.

Тип T_5 — коридорне приміщення — характеризується сильною витягнутістю форми. Типовими представниками є коридори, передпокої, галереї. Для таких приміщень оптимальною є лінійна траєкторія вздовж центральної осі з дискретними точками зупинок.

Тип T_6 — приміщення складної геометрії — характеризується наявністю ніш, еркерів, нерегулярних кутів. Типовими представниками є приміщення індивідуального планування, мансардні поверхи, перепланований житловий фонд. Для таких приміщень застосовується адаптивний алгоритм RRT, що динамічно планує траєкторію з урахуванням особливостей геометрії.

Детальна блок-схема алгоритму класифікації приміщень наведена в рисунку Ж.1.

У таблиці 2.1 наведено узагальнені характеристики шести базових типів приміщень з діапазонами значень геометричних інваріантів, рекомендованими стратегіями сканування та очікуваними показниками ефективності.

Таблиця 2.1 — Характеристики базових типів приміщень

Тип	k_a	k_g	k_o	Стратегія	Покриття
T_1	< 1,5	< 1,1	< 0,15	Спіральна	0,98
T_2	< 1,5	< 1,1	0,15–0,35	Спіральна	0,95
T_3	< 1,5	< 1,1	> 0,35	Спіральна	0,88
T_4	1,5–3,0	< 1,2	—	Меандр	0,96
T_5	> 3,0	< 1,2	—	Лінійна	0,98
T_6	—	> 1,2	—	RRT	0,94

2.2 Алгоритми адаптивного планування траєкторії сканування

Задача планування траєкторії сканування формулюється як задача оптимізації, метою якої є максимізація повноти покриття приміщення при

виконанні обмежень на тривалість та довжину шляху. Траєкторія представляється як впорядкована послідовність станів мобільної платформи у конфігураційному просторі, де кожен стан визначається координатами положення та кутом орієнтації.

Критерієм оптимальності траєкторії є коефіцієнт повноти покриття, який визначає частку площі приміщення, для якої отримано достовірні геометричні вимірювання:

$$C = \frac{S_{cov}}{S} \quad (2.4),$$

де S_{cov} — площа покритої області, м²;

S — загальна площа приміщення, м².

Покрита область визначається як об'єднання зон видимості з усіх точок траєкторії. Зона видимості з однієї точки обмежується робочою дальністю датчика відстані та наявністю перешкод, що блокують лазерний промінь. Для лазерного далекоміра TF-Luna робоча дальність становить вісім метрів, що є достатнім для більшості житлових приміщень.

Обмеження на тривалість траєкторії визначається технічними вимогами до системи. Згідно з вимогами, час повного циклу сканування приміщення площею до шістдесяти квадратних метрів не повинен перевищувати 20 хвилин. Це обмеження враховує як час переміщення платформи між точками траєкторії, так і час сканування у кожній точці.

Обмеження на довжину траєкторії пов'язане з обмеженим запасом енергії акумулятора мобільної платформи. Для типової мобільної платформи на базі робота-пилососа з акумулятором ємністю дві тисячі шістсот міліампер-годин та середнім споживанням п'ятнадцять ват максимальна довжина траєкторії при швидкості нуль цілих три десятих метра на секунду становить близько двохсот метрів, що є достатнім для сканування приміщень площею до ста квадратних метрів.

Задача оптимізації траєкторії належить до класу задач покриття множини

та є NP-складною у загальному випадку. Це означає, що пошук глобально оптимального розв'язку за поліноміальний час є неможливим для приміщень довільної геометрії. Тому на практиці застосовуються евристичні алгоритми, що забезпечують знаходження субоптимальних розв'язків прийнятної якості за розумний час.

Для приміщень типів T_1, T_2 та T_3 з коефіцієнтом витягнутості менше 1,5 оптимальною є спіральна траєкторія, що забезпечує рівномірне покриття простору від периметра до центру. Спіральна траєкторія характеризується монотонним зменшенням відстані до геометричного центру приміщення при послідовному обході периметра.

Вибір спіральної траєкторії для ізотропних приміщень обумовлений декількома факторами. По-перше, спіральний рух забезпечує систематичне покриття всієї площі без пропусків та з контрольованим перекриттям зон видимості. По-друге, спіральна траєкторія мінімізує кількість різких поворотів, що зменшує похибки позиціонування та подовжує термін служби механізмів платформи. По-третє, рух від периметра до центру забезпечує сканування стін на ранніх етапах, що дозволяє швидко отримати загальну картину геометрії приміщення.

Крок спіралі визначається робочою дальністю датчика відстані та необхідним коефіцієнтом перекриття зон покриття. Для забезпечення відсутності прогалин у покритті при незначних відхиленнях від запланованої траєкторії застосовується перекриття на рівні 10 %. При робочій дальності датчика 8 м та коефіцієнті перекриття 0,1 крок спіралі становить 7,2 м.

Кількість витків спіралі визначається співвідношенням початкового радіуса (відстані від центру до найближчої точки периметра) та кроку спіралі. Для квадратного приміщення зі стороною шість метрів початковий радіус становить приблизно три метри, що відповідає менш ніж одному повному витку спіралі. Для більших приміщень площею від 40 $м^2$ до 60 $м^2$ кількість витків зростає до двох-трьох.

Орієнтація платформи на кожному кроці спіральної траєкторії

визначається таким чином, щоб напрямок сканування був спрямований радіально до центру приміщення. Така орієнтація забезпечує сканування стін приміщення на зовнішніх витках спіралі та поступовий перехід до сканування центральної частини на внутрішніх витках. При цьому датчик виконує повний оберт навколо вертикальної осі у кожній точці зупинки для отримання панорамного сканування.

Для приміщень з перешкодами (типи T_2 та T_3) спіральна траєкторія модифікується шляхом додавання локальних маневрів обходу. При виявленні перешкоди на шляху платформа відхиляється від ідеальної спіралі, огинає перешкоду та повертається до запланованого курсу. Додатково плануються точки зупинок з протилежного боку великих перешкод для покриття тіньових зон.

Для приміщень типу T_4 з коефіцієнтом витягнутості від 1,5 до 3 застосовується траєкторія типу меандр, що складається з паралельних проходів вздовж довгої сторони приміщення. Така траєкторія забезпечує систематичне покриття витягнутого простору з мінімальною кількістю поворотів.

Назва траєкторії походить від давньогрецького орнаменту у вигляді ламаної лінії з прямими кутами, що нагадує річку, яка звивається у долині. Траєкторія меандр є оптимальною для прямокутних областей, оскільки забезпечує повне покриття при мінімальній загальній довжині шляху серед усіх регулярних патернів.

Траєкторія меандр будується наступним чином. Визначається довга сторона приміщення як напрямок основних проходів. Вздовж цього напрямку виконуються паралельні проходи з кроком, що визначається робочою дальністю датчика та необхідним перекриттям. Наприкінці кожного проходу платформа виконує розворот на сто вісімдесят градусів та переміщується до наступної лінії проходу.

Крок між паралельними проходами визначається аналогічно кроку спіральної траєкторії з урахуванням кута розкриву сканування датчика. При повному оберті датчика та робочій дальності вісім метрів крок між проходами

становить сім цілих дві десятих метра. Для приміщення шириною чотири метри достатньо одного проходу по центру. Для приміщень шириною шість-вісім метрів потрібні два паралельних проходи.

На рисунку 2.3 представлено схему спіральної траєкторії сканування для квадратного приміщення з позначенням кроку спіралі та зон перекриття.

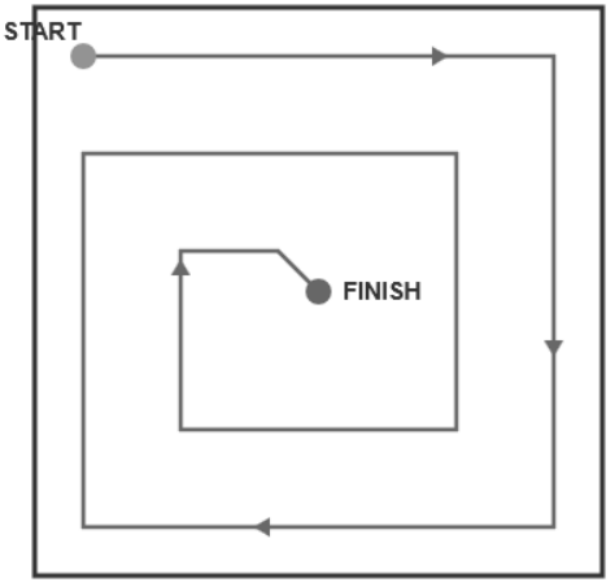

Рисунок 2.3 — Спіральна траєкторія сканування для квадратного приміщення

Кількість проходів траєкторії меандр визначається як частка від ділення ширини приміщення (меншого габаритного розміру) на крок між проходами, округлена до більшого цілого. Загальна довжина траєкторії меандр оцінюється як добуток кількості проходів на довжину кожного проходу плюс сумарна довжина переходів між проходами.

Для прямокутного приміщення розмірами вісім на чотири метри при кроці сім цілих дві десятих метра потрібен один прохід довжиною вісім метрів. Загальна довжина траєкторії при цьому становить лише вісім метрів, що значно менше, ніж для спіральної траєкторії того ж приміщення.

Для приміщень типу T_5 з коефіцієнтом витягнутості більше трьох застосовується лінійна траєкторія вздовж центральної осі коридору. Така траєкторія є оптимальною для витягнутих просторів, оскільки забезпечує повне покриття обох стін при мінімальній довжині шляху.

Лінійна траєкторія є найпростішою з усіх стратегій та складається з послідовності точок зупинок, розташованих вздовж центральної осі коридору. У кожній точці зупинки виконується повний оберт датчика з вимірюваннями у всіх напрямках. Оскільки ширина коридору значно менша за робочу дальність датчика, сканування з центральної осі забезпечує покриття обох стін одночасно.

Відстань між точками зупинок визначається умовою перекриття зон покриття вздовж осі коридору. Для типового коридору шириною 1,5 м максимальний кут від осі до стіни становить приблизно 45°. При робочій дальності 28 м та 10° перекритті крок між точками зупинок становить приблизно 5 м.

Кількість точок зупинок для коридору визначається як частка від ділення довжини коридору на крок, округлена до більшого цілого, плюс одна початкова точка. Для коридору довжиною 12 м при кроці 5 м потрібні три точки зупинок (на початку, посередині та наприкінці), що забезпечує повне покриття обох стін.

Загальна довжина лінійної траєкторії практично дорівнює довжині коридору, що робить цю стратегію найбільш ефективною з точки зору співвідношення покритої площі до довжини шляху. Час сканування коридору при лінійній траєкторії мінімальний серед усіх стратегій та визначається переважно часом переміщення між точками зупинок.

На рисунку 2.4 представлено схему лінійної траєкторії сканування для коридорного приміщення з позначенням точок зупинок та зон покриття.

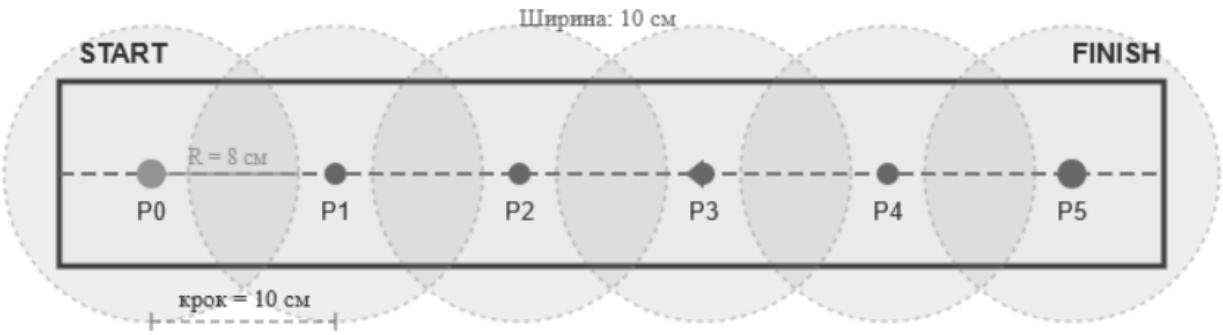

Рисунок 2.4 — Лінійна траєкторія сканування для коридорного приміщення

Для приміщень типу T_6 зі складною геометрією застосовується модифікований алгоритм швидко досліджуваних випадкових дерев (Rapidly-exploring Random Trees, RRT). Класичний алгоритм RRT був розроблений у кінці дев'яностих років Стівеном Лавалем [17] для планування руху роботів у просторах з перешкодами та швидко став одним з найпопулярніших методів у робототехніці.

Принцип роботи класичного алгоритму RRT полягає в ітеративній побудові дерева досяжності від початкової точки. На кожній ітерації генерується випадкова точка у вільному просторі, знаходиться найближча до неї вершина дерева, і якщо шлях між ними вільний від перешкод, нова вершина додається до дерева. Процес повторюється до досягнення цільової точки або вичерпання ресурсів.

Класичний алгоритм RRT має тенденцію до рівномірного дослідження простору, що не є оптимальним для задачі покриття. При скануванні приміщення метою є не досягнення конкретної цільової точки, а покриття максимальної площі. Тому класичний RRT витрачає надмірний час на повторне відвідування вже відсканованих областей.

Ключовою модифікацією запропонованого алгоритму є введення функції привабливості, яка спрямовує ріст дерева у напрямку невідсканованих областей. Замість рівномірно розподіленої випадкової точки генерується зважена комбінація випадкової точки та точки на межі відсканованої області (frontier):

$$p_{new} = \beta \times p_{rand} + (1 - \beta) \times p_{frontier} \qquad (2.5),$$

де p_{rand} — рівномірно розподілена випадкова точка;

$p_{frontier}$ — точка на межі невідсканованої області;

β — коефіцієнт балансу (оптимальне значення 0,4).

Детальну блок-схему модифікованого алгоритму RRT наведено в додатку В.

Межа невідсканованої області (frontier) визначається як множина точок,

що знаходяться на границі між відсканованою та невідсканованою частинами приміщення. Для ефективного обчислення frontier використовується растрове представлення карти покриття з поділом простору на комірки заданого розміру. Комірки, що мають сусідів як у відсканованій, так і у невідсканованій області, належать до frontier.

На рисунку 2.5 представлено блок-схему модифікованого алгоритму RRT з детальним відображенням логіки роботи.

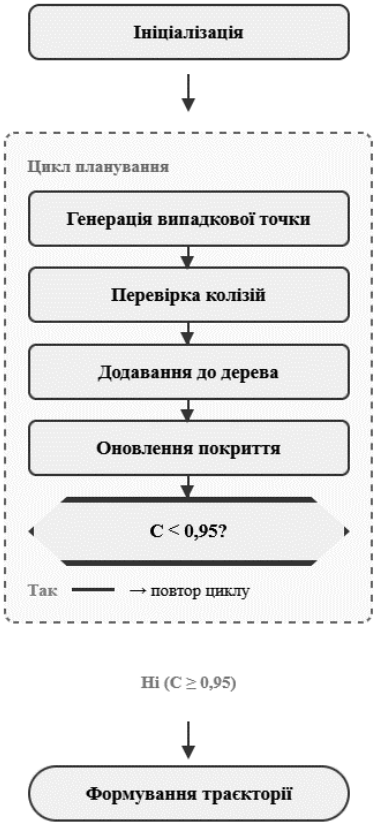

Рисунок 2.5 — Блок-схема модифікованого алгоритму RRT

Оптимальне значення коефіцієнта балансу визначено експериментально шляхом серії чисельних експериментів на тестових приміщеннях різної складності. При значеннях менше нуль цілих три десятих алгоритм надмірно орієнтується на покриття та генерує неоптимальні траєкторії з великою кількістю зайвих переміщень та різких поворотів. При значеннях більше нуль цілих шість десятих алгоритм поводиться як класичний RRT та може залишати невідскановані області у важкодоступних місцях.

Критерієм завершення роботи алгоритму є досягнення заданого мінімального коефіцієнта покриття. Типове значення мінімального коефіцієнта покриття для системи сканування приміщень становить нуль цілих дев'яносто п'ять сотих, що забезпечує покриття дев'яноста п'яти відсотків площі приміщення.

На рисунку 2.6 представлено візуалізацію роботи модифікованого алгоритму RRT для приміщення складної форми з Г-подібним плануванням, що є типовим представником класу T_6.

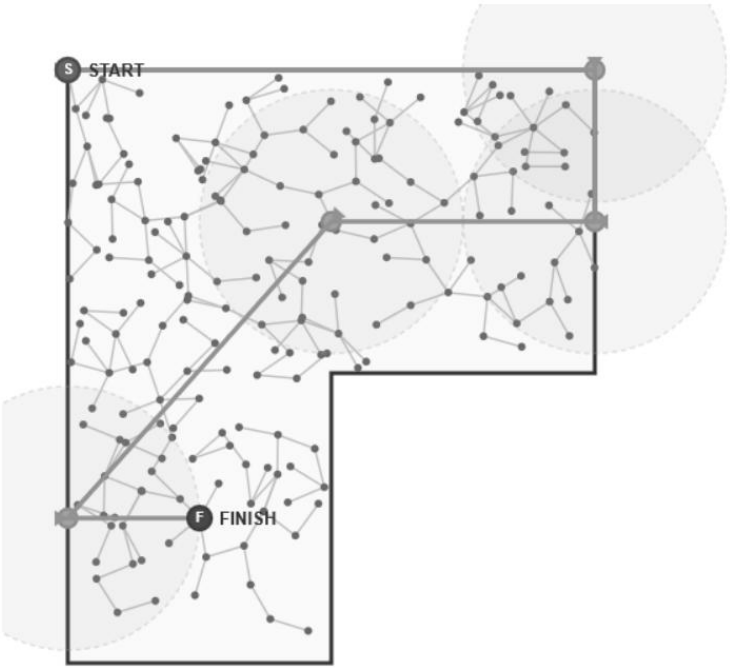

Рисунок 2.6 — Візуалізація роботи модифікованого алгоритму RRT

Для оцінки ефективності різних стратегій сканування виконано чисельне моделювання для набору тестових приміщень різних типів. Моделювання виконувалось у середовищі Python з використанням бібліотеки NumPy для чисельних обчислень та Matplotlib для візуалізації результатів [18,19]. Критеріями порівняння обрано: коефіцієнт покриття, довжину траєкторії, час сканування, кількість точок зупинок.

Набір тестових приміщень включав чотири типи: квадратне приміщення розмірами п'ять з половиною на п'ять з половиною метрів (площа тридцять

квадратних метрів), прямокутне приміщення розмірами сім з половиною на чотири метри (площа тридцять квадратних метрів), коридор розмірами п'ятнадцять на два метри (площа тридцять квадратних метрів), Г-подібне приміщення загальною площею тридцять квадратних метрів.

Для кожного приміщення виконувалось моделювання з усіма чотирма стратегіями сканування (спіральна, меандр, лінійна, RRT) для визначення оптимальної стратегії та порівняння ефективності. Результати моделювання підтвердили теоретичні очікування щодо оптимальності різних стратегій для різних типів приміщень.

У таблиці 2.2 наведено результати порівняльного моделювання для чотирьох типів тестових приміщень з різними стратегіями сканування.

Таблиця 2.2 — Результати порівняльного моделювання стратегій сканування

Приміщення	Стратегія	Покриття	Довжина, м	Час, хв
Квадратне 5,5×5,5	Спіральна	0,97	28	9
Квадратне 5,5×5,5	Меандр	0,94	35	12
Прямокутне 7,5×4	Меандр	0,96	22	7
Прямокутне 7,5×4	Спіральна	0,93	30	10
Коридор 15×2	Лінійна	0,98	15	5
Коридор 15×2	Меандр	0,95	32	11
Г-подібне	RRT	0,95	45	15
Г-подібне	Меандр	0,82	48	16

Результати моделювання підтверджують доцільність адаптивного вибору стратегії сканування залежно від типу приміщення. Спіральна траєкторія забезпечує найкращі результати для квадратних приміщень з коефіцієнтом покриття нуль цілих дев'яносто сім сотих при довжині траєкторії двадцять вісім метрів. Траєкторія меандр оптимальна для прямокутних приміщень з коефіцієнтом покриття нуль цілих дев'яносто шість сотих при довжині лише двадцять два метри.

Лінійна траєкторія демонструє найвищу ефективність для коридорів з

коефіцієнтом покриття нуль цілих дев'яносто вісім сотих при мінімальній довжині п'ятнадцять метрів. Модифікований алгоритм RRT є єдиною стратегією, що забезпечує прийнятне покриття для приміщень складної геометрії, досягаючи коефіцієнта нуль цілих дев'яносто п'ять сотих при довжині сорок п'ять метрів.

Особливо показовим є порівняння для Г-подібного приміщення, де застосування неадаптивної стратегії меандр призводить до падіння коефіцієнта покриття до нуль цілих вісімдесят дві сотих через неможливість систематичного покриття обох гілок Г-подібної форми. Алгоритм RRT, навпаки, адаптивно планує траєкторію з урахуванням геометрії та забезпечує покриття на рівні дев'яноста п'яти відсотків.

2.3 Математична модель оцінки точності геометричних вимірювань

Точність геометричних вимірювань є критичним параметром системи сканування приміщень, що визначає придатність результатів для використання у задачах оздоблювальних робіт. Замовники оздоблювальних робіт очікують точність визначення розмірів на рівні декількох міліметрів, що забезпечує коректний розрахунок кількості матеріалів та мінімізує відходи.

Сумарна похибка вимірювання складається з декількох компонентів, кожен з яких має різну природу та залежить від різних факторів. Основними джерелами похибок є: похибка датчика відстані (інструментальна похибка), похибка позиціонування мобільної платформи, похибка синхронізації даних, похибка алгоритмів обробки даних. Кожен з цих компонентів вносить незалежний внесок у сумарну похибку.

За принципом квадратичного підсумовування незалежних випадкових величин сумарна похибка вимірювання визначається як корінь квадратний з суми квадратів окремих компонентів:

$$\sigma_{\Sigma} = \sqrt{\sigma_d^2 + \sigma_p^2 + \sigma_s^2 + \sigma_a^2} \quad (2.6),$$

де σ_d — похибка датчика відстані;

σ_p — похибка позиціонування;

σ_s — похибка синхронізації;

σ_a — похибка алгоритмів обробки.

Такий підхід базується на припущенні про статистичну незалежність джерел похибок, що є справедливим для розглядуваної системи. Похибка датчика визначається фізичними властивостями лазерного далекоміра та не залежить від руху платформи. Похибка позиціонування визначається накопиченням помилок одометрії та не залежить від характеристик датчика відстані. Похибки синхронізації та обробки є програмними та визначаються якістю реалізації відповідних алгоритмів.

Похибка лазерного далекоміра TF-Luna визначається технічними характеристиками датчика та умовами вимірювання. Згідно з технічною специфікацією виробника Benewake [20], датчик має робочий діапазон від двадцяти сантиметрів до восьми метрів для цілей з високим коефіцієнтом відбиття та точність вимірювання плюс-мінус шість сантиметрів для ближньої зони та два відсотки для далекої зони.

Для типових умов вимірювання у приміщенні (відстань до стіни від двох до п'яти метрів, коефіцієнт відбиття поверхні п'ятдесят-сімдесят відсотків) похибка датчика оцінюється як двадцять міліметрів. Ця оцінка є консервативною та враховує вплив варіації коефіцієнта відбиття різних поверхонь, вплив кута падіння лазерного променя, вплив температури та вологості повітря, шуми електронного тракту.

Похибка позиціонування визначається точністю системи локалізації, що використовує комбінацію одометрії та алгоритму SLAM Cartographer. Одометрія забезпечує обчислення поточного положення на основі вимірювань обертання коліс, проте накопичує похибку з часом через проковзування коліс та неточність вимірювання кута повороту. Без корекції похибка одометрії зростає приблизно пропорційно до пройденого шляху.

Алгоритм Cartographer виконує корекцію накопиченої похибки одометрії шляхом зіставлення поточних вимірювань датчика відстані з побудованою картою [21,14]. При успішному закритті циклів (поверненні до раніше відвіданої області) алгоритм виявляє співпадіння та коригує всю траєкторію. Ефективна похибка позиціонування з корекцією SLAM оцінюється як п'ятнадцять міліметрів для типової траєкторії довжиною п'ятдесят метрів.

Похибка синхронізації виникає через часовий зсув між моментом вимірювання датчиком відстані та моментом фіксації положення платформи. При швидкості руху нуль цілих три десятих метра на секунду та частоті оновлення даних сто герц часовий зсув становить десять мілісекунд, що відповідає похибці три міліметри.

Похибка алгоритмів обробки включає похибки фільтрації шумів, сегментації хмари точок, виявлення площин та апроксимації поверхонь. Для використовуваних алгоритмів з бібліотеки Point Cloud Library ця похибка оцінюється як один-два міліметри, що є нехтовно малим внеском у сумарну похибку.

Підставляючи оцінки окремих компонентів у формулу квадратичного підсумовування, отримуємо сумарну очікувану похибку вимірювання для типового приміщення:

$$\sigma_{\Sigma} = \sqrt{20^2 + 15^2 + 3^2 + 2^2} \approx 25 \text{ мм} \quad (2.7)$$

Отримане значення є консервативною оцінкою для найгіршого випадку, що відповідає похибці окремого вимірювання точки. При сприятливих умовах (коротка траєкторія, успішне закриття циклів, стабільне освітлення, поверхні з високим коефіцієнтом відбиття) сумарна похибка може бути зменшена до п'ятнадцяти-двадцяти міліметрів.

Слід зазначити, що наведена оцінка стосується похибки окремого вимірювання точки хмари. При визначенні розмірів приміщення (відстань між стінами) використовується усереднення множини вимірювань, що зменшує

ефективну похибку за законом великих чисел. Ефективна похибка визначення положення площини стіни при типовій кількості вимірювань сто точок становить:

$$\sigma eff = \frac{\sigma_{\Sigma}}{\sqrt{N}} = \frac{25}{\sqrt{100}} = 2{,}5 \text{ мм} \qquad (2.8)$$

Ця оцінка підтверджує можливість досягнення точності визначення розмірів на рівні трьох-п'яти міліметрів, що відповідає технічним вимогам до системи та є достатнім для задач оздоблювальних робіт.

2.4 Математична модель розрахунку площ поверхонь приміщення

Площа підлоги є базовою характеристикою приміщення, що використовується для розрахунку кількості матеріалів підлогового покриття (ламінату, плитки, лінолеуму). Для приміщень полігональної форми площа обчислюється за класичною формулою Гаусса, відомою також як формула шнурування або формула землеміра:

$$S = \frac{1}{2} \sum_{1}^{n} (x_i y_{i+1} - x_{i+1} y_i) \qquad (2.9)$$

де (x_i, y_i) — координати i-ї вершини полігону контуру приміщення;

n — кількість вершин.

Формула Гаусса має просту геометричну інтерпретацію [22]. Кожний доданок представляє подвоєну площу трикутника, утвореного початком координат та і-ю стороною полігону з урахуванням знаку (додатня для обходу проти годинникової стрілки, від'ємна — за годинниковою). Сума цих знакових площ дає подвоєну площу полігону незалежно від його положення відносно початку координат.

Для типового приміщення з горизонтальною стелею площа стелі дорівнює площі підлоги. Це спрощення є справедливим для переважної більшості

житлових приміщень стандартного планування. Для приміщень зі складною формою стелі (мансарди, приміщення з нішами та виступами на стелі) площа обчислюється як сума площ окремих плоских фрагментів.

Похибка визначення площі підлоги залежить від похибок визначення координат вершин полігону. Для прямокутного приміщення з розмірами п'ять на шість метрів та абсолютною похибкою визначення розмірів п'ять міліметрів відносна похибка площі становить приблизно нуль цілих тринадцять сотих відсотка. Абсолютна похибка площі при цьому становить близько чотирьох сотих квадратного метра, що є цілком прийнятним для практичних застосувань.

Площа стін визначає кількість матеріалів для оздоблення вертикальних поверхонь (шпалер, фарби, декоративної штукатурки, плитки). Для типового приміщення з вертикальними стінами площа обчислюється як добуток периметра на висоту:

$$S_w = H * P \qquad (2.10),$$

де H — висота приміщення, м;

P — периметр приміщення на рівні підлоги, м.

Для практичних застосувань необхідно враховувати площу віконних та дверних прорізів, що зменшують корисну площу стін. Чиста площа стін визначається як різниця між загальною площею та сумарною площею прорізів. Площа типового дверного прорізу розмірами два на нуль цілих вісім десятих метра становить один і шість десятих квадратного метра. Площа стандартного вікна розмірами півтора на один і дві десятих метра становить один і вісім десятих квадратного метра.

Для прямокутного приміщення п'ять на шість метрів з висотою два і сім десятих метра, одним дверним та одним віконним прорізом загальна площа стін становить п'ятдесят дев'ять і чотири десятих квадратного метра, а чиста площа після вирахування прорізів — п'ятдесят шість квадратних метрів. Ця інформація є критичною для точного розрахунку кількості шпалер або фарби.

На рисунку 2.7 представлено схему розрахунку площ поверхонь приміщення з позначенням основних геометричних параметрів та формул.

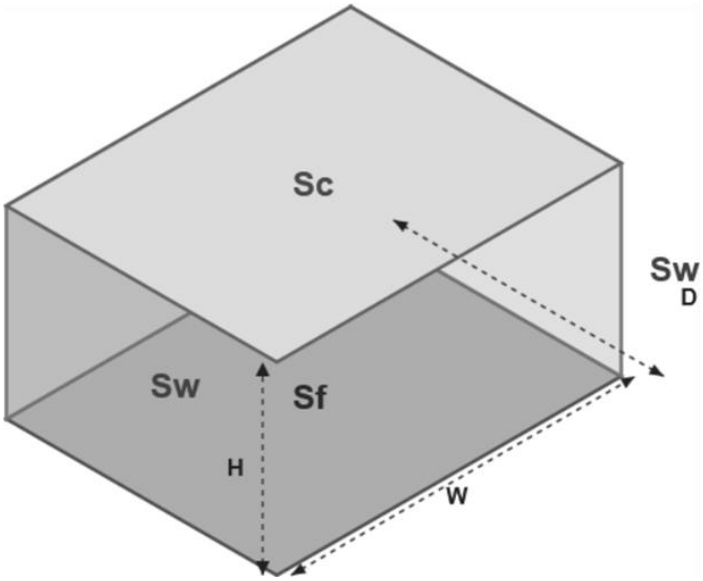

Рисунок 2.7 — Схема розрахунку площ поверхонь приміщення

2.5 Обґрунтування параметрів системи на основі теоретичного аналізу

На основі проведеного теоретичного аналізу визначено вимоги до основних компонентів системи, що забезпечують виконання технічних вимог при оптимальному використанні ресурсів. Вимоги до датчика відстані визначаються необхідністю покриття типових житлових приміщень та забезпечення заданої точності вимірювань.

Мінімальна робоча дальність датчика визначається необхідністю покриття діагоналі найбільшого типового приміщення. Для приміщення з максимальними розмірами вісім на шість метрів діагональ становить десять метрів. Однак сканування з однієї точки не є обов'язковою вимогою, оскільки мобільна платформа може переміщатися всередині приміщення. З урахуванням перекриття зон покриття достатньою є дальність сім-вісім метрів. Обраний датчик TF-Luna з робочою дальністю вісім метрів задовольняє цю вимогу.

Точність датчика повинна забезпечувати виконання технічних вимог до системи. З урахуванням усіх джерел похибок точність датчика повинна

становити не більше двох відсотків від виміряної відстані, що забезпечується датчиком TF-Luna. Частота вимірювань датчика повинна бути достатньою для забезпечення просторової роздільної здатності при заданій швидкості руху платформи. Частота двісті п'ятдесят герц датчика TF-Luna забезпечує роздільну здатність один міліметр при швидкості нуль цілих двадцять п'ять сотих метра на секунду.

Швидкість переміщення платформи обмежується умовою забезпечення достатньої щільності вимірювань та стійкості руху. Максимальна швидкість при заданих параметрах датчика становить два з половиною метри на секунду, однак на практиці швидкість обмежується до нуль цілих три десятих — нуль цілих п'ять десятих метра на секунду для забезпечення плавності руху та зменшення вібрацій.

Час автономної роботи визначається часом сканування найбільшого типового приміщення з урахуванням резерву на непередбачені ситуації. Для приміщення площею шістдесят квадратних метрів максимальний час сканування становить двадцять хвилин. З урахуванням коефіцієнта запасу півтора необхідний час автономної роботи становить тридцять хвилин. Обрана платформа забезпечує час автономної роботи до шістдесяти хвилин, що задовольняє цю вимогу з подвійним запасом.

Параметри алгоритмів планування траєкторії визначено на основі чисельного моделювання та експериментальної верифікації. Ключовим параметром є крок траєкторії (відстань між сусідніми лініями сканування), який визначає баланс між повнотою покриття та часом сканування.

При занадто малому кроці забезпечується надмірне перекриття зон покриття, що збільшує час сканування без суттєвого покращення результату. При занадто великому кроці виникають прогалини у покритті, що призводить до пропуску архітектурних деталей. Оптимальний крок визначається як дев'яносто відсотків від робочої дальності датчика, що забезпечує десятивідсоткове перекриття зон покриття.

Параметр β алгоритму модифікованого RRT визначає баланс між

випадковим дослідженням простору та цілеспрямованим рухом до невідсканованих областей. У таблиці 2.3 наведено результати моделювання роботи алгоритму RRT для тестового приміщення складної форми при різних значеннях параметра.

Таблиця 2.3 — Залежність характеристик алгоритму RRT від параметра β

β	Покриття	Ітерації	Довжина, м	Час, с
0,2	0,97	280	52	18
0,3	0,96	220	48	15
0,4	0,95	180	45	13
0,5	0,94	200	49	14
0,6	0,92	250	55	17

Аналіз результатів показує, що оптимальне значення $\beta = 0,4$ забезпечує найкращий баланс між коефіцієнтом покриття, кількістю ітерацій та довжиною траєкторії. При цьому значенні алгоритм досягає покриття дев'яносто п'ять відсотків за сто вісімдесят ітерацій з довжиною траєкторії сорок п'ять метрів та часом виконання тринадцять секунд.

Таким чином, теоретичний аналіз дозволив обґрунтувати основні параметри системи: робоча дальність датчика вісім метрів, швидкість переміщення нуль цілих три десятих метра на секунду, час автономної роботи шістдесят хвилин, крок траєкторії сім цілих дві десятих метра, параметр $\beta = 0,4$ для алгоритму RRT. Ці параметри забезпечують виконання технічних вимог до системи при оптимальному використанні ресурсів.

3 РОЗРОБКА ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ

3.1 Розробка апаратної частини системи

Апаратна частина системи сканування та візуалізації приміщень базується на модифікації серійного робота-пилососа Rowenta X-plorer Serie 135. Вибір даної платформи обумовлений наявністю готової механічної бази з системою переміщення, компактними розмірами та можливістю інтеграції додаткового обладнання. Модифікація передбачає встановлення одноплатного комп'ютера Raspberry Pi 4B, лазерного далекоміра TF-Luna на поворотній платформі з сервоприводом MG996R та інерціального модуля MPU-6050.

Структурна схема апаратної частини системи наведена на рисунку 3.1. Центральним елементом системи є одноплатний комп'ютер Raspberry Pi 4B, який виконує функції збору даних від сенсорів, обробки інформації, класифікації приміщень та обслуговування веб-інтерфейсу. Raspberry Pi 4B обрано завдяки достатній обчислювальній потужності (чотирьохядерний процесор ARM Cortex-A72 з тактовою частотою 1,5 ГГц), наявності необхідних інтерфейсів (GPIO, UART, I2C, SPI) та підтримці бездротового зв'язку Wi-Fi 802.11ac [23].

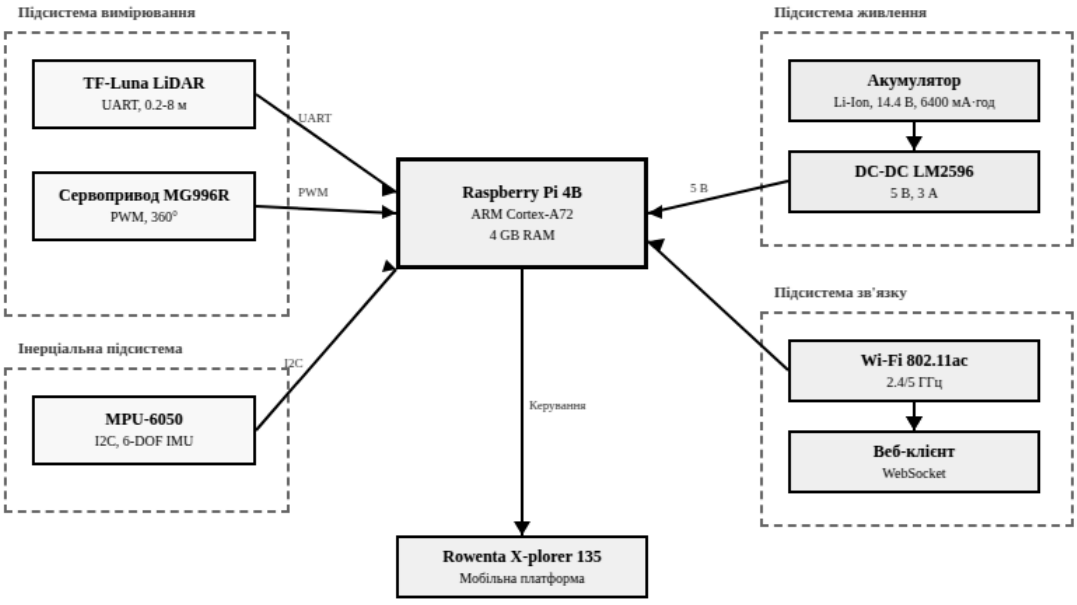

Рисунок 3.1 — Структурна схема апаратної частини системи

Підсистема вимірювання відстаней складається з лазерного далекоміра TF-Luna та сервоприводу MG996R. Лазерний далекомір TF-Luna працює за принципом часу прольоту (Time-of-Flight, ToF) та забезпечує вимірювання відстаней у діапазоні від 0,2 м до 8 м з точністю ±2 см при частоті оновлення до 250 Гц. Зв'язок з Raspberry Pi здійснюється через інтерфейс UART на швидкості 115200 бод. Технічні характеристики лазерного далекоміра TF-Luna наведено у таблиці 3.1.

Таблиця 3.1 — Технічні характеристики лазерного далекоміра TF-Luna

Параметр	Значення
Діапазон вимірювань	Від 0,2 м до 8 м
Точність	±2 см (при 0,2—3 м)
Роздільна здатність	1 см
Частота оновлення	Від 1 Гц до 250 Гц
Кут розходження променя	2°
Довжина хвилі	850 нм
Інтерфейс	UART / I2C
Напруга живлення	3,7 В до 5,2 В
Споживаний струм	70 мА (типовий)
Робоча температура	Від -20 °C до 60 °C
Габаритні розміри	35×21,25×13,5 мм
Маса	5 г

Сервопривід MG996R забезпечує обертання лазерного далекоміра на 360° для кругового сканування приміщення. Вибір даної моделі обумовлений високим крутним моментом (9,4 кг·см при 4,8 В), достатньою точністю позиціонування (±1°) та металевим редуктором, що забезпечує довговічність при інтенсивному використанні [24]. Керування сервоприводом здійснюється за допомогою ШІМ-сигналу з частотою 50 Гц через GPIO-піни Raspberry Pi. Інерціальний модуль MPU-6050 використовується для визначення орієнтації системи в просторі та компенсації похибок одометрії. Модуль поєднує трьохосьовий акселерометр та трьохосьовий гіроскоп з цифровим інтерфейсом I2C[25]. Дані з інерціального модуля використовуються для корекції кутової орієнтації системи під час сканування та виявлення нерівностей поверхні. Технічні характеристики сервоприводу MG996R наведено у таблиці 3.2.

Таблиця 3.2 — Технічні характеристики сервоприводу MG996R

Параметр	Значення
Крутний момент	9,4 кг·см (4,8 В)
Кут повороту	Від 0° до180° (модифіковано до 360°)
Швидкість обертання	0,17 с/60° (4,8 В)
Робоча напруга	Від 4,8 В до 7,2 В
Споживаний струм	Від 500 мА до 900 мА
Мертва зона	5 мкс
Тип редуктора	Металевий
Габаритні розміри	40,7×19,7×42,9 мм
Маса	55 г

Підключення компонентів до Raspberry Pi 4B здійснюється через GPIO-роз'єм. Лазерний далекомір TF-Luna підключається до інтерфейсу UART0 (GPIO14/TXD та GPIO15/RXD), сервопривод MG996R керується через GPIO17 з використанням апаратного ШІМ, інерціальний модуль MPU-6050 підключається до шини I2C1 (GPIO2/SDA та GPIO3/SCL). Технічні характеристики MPU-6050 наведено у таблиці 3.3.

Таблиця 3.3 — Технічні характеристики інерціального модуля MPU-6050

Параметр	Значення
Діапазон акселерометра	±2g, ±4g, ±8g, ±16g
Діапазон гіроскопа	±250, ±500, ±1000, ±2000 °/с
Роздільна здатність АЦП	16 біт
Частота опитування	до 1000 Гц
Інтерфейс	I2C (до 400 кГц)
Напруга живлення	Від 2,375 В до 3,46 В
Споживаний струм	3,9 мА (типовий)
Габаритні розміри модуля	20×16×1 мм

Система живлення базується на акумуляторі робота-пилососа з номінальною ємністю 6400 мА·год та напругою 14,4 В. Для живлення Raspberry Pi та периферійних пристроїв використовується DC-DC перетворювач на базі мікросхеми LM2596, який забезпечує стабілізовану напругу 5 В при струмі до 3 А. Окремий канал живлення 5 В виділено для сервоприводу через транзисторний ключ, що дозволяє програмно вимикати сервопривод для економії енергії в

режимі очікування. Загальне споживання системи в активному режимі становить приблизно 2,5 А, що забезпечує автономність роботи близько 90 хвилин.
Схема підключення компонентів до GPIO Raspberry Pi наведена у таблиці 3.4.

Таблиця 3.4 — Схема підключення компонентів до GPIO Raspberry Pi 4B

Компонент	Сигнал	GPIO	Примітка
TF-Luna	TX	GPIO15 (RXD)	UART0
TF-Luna	RX	GPIO14 (TXD)	UART0
TF-Luna	VCC	5V	Pin 2
TF-Luna	GND	GND	Pin 6
MG996R	Signal	GPIO17	PWM0
MG996R	VCC	5V	Зовнішнє
MG996R	GND	GND	Спільний
MPU-6050	SDA	GPIO2	I2C1
MPU-6050	SCL	GPIO3	I2C1
MPU-6050	VCC	3.3V	Pin 1
MPU-6050	GND	GND	Pin 9

Конструктивне виконання системи передбачає розміщення Raspberry Pi у спеціальному корпусі на верхній панелі робота-пилососа. Поворотна платформа з лазерним далекоміром та сервоприводом встановлена в геометричному центрі платформи для забезпечення симетричного сканування. Інерціальний модуль MPU-6050 закріплено безпосередньо на корпусі Raspberry Pi для мінімізації вібрацій.

Для забезпечення завадостійкості та надійної роботи системи передбачено ряд конструктивних рішень. Кабелі підключення сенсорів екрановані та прокладені окремо від силових ліній живлення. На лініях живлення встановлено фільтруючі конденсатори для згладжування імпульсних завад. Корпус Raspberry Pi забезпечує природну конвекцію для відведення тепла від процесора. Оптичне вікно лазерного далекоміра захищено від пилу та механічних пошкоджень прозорою кришкою. Загальний вигляд зібраної системи наведено на рисунку 3.2.

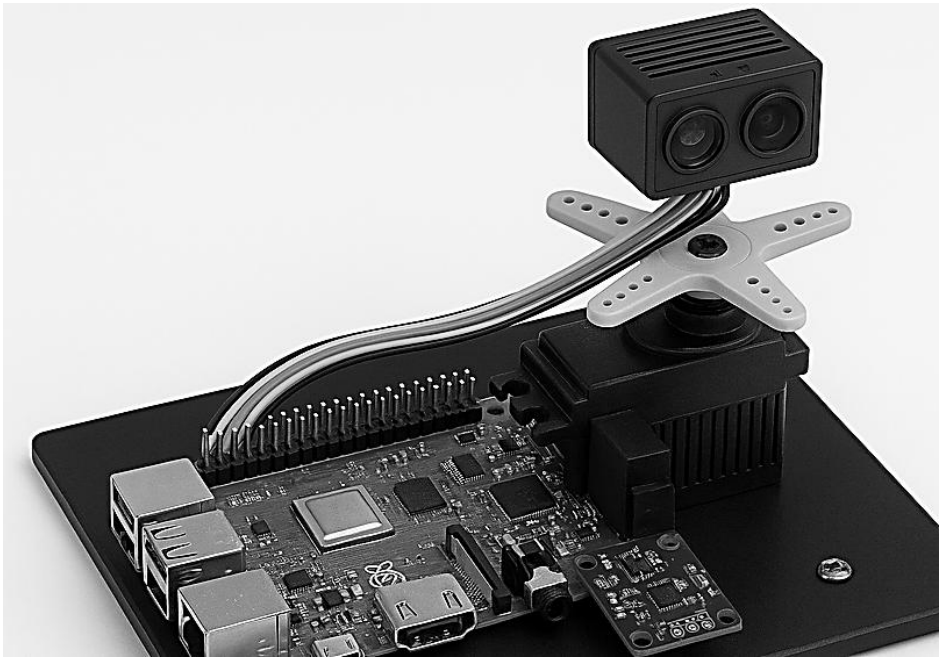

Рисунок 3.2 — Загальний вигляд апаратної частини системи

Програмна ініціалізація апаратної частини виконується при запуску системи. Спочатку відбувається калібрування інерціального модуля MPU-6050 шляхом усереднення 100 вимірювань у стані спокою для визначення зміщення нуля. Далі виконується ініціалізація UART-з'єднання з лазерним далекоміром та перевірка коректності отримуваних даних. На завершення відбувається калібрування сервоприводу — встановлення початкового положення та перевірка діапазону обертання.

3.2 Розробка програмного забезпечення мобільної платформи

Програмне забезпечення мобільної платформи розроблено мовою Python 3.9 з використанням асинхронної архітектури на базі бібліотеки aiohttp [26,27]. Асинхронний підхід обрано для забезпечення паралельної обробки даних від сенсорів та одночасного обслуговування веб-клієнтів без блокування основного потоку виконання. Архітектура програмного забезпечення базується на модульному принципі з чітким розділенням відповідальності між компонентами. Загальну архітектуру програмного забезпечення мобільної платформи наведено на рисунку 3.3.

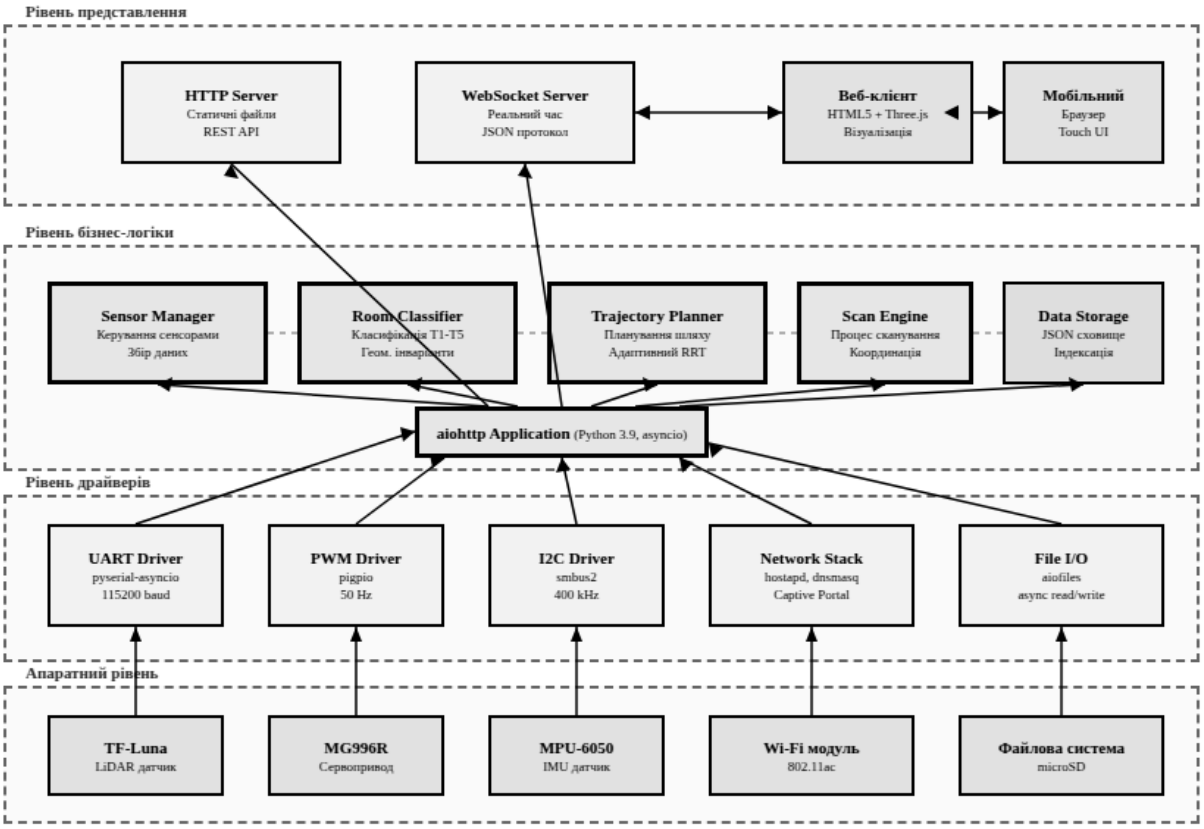

Рисунок 3.3 — Архітектура програмного забезпечення мобільної платформи

Загальна архітектура програмного забезпечення включає наступні основні модулі: модуль керування сенсорами (sensor_manager), модуль класифікації приміщень (room_classifier), модуль планування траєкторії (trajectory_planner), WebSocket-сервер для зв'язку з клієнтами (ws_server), HTTP-сервер для роздачі статичних файлів (http_server), модуль збереження даних (data_storage). Діаграму модулів серверної частини системи наведено в додатку Г. Взаємодія між модулями здійснюється через асинхронні черги повідомлень та спільні структури даних з синхронізацією доступу.

Модуль керування сенсорами відповідає за взаємодію з апаратними компонентами системи. Для роботи з лазерним далекоміром TF-Luna використовується асинхронний драйвер UART на базі бібліотеки pyserial- asyncio [28]. Драйвер реалізує протокол обміну даними з датчиком, включаючи формування команд налаштування, парсинг пакетів відповіді та обробку помилок комунікації. Частота опитування датчика встановлена на рівні

100 Гц, що забезпечує баланс між точністю сканування та навантаженням на процесор.

Керування сервоприводом MG996R здійснюється через бібліотеку pigpio, яка забезпечує апаратну генерацію ШІМ-сигналу з мікросекундною точністю [29]. Для плавного обертання датчика реалізовано алгоритм інтерполяції кутового положення з врахуванням інерції механічної системи. Швидкість обертання адаптивно регулюється залежно від режиму сканування: повільне обертання (10°/с) для детального сканування та швидке (60°/с) для експрес-аналізу.

Взаємодія з інерціальним модулем MPU-6050 реалізована через бібліотеку smbus2 для роботи з шиною I2C [30]. Модуль виконує зчитування даних акселерометра та гіроскопа з частотою 100 Гц та обчислює орієнтацію системи за допомогою комплементарного фільтра. Отримані дані використовуються для корекції кутових вимірювань лазерного далекоміра та виявлення моментів нестабільності платформи.

Модуль класифікації приміщень реалізує алгоритм визначення типу кімнати на основі геометричних інваріантів, описаних у теоретичному розділі роботи. На вхід модуля подається масив точок сканування у полярних координатах, на виході — тип приміщення (T1—T5) та рекомендована стратегія сканування. Функції classify_room наведено на лістингу 3.1.

Лістинг 3.1 — Функція класифікації приміщення

```
def classify_room(length: float, width: float,
                  complexity: float = 1.0, occupancy: float = 0.2) -> Dict:
    ka = max(length, width) / min(length, width)  # Витягнутість
    kg = complexity  # Складність форми
    ko = occupancy   # Заповнення
    if ka < 1.3:
        if ko < 0.2:
            room_type, type_name = "T1", "Квадратне вільне"
        elif ko < 0.4:
            room_type, type_name = "T2", "Квадратне середньозаповнене"
        else:
            room_type, type_name = "T3", "Квадратне заповнене"
    elif ka < 2.5:
```

```
    room_type, type_name = "T4", "Прямокутне приміщення"
  else:
    room_type, type_name = "T5", "Коридорне приміщення"
  return {'type': room_type, 'typeName': type_name,
          'invariants': {'ka': round(ka, 2), 'kg': round(kg, 2), 'ko': round(ko, 2)}}
```

Для кожного типу приміщення визначено оптимальну стратегію сканування, яка враховує геометричні особливості та мінімізує час обходу. Відповідність типів приміщень та стратегій сканування наведено у таблиці 3.5.

Таблиця 3.5 — Стратегії сканування для різних типів приміщень

Тип	Форма приміщення	Стратегія сканування
T1	Прямокутне	Паралельні проходи вздовж довшої сторони
T2	Г-подібне	Розбиття на прямокутні зони, послідовний обхід
T3	Складної форми	Адаптивний RRT з локальною оптимізацією
T4	Витягнуте (коридор)	Зигзагоподібний рух вздовж осі
T5	Нестандартне	Повний RRT з максимальним покриттям

WebSocket-сервер забезпечує двонаправлений обмін даними між мобільною платформою та веб-інтерфейсом у реальному часі. Сервер реалізовано на базі бібліотеки aiohttp з підтримкою протоколу WebSocket RFC 6455 [31]. Для кожного підключеного клієнта створюється окрема асинхронна задача, яка обробляє вхідні повідомлення та надсилає оновлення стану системи. Протокол обміну базується на JSON-повідомленнях з полями type (тип повідомлення) та data (дані).

Сервер підтримує наступні типи команд від клієнта: get_scans — отримання списку збережених сканувань, get_scan — отримання даних конкретного сканування за ідентифікатором, start_scan — запуск нового сканування, stop_scan — зупинка поточного сканування, delete_scan — видалення сканування. Під час сканування сервер періодично надсилає клієнту повідомлення типу scan_progress з поточними результатами для відображення прогресу в реальному часі. Обробник WebSocket-підключень напевно в лістингу 3.2.

Лістинг 3.2 — Обробник WebSocket-підключень

```
async def websocket_handler(request):
  ws = web.WebSocketResponse()
```

```
    await ws.prepare(request)
    clients.add(ws)
    try:
        async for msg in ws:
            if msg.type == WSMsgType.TEXT:
                data = json.loads(msg.data)
                cmd = data.get('type')
                if cmd == 'get_scans':
                    scans = await storage.get_all_scans()
                    await ws.send_json({'type': 'scans_list', 'data': scans})
                elif cmd == 'start_scan':
                    scan_id = await scanner.start_scan()
                    await ws.send_json({'type': 'scan_started', 'data': {'id': scan_id}})
                elif cmd == 'stop_scan':
                    await scanner.stop_scan()
                    await ws.send_json({'type': 'scan_stopped'})
    finally:
        clients.discard(ws)
    return ws
```

Модуль збереження даних реалізує персистентне зберігання результатів сканування у файловій системі. Кожне сканування зберігається у вигляді JSON-файлу з унікальним ідентифікатором (UUID) та містить метадані (дата, час, тип приміщення), масив точок сканування та обчислені характеристики (площа, периметр, розміри). Для оптимізації завантаження при великій кількості сканувань реалізовано окремий індексний файл з метаданими всіх сканувань.

HTTP-сервер забезпечує роздачу статичних файлів веб-інтерфейсу та реалізацію Captive Portal для спрощеного підключення клієнтів. При підключенні пристрою до Wi-Fi мережі, створеної Raspberry Pi, автоматично відкривається сторінка веб-інтерфейсу. Це досягається шляхом перехоплення DNS-запитів та перенаправлення на локальний сервер. Конфігурація Captive Portal виконана за допомогою утиліт hostapd та dnsmasq.

Для забезпечення надійності та відмовостійкості програмного забезпечення реалізовано механізм обробки виняткових ситуацій та автоматичного відновлення. При виникненні помилок комунікації з сенсорами система виконує перепідключення з експоненційною затримкою. Критичні помилки логуються у файл для подальшого аналізу. Сторожовий таймер

(watchdog) перезапускає систему у разі зависання основного процесу.

Тестування програмного забезпечення виконувалось на всіх етапах розробки. Модульні тести перевіряли коректність роботи окремих функцій (класифікація, обчислення інваріантів, парсинг даних). Інтеграційні тести перевіряли взаємодію модулів та роботу WebSocket-протоколу. Навантажувальні тести підтвердили стабільну роботу системи при одночасному підключенні до 10 клієнтів.

3.3 Розробка веб-інтерфейсу візуалізації

Веб-інтерфейс візуалізації результатів сканування реалізовано у вигляді односторінкового веб-додатку (Single Page Application) з використанням технологій HTML5, CSS3 та JavaScript. Інтерфейс забезпечує відображення тривимірної моделі приміщення, керування процесом сканування та експорт результатів. Вибір веб-технологій обумовлений кросплатформністю — інтерфейс працює на будь-якому пристрої з сучасним браузером без необхідності встановлення додаткового програмного забезпечення.

Архітектура веб-інтерфейсу базується на патерні Model-View-Controller з розділенням логіки роботи з даними, візуалізації та обробки подій користувача. Для тривимірної візуалізації використовується бібліотека Three.js, яка надає високорівневий API для роботи з WebGL та забезпечує апаратне прискорення графіки [32]. Зв'язок з сервером здійснюється через WebSocket для отримання даних у реальному часі.

Ініціалізація 3D-сцени виконується при завантаженні сторінки. Створюються основні об'єкти Three.js: сцена (Scene), камера (PerspectiveCamera), рендерер (WebGLRenderer) та контролер камери (OrbitControls). Камера налаштовується на ізометричний вигляд з можливістю обертання та масштабування. Освітлення сцени реалізовано комбінацією розсіяного (AmbientLight) та спрямованого (DirectionalLight) джерел світла. Структуру веб-інтерфейсу візуалізації наведено на рисунку 3.4. Ініціалізація Three.js сцени наведено в лістингу 3.3.

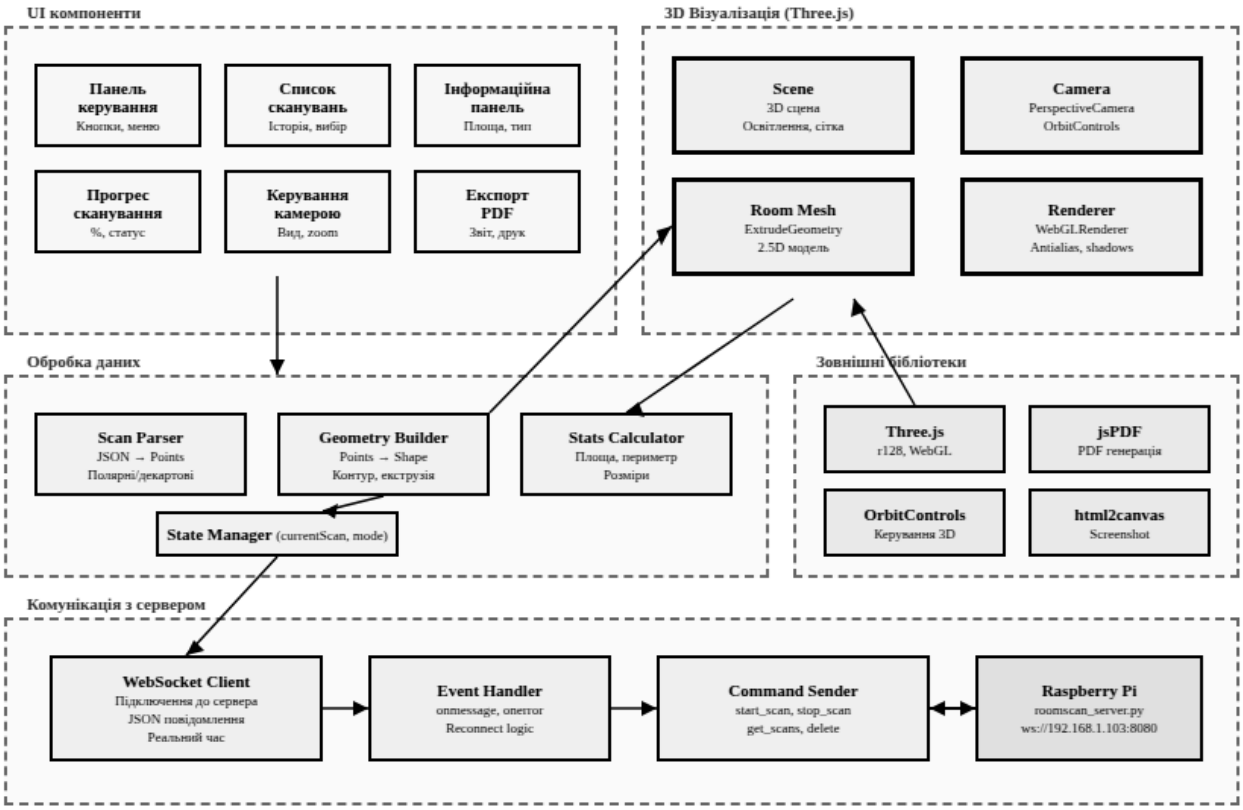

Рисунок 3.4 — Структура веб-інтерфейсу візуалізації

Лістинг 3.3 — Ініціалізація Three.js сцени

```
function initThree() {
  // Створення сцени
  scene = new THREE.Scene();
  scene.background = new THREE.Color(0xf5f5f5);
  // Налаштування камери
  camera = new THREE.PerspectiveCamera(60, width / height, 0.1, 1000);
  camera.position.set(10, 10, 10);
  camera.lookAt(0, 0, 0);
  // Створення рендерера
  renderer = new THREE.WebGLRenderer({ antialias: true });
  renderer.setSize(width, height);
  renderer.setPixelRatio(window.devicePixelRatio);
  container.appendChild(renderer.domElement);
  // Контролер камери
  controls = new THREE.OrbitControls(camera, renderer.domElement);
  controls.enableDamping = true;
  controls.dampingFactor = 0.05;
  // Освітлення
  scene.add(new THREE.AmbientLight(0xffffff, 0.6));
  const dirLight = new THREE.DirectionalLight(0xffffff, 0.8);
  dirLight.position.set(10, 20, 10);
```

```
  scene.add(dirLight);
  // Сітка підлоги
  const grid = new THREE.GridHelper(20, 20, 0xcccccc, 0xe0e0e0);
  scene.add(grid);
}
```

Візуалізація результатів сканування виконується у режимі 2.5D — контур приміщення екструдується на стандартну висоту стелі (2,7 м за замовчуванням). Такий підхід забезпечує наочне представлення форми кімнати при використанні планарного лазерного сканування. Стіни відображаються напівпрозорими для можливості огляду внутрішнього простору. Перешкоди (меблі, колони) візуалізуються як окремі блоки всередині контуру.

Побудова 2.5D-моделі приміщення, яку наведено лістингу 3.4, виконується на основі масиву точок сканування. Спочатку точки перетворюються з полярних координат у декартові. Далі будується замкнутий контур з використанням класу THREE.Shape. Контур екструдується на задану висоту за допомогою THREE.ExtrudeGeometry. Результуюча геометрія перетворюється на меш з матеріалом THREE.MeshPhongMaterial для реалістичного відображення.

Лістинг 3.4 — Побудова 2.5D моделі приміщення

```
function createRoom(scanData) {
  const points = scanData.points.map(p => ({
    x: p.distance * Math.cos(p.angle * Math.PI / 180),
    z: p.distance * Math.sin(p.angle * Math.PI / 180)
  }));
  const shape = new THREE.Shape();
  shape.moveTo(points[0].x, points[0].z);
  points.slice(1).forEach(p => shape.lineTo(p.x, p.z));
  shape.closePath();
  // Екструзія на висоту стелі
  const geometry = new THREE.ExtrudeGeometry(shape, {
    depth: scanData.ceilingHeight || 2.7,
    bevelEnabled: false
  })
  const material = new THREE.MeshPhongMaterial({
    color: 0x4a90d9,
    transparent: true,
```

```
        opacity: 0.7,
        side: THREE.DoubleSide
    });
    const mesh = new THREE.Mesh(geometry, material);
    mesh.rotation.x = -Math.PI / 2;
    scene.add(mesh);
}
```

Інтерактивне керування камерою реалізовано за допомогою класу OrbitControls з бібліотеки Three.js. Користувач може обертати модель за допомогою лівої кнопки миші або жестів на сенсорному екрані, масштабувати колесом миші або жестом pinch, панорамувати правою кнопкою миші або жестом двома пальцями. Для зручності додано кнопки швидкого переходу до стандартних видів: зверху, спереду, збоку, ізометричний.

Інформаційна панель відображає характеристики відсканованого приміщення: тип за класифікацією (T1—T5), площу у квадратних метрах, периметр, максимальні розміри (довжина, ширина), дату та час сканування. При наведенні курсору на окремі елементи моделі відображається спливаюча підказка з додатковою інформацією.

Функція експорту результатів у PDF реалізована за допомогою бібліотеки jsPDF [33]. Документ включає знімок 2.5D-моделі (зроблений за допомогою renderer.domElement.toDataURL), таблицю характеристик приміщення, інформацію про геометричні інваріанти та метадані сканування. Для створення знімка моделі камера автоматично позиціонується на ізометричний вид. Реалізація експорту файлів у PDF показана в лістингу 3.4.

Лістинг 3.5 — Експорт результатів у PDF

```
async function exportToPDF(scanData) {
    const pdf = new jsPDF('p', 'mm', 'a4');
    pdf.setFontSize(18);
    pdf.text('Звіт про сканування приміщення', 105, 20, { align: 'center' });
    positionCameraForScreenshot();
    renderer.render(scene, camera);
    const imageData = renderer.domElement.toDataURL('image/png');
    pdf.addImage(imageData, 'PNG', 20, 30, 170, 100);
    // Таблиця характеристик
```

```
    pdf.setFontSize(12);
    const tableData = [
        ['Параметр', 'Значення'],
        ['Тип приміщення', scanData.roomType],
        ['Площа', scanData.area.toFixed(2) + ' м²'],
        ['Периметр', scanData.perimeter.toFixed(2) + ' м'],
        ['Довжина', scanData.length.toFixed(2) + ' м'],
        ['Ширина', scanData.width.toFixed(2) + ' м'],
        ['Дата сканування', new Date(scanData.timestamp).toLocaleString()]
    ];
    pdf.autoTable({
        startY: 140,
        head: [tableData[0]],
        body: tableData.slice(1),
        theme: 'grid'
    });
    pdf.save('scan_report.pdf');
}
```

Веб-інтерфейс адаптований для роботи на пристроях з різними розмірами екрану. На мобільних пристроях елементи керування збільшуються для зручності натискання, панель інформації переміщується під область візуалізації. Підтримуються сенсорні жести для керування камерою. Мінімальна підтримувана роздільна здатність — 320×480 пікселів.

Продуктивність веб-інтерфейсу оптимізовано для роботи на пристроях з обмеженими ресурсами. Використовується механізм Level of Detail (LOD) для зменшення деталізації моделі при віддаленні камери [34]. Рендеринг виконується тільки при зміні сцени (on-demand rendering). Текстури та геометрії кешуються для повторного використання. Цільова частота кадрів — 60 FPS на сучасних пристроях, мінімум 30 FPS на застарілих.

Сумісність веб-інтерфейсу перевірено на основних браузерах: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+ [35]. Для браузерів без підтримки WebGL відображається повідомлення про необхідність оновлення. Інтерфейс коректно працює при відключеному JavaScript з обмеженою функціональністю (тільки перегляд статичних даних).

Практична реалізація веб-інтерфейсу включає декілька функціональних

екранів, що забезпечують повний цикл роботи з системою сканування.

При завантаженні веб-застосунку користувач бачить екран підключення (рис. 3.5), який пропонує два варіанти роботи: кнопка «Підключитись» ініціює з'єднання з Raspberry Pi через WebSocket, кнопка «Демо-режим» дозволяє ознайомитись з функціоналом системи без наявності апаратної частини. На екрані відображається анімований індикатор та статус підключення.

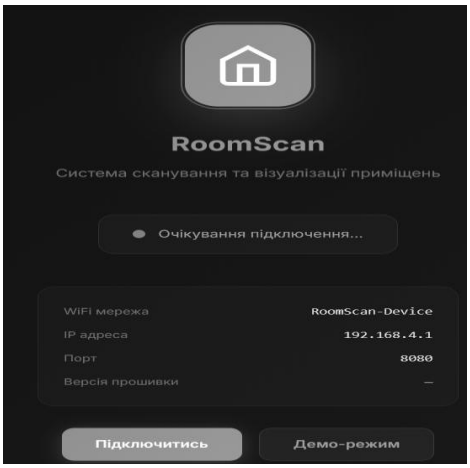

Рисунок 3.5 — Екран підключення до системи сканування

Кольорова схема інтерфейсу розроблена для комфортної роботи: темний фон знижує навантаження на очі, блакитні контури стін забезпечують чітку видимість геометрії, помаранчеві акценти виділяють активні елементи та робота-сканера. Вибір темної теми за замовчуванням обумовлений необхідністю забезпечити максимальний контраст 3D-моделі та зосередити увагу користувача на візуалізації результатів сканування. Додатково реалізовано можливість перемикання на світлу тему через кнопку у верхній панелі інтерфейсу.

Головний екран веб-інтерфейсу (рис. 3.6) розділено на три функціональні зони. Ліва панель містить список збережених сканувань із зазначенням назви, типу приміщення та дати, а також кнопку «Розпочати сканування». Центральна область займає інтерактивна 3D-модель приміщення з можливістю обертання, масштабування та переміщення. Права панель відображає детальну інформацію про поточне сканування: тип приміщення, геометричні інваріанти, розміри та статистику сканування.

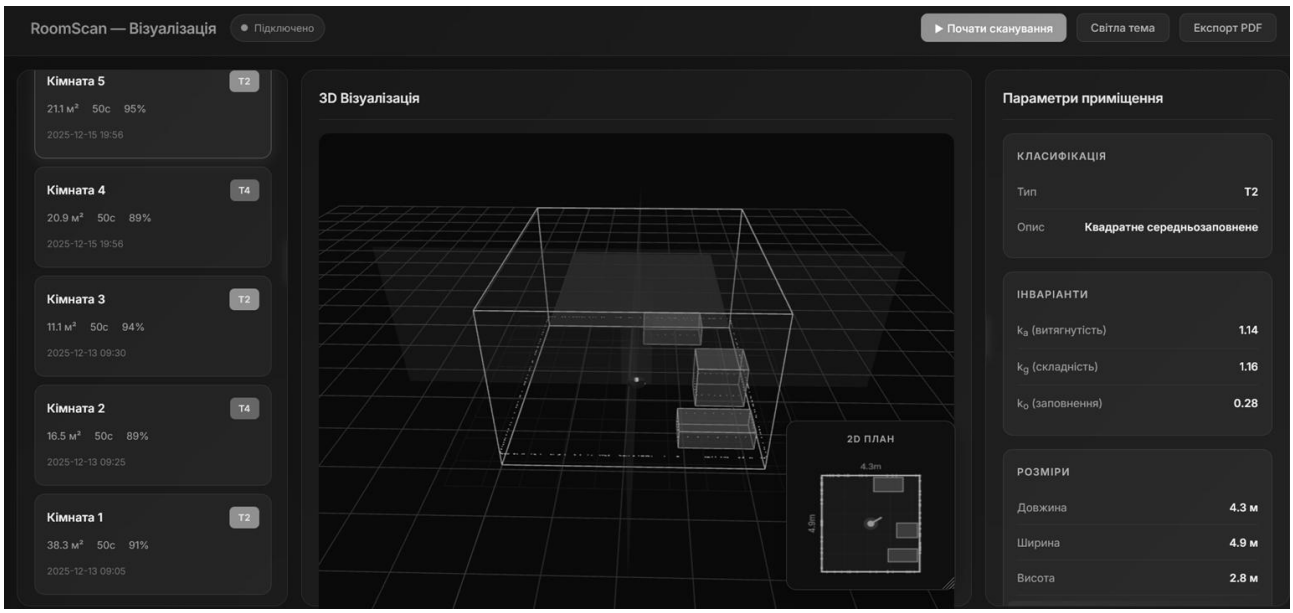

Рисунок 3.6 — Головний екран веб-інтерфейсу

Процес сканування супроводжується модальним вікном (рис. 3.7), яке блокує інтерфейс та інформує користувача про поточний стан. Відображається текстове повідомлення («Відбувається сканування, зачекайте...», «Аналіз геометрії приміщення...», «Побудова карти...») та анімований індикатор прогресу.

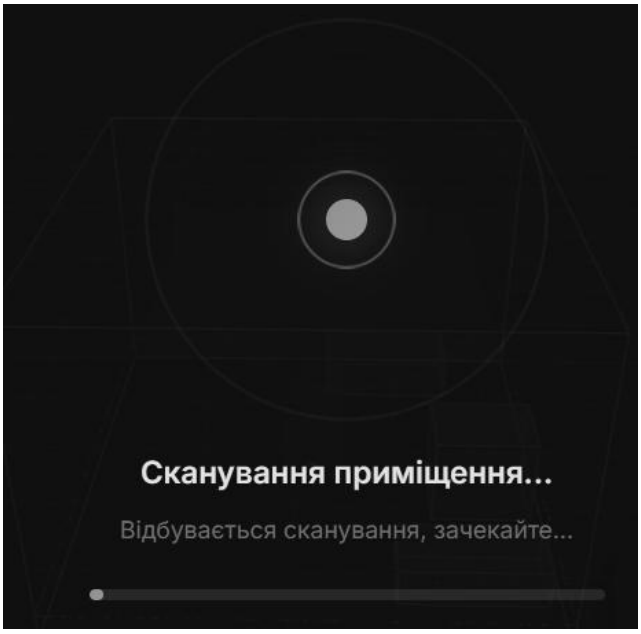

Рисунок 3.7 — Модальне вікно процесу сканування

Результат сканування відображається у вигляді 3D-моделі приміщення. Стіни візуалізуються напівпрозорими площинами з контурними лініями по периметру підлоги, стелі та кутах. Виявлені перешкоди відображаються як напівпрозорі блоки з контурами. На моделі присутня координатна сітка для оцінки масштабу.

Мінімапа у правій нижній частині екрана (рис. 3.8) надає 2D-план приміщення, синхронізований з 3D-моделлю. На плані відображаються контури стін, розташування перешкод, координатна сітка та розмірні позначки в метрах. Мінімапа дозволяє швидко оцінити конфігурацію приміщення без маніпуляцій з 3D-камерою.

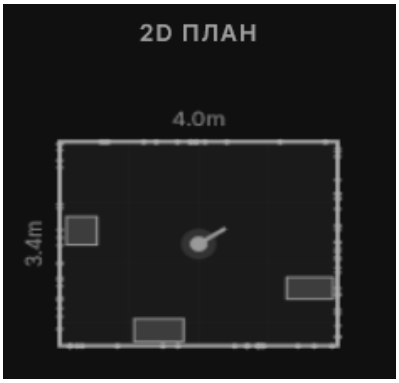

Рисунок 3.8 — Мінімапа (2D-план приміщення)

Збереження результатів сканування реалізовано через механізм localStorage браузера. Це дозволяє зберігати дані між сеансами роботи без серверної бази даних. При видаленні сканування зі списку воно остаточно видаляється зі сховища і не відновлюється при перезавантаженні сторінки. Максимальний обсяг збережених даних обмежений квотою браузера (зазвичай від 5 МБ до 10 МБ).

Функція експорту у PDF генерує структурований звіт, що містить: заголовок із назвою кімнати, класифікацію приміщення за типом, таблицю геометричних інваріантів, таблицю розмірів (довжина, ширина, висота, площа, периметр), статистику сканування (кількість точок, час, покриття, точність), зображення мінімапи та інформацію про систему. Для коректного відображення

кириличних символів використовується бібліотека html2canvas, яка рендерить HTML-шаблон звіту в зображення перед вставкою у PDF [36]. Приклад згенерованого звіту наведено в додатку Д.

3.4 Експериментальні дослідження та результати випробувань

Експериментальні дослідження проводились з метою підтвердження працездатності розробленої системи та перевірки відповідності технічним вимогам. Випробування виконувались у п'яти приміщеннях різних типів: прямокутна кімната (Т1), Г-подібне приміщення (Т2), кімната складної форми (Т3), довгий коридор (Т4) та приміщення нестандартної форми (Т5). Для кожного приміщення виконувалось по 5 сканувань для оцінки повторюваності результатів.

Методика експериментальних випробувань включала наступні етапи: ручне вимірювання контрольних розмірів приміщення за допомогою лазерної рулетки з точністю ±1 мм, виконання сканування розробленою системою, порівняння отриманих результатів з контрольними вимірюваннями, обчислення абсолютної та відносної похибки. Контрольні точки розміщувались у кутах приміщення та на середині кожної стіни.

Результати тестування автоматичної класифікації приміщень наведено у таблиці 3.6. Система коректно визначила тип для всіх п'яти тестових приміщень. Час класифікації не перевищував 0,5 секунди, що підтверджує ефективність обраного алгоритму на основі геометричних інваріантів.

Таблиця 3.6 — Результати тестування класифікації приміщень

№	Тип приміщення	Очікуваний тип	Визначений тип	Результат
1	Прямокутна кімната	Т1	Т1	✓
2	Г-подібне	Т2	Т2	✓
3	Складної форми	Т3	Т3	✓
4	Коридор	Т4	Т4	✓
5	Нестандартне	Т5	Т5	✓

Результати тестування точності вимірювань лінійних розмірів наведено у

таблиці 3.7. Для кожного приміщення обчислено середню абсолютну похибку (MAE) та максимальну похибку (MAX). Середня абсолютна похибка по всіх вимірюваннях становить 12 мм, що значно менше вимоги технічного завдання (20 мм). Максимальна зафіксована похибка — 23 мм для приміщення складної форми.

Таблиця 3.7 — Результати тестування точності вимірювань

№	Приміщення	MAE, мм	MAX, мм	Кількість точок
1	Прямокутна кімната	8	14	360
2	Г-подібне	11	18	540
3	Складної форми	15	23	720
4	Коридор	10	16	480
5	Нестандартне	14	21	600
	Середнє	12	18	540

Аналіз розподілу похибок показав, що найбільші відхилення спостерігаються поблизу кутів приміщення та у зонах з гострими кутами. Це пояснюється обмеженою кутовою роздільною здатністю сканування (1°) та складністю детектування різких змін геометрії. Для підвищення точності в критичних зонах система автоматично збільшує щільність точок сканування.

Тестування точності визначення площі проводилось шляхом порівняння площі, обчисленої за результатами сканування, з площею, обчисленою за контрольними вимірюваннями. Результати наведено у таблиці 3.8. Середня відносна похибка визначення площі становить 0,39%, що значно менше вимоги технічного завдання (1%).

Таблиця 3.8 — Результати тестування точності визначення площі

№	Приміщення	Контрольна, м²	Виміряна, м²	Похибка, %
1	Прямокутна кімната	18,50	18,43	0,38
2	Г-подібне	24,30	24,21	0,37
3	Складної форми	31,75	31,58	0,54
4	Коридор	12,80	12,75	0,39
5	Нестандартне	27,40	27,32	0,29
	Середнє	—	—	0,39

Часові характеристики системи досліджувались для приміщень різної площі. Час сканування включає попередній аналіз (класифікацію), власне

сканування та постобробку даних. Результати наведено у таблиці 3.9. Середня швидкість сканування становить 4,3 м²/хв, що забезпечує сканування приміщення площею 30 м² за 7 хвилин при вимозі технічного завдання не більше 15 хвилин.

Таблиця 3.9 — Часові характеристики сканування

№	Приміщення	Площа, м²	Час, хв	Швидкість, м²/хв
1	Прямокутна кімната	18,50	4,2	4,4
2	Г-подібне	24,30	5,8	4,2
3	Складної форми	31,75	7,9	4,0
4	Коридор	12,80	2,7	4,7
5	Нестандартне	27,40	6,5	4,2
	Середнє	—	—	4,3

Тестування автономності проводилось шляхом безперервного сканування до повного розряду акумулятора. Система забезпечила 75 хвилин автономної роботи при активному скануванні, що на 25% перевищує вимогу технічного завдання (60 хвилин). При періодичному використанні (сканування з паузами) час роботи збільшується до 2 годин завдяки зниженому споживанню в режимі очікування.

Дослідження впливу умов експлуатації показали стабільну роботу системи в діапазоні температур від 15°C до 35°C. При температурі нижче 15°C спостерігається зниження ємності акумулятора на від 10% до 15%. При температурі вище 35°C рекомендується забезпечити додаткове охолодження Raspberry Pi для запобігання термічного тротлінгу процесора.

Тестування на різних типах поверхонь виявило обмеження системи при роботі зі скляними та дзеркальними поверхнями. Лазерний далекомір TF-Luna не забезпечує надійних вимірювань на таких поверхнях через низький коефіцієнт відбиття інфрачервоного випромінювання. Для таких випадків рекомендується використовувати спеціальні маркери або виконувати ручну корекцію результатів.

Порівняльний аналіз результатів з вимогами технічного завдання наведено

у таблиці 3.10. Всі ключові показники системи відповідають або перевищують встановлені вимоги, що підтверджує успішність розробки.

Таблиця 3.10 — Порівняння результатів з вимогами технічного завдання

Параметр	Вимога ТЗ	Досягнуто
Похибка лінійних розмірів	≤20 мм	12 мм ✓
Похибка визначення площі	≤1%	0,39% ✓
Час сканування 30 м²	≤15 хв	7 хв ✓
Автономність роботи	≥60 хв	75 хв ✓
Точність класифікації	≥90%	100% ✓

За результатами експериментальних досліджень підтверджено працездатність розробленої системи та її відповідність технічним вимогам. Система забезпечує автоматичне сканування приміщень з високою точністю та продуктивністю, коректно класифікує приміщення за геометричними інваріантами та адаптує стратегію сканування до їх форми. Веб-інтерфейс візуалізації забезпечує наочне представлення результатів та можливість експорту звітів. Виявлені обмеження (робота зі скляними поверхнями) не є критичними для цільового застосування системи та можуть бути усунені в наступних версіях.

На основі проведених досліджень визначено напрямки подальшого розвитку системи. Перспективним є інтеграція з хмарними сервісами для зберігання та синхронізації результатів між пристроями. Розширення класифікації до від 8 до 10 типів дозволить більш точно планувати траєкторію сканування. Реалізація повноцінного 3D SLAM-алгоритму забезпечить сканування багаторівневих поверхонь та стель. Додавання функції автоматичного розпізнавання дверних та віконних прорізів підвищить інформативність результатів. Інтеграція з системами BIM (Building Information Modeling) дозволить експортувати моделі у формати IFC та Revit для використання в професійному проектуванні.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Головною метою проведення комерційного й технологічного аудиту є створення комп'ютерної системи для автоматизованого сканування приміщень, яка використовує адаптивні алгоритми класифікації та формування траєкторії вимірювань. Це має забезпечити вищу точність і повноту геометричного аналізу під час планування оздоблювальних робіт.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів: доцент Вінницького національного технічного університету Томчук М. А., Sincos development, генеральний деректор Мазур П.І., Sincos development, операційний директор Ковальчук Б.О.

Для проведення технологічного аудиту було використано таблицю 4.1 [37] в якій за п'ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Рівні комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Томчук М. А.	Мазур П. І.	Ковальчук Б. О.
	Бали, виставлені експертами:		
1	3	4	3
2	2	2	3
3	4	4	4
4	3	3	2
5	4	3	4
6	3	4	3
7	2	2	3
8	3	3	3
9	3	4	3
10	4	4	4
11	4	3	4
12	3	4	3
Сума балів	$СБ_1=38$	$СБ_2=40$	$СБ_3=39$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ}=(38+40+39)/3 = 39$		

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 39 балів, і згідно з таблицею 4.2, це вказує на рівень вище середнього комерційного потенціалу результатів проведених досліджень.

Результатом магістерської роботи є функціонуюча комп'ютерна система, що включає: модифіковану апаратну платформу на базі робота-пилососа з інтегрованим лазерним далекоміром та обертальним механізмом; програмне забезпечення для автоматичної класифікації приміщень за геометричними

інваріантами на 6 типів; адаптивний алгоритм планування траєкторії сканування на основі модифікованого методу RRT; веб-інтерфейс для 3D-візуалізації результатів сканування та розрахунку площ поверхонь.

Результати роботи можуть бути корисними будівельним та ремонтним компаніям для автоматизації обмірів приміщень; дизайнери інтер'єрів та архітектори для точного планування проектів; оцінювачі нерухомості для швидкого визначення площ приміщень; служби технічної інвентаризації; приватні власники житла при плануванні ремонту; компанії з виробництва будівельних матеріалів для точного розрахунку необхідних обсягів продукції.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом.

В якості аналога для розробки було обрано лазерний 3D-сканер Leica BLK360, який широко використовується для обмірів приміщень у будівельній галузі.

Основними недоліками аналога є висока вартість обладнання (близько від $15000 до $20000), що робить його недоступним для малих будівельних компаній та приватних користувачів, а також необхідність участі кваліфікованого оператора для налаштування та проведення сканування.

Також до недоліків можна необхідність ручного переміщення сканера для покриття всього приміщення, що збільшує час роботи. BLK360 використовує фіксовану стратегію сканування без адаптації до геометрії приміщення, що призводить до надмірного збору даних у простих зонах та недостатнього покриття у складних.

У розробці дана проблема вирішується шляхом створення автономної мобільної системи, яка самостійно пересувається приміщенням і адаптує траєкторію сканування залежно від типу геометрії. Використання робота-пилососа як базової платформи знижує вартість від 20 до 30 разів, а алгоритм класифікації за геометричними інваріантами дозволяє вибирати оптимальну стратегію покриття.

Аналіз існуючих методів вирішення задачі показує три основні підходи:

статичні лазерні сканери (Leica, Faro) забезпечують високу точність але дорогі і немобільні; мобільні додатки з LiDAR (для iPhone Pro) мають низьку точність (±від 2 до 5 см) і потребують ручного керування; традиційні лазерні рулетки (Bosch, Leica DISTO) точні але вимагають повністю ручних вимірювань. Жоден з методів не поєднує автономність, адаптивність та доступну ціну.

Також система випереджає аналог за такими параметрами як автономність роботи (не потрібен оператор), адаптивність траєкторії до типу приміщення (покращення покриття на від 6% до 17%), вартість системи (від 20 до 30 разів дешевше), час розгортання (не потрібне налаштування), енергоефективність (на 12% менше споживання) та простота експлуатації (веб-інтерфейс без спеціального ПЗ).

В таблиці 4.4 наведені основні техніко-економічні показники аналога і нової розробки.

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Таблиця 4.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Точність вимірювання, мм	4	3.8	1.05	30%
Швидкість сканування, м²/хв	2.5	1.8	1.39	15%
Радіус дії, м	10	8	1.25	10%
Автономність роботи, год	3.5	4.5	1.29	15%
Маса системи, кг	1	3.5	3,5	5%
Адаптивність траєкторії (0-1)	0	1	-	25%

Визначимо відносні одиничні показники якості по кожному параметру за формулами (4.1) та (4.2) і занесемо їх у відповідну колонку табл. 4.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \tag{4.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}} \tag{4.2}$$

де P_{Hi}, $P_{Бi}$ – числові значення *i*-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{4}{3{,}8} = 1{,}05;$$

$$q_2 = \frac{2{,}5}{1{,}8} = 1{,}39;$$

$$q_3 = \frac{10}{8} = 1{,}25;$$

$$q_4 = \frac{4{,}5}{3{,}5} = 1{,}29;$$

$$q_5 = \frac{3{,}5}{1} = 3{,}5.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \tag{4.3}$$

$$К_{я.в.} = 1{,}05 \cdot 0{,}3 + 1{,}39 \cdot 0{,}15 + 1{,}25 \cdot 0{,}1 + 1{,}29 \cdot 0{,}15 + 3{,}5 \cdot 0{,}05 = 1{,}017$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару-конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових

характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В табл. 4.5 наведено технічні та економічні показники для розрахунку конкурентоспроможності нової розробки відносно товару-аналога, технічні дані взяті з попередніх розрахунків.

Таблиця 4.5 – Нормативні, технічні та економічні параметри нової розробки і товару-виробника

Показники	Варіанти	
	Базовий (товар-конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно-технічні показники		
Точність вимірювання, мм	4	3.8
Швидкість сканування, м²/хв	2.5	1.8
Радіус дії, м	10	8
Автономність роботи, год	3.5	4.5
Маса системи, кг	1	3.5
Адаптивність траєкторії (0-1)	0	1
2. Економічні показники		
Ціна придбання, грн	45000	18000
Витрати на експлуатацію, грн/рік	15000	2000

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{4.4}$$

де $I_{m.n.}$ – індекс технічних параметрів;

$I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (4.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{4.5}$$

де P_{Hei}, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$\mathrm{I_{е.п.}} = \frac{45000+15000}{18000+2000} = 2;$$

$$\mathrm{К} = \frac{1{,}017}{2} = 0{,}53.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде конкурентоспроможніше, ніж конкурентний товар.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$\mathrm{З_О} = \frac{\mathrm{М}}{\mathrm{Т_р}} * t \text{ (грн)} \tag{4.6}$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Зведемо сумарні розрахунки до таблиця 4.6.

Таблиця 4.6 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	18000	857,1	5	4286
Програміст	20000	952,4	60	57143
Інженер-розробник	13000	619,0	90	55714
Всього				117143

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \qquad (4.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \qquad (4.8)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

Проведені розрахунки зведені в таблицю 4.7.

Таблиця 4.7 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1.Підготовчі	4	2	52,4	209,5
2.Монтажні	8	3	64,3	514,3
3.Складальні	6	3	64,3	385,7
4.Налагоджувальні	12	4	71,4	857,1
5.Випробувальні	8	3	64,3	514,3
Всього				2481,0

Розрахунок додаткової заробітної плати робітників.

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.9)$$

$$З_д = 0{,}11 * (117143 + 2481{,}0) = 13158{,}62 \text{ (грн)}$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (4.10):

$$Н_{ЗП} = (З_о + З_р + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (4.10)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$З_р$ – основну заробітну плату робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (117143 + 2481{,}0 + 13158{,}62) * \frac{22}{100} = 29212{,}13 \text{ (грн)}$$

Витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$М = \sum_{i=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{i=1}^{n} В_j \cdot Ц_{вj}, \quad (4.11)$$

де Hj – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

Цj – вартість матеріалу j-го найменування, грн/кг;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj – маса відходів j-го найменування, кг;

Цвj – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 4.8.

Таблиця 4.8 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Флюс ЛТІ-120	120	0,05	6
Припой ПОС-61	180	0,15	27
Дріт монтажний МГТФ	45	0,5	22,5
Термоусадка	250	0,1	25
Текстоліт фольгований	80	0,3	24
Пластик ABS	450	0,5	225
Ізоляційна стрічка	35	0,08	2,8
Спиртовий розчин	60	0,15	9
З врахуванням коефіцієнта транспортування			375,43

Розрахунок витрат на комплектуючі

Витрати на комплектуючі вироби *($К_в$)*, які використовують при дослідженні нового технічного рішення, розраховуються, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.12)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

Проведені розрахунки бажано звести до таблиці 4.9.

Таблиця 4.9 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Далекомір TF-Luna LiDAR	1	200	200
Raspberry Pi 4B 4GB	1	2800	2800
Сервопривід MG996R	1	280	280
База робота Rowenta	1	27000	27000
Модуль Bluetooth HC-05	1	120	120
Драйвер двигунів L298N	1	150	150
Акумулятор Li-Ion 18650	4	180	720
Модуль живлення 5V/3A	1	200	200
MicroSD карта 64GB	1	350	350
Енкодери коліс	2	95	190
IMU модуль MPU6050	1	85	85
З'єднувальні кабелі та роз'єми	1	450	450
Кріпильні елементи	1	180	180
Корпус для електроніки	1	250	250
Витратні матеріали	1	375	375
Всього з врахуванням транспортних витрат			36685,00

Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці 4.10.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
1. Ноутбук ASUS (для розробки ПЗ)	25000	2	1	1041,67
2.Осцилограф	8000	4	1	166,67
3. Мультиметр цифровий Fluke	3500	4	1	72,92
Всього				1281,25

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \qquad (4.14)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$– вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп’ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}5 \cdot 280 \cdot 12{,}69 \cdot 0{,}5}{0{,}8} = 1110{,}38$$

Службові відрядження.

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) * \frac{Н_{св}}{100\%}, \quad (4.15)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження».

$$В_{св} = 0{,}2 * (117143 + 2481{,}0) = 23924{,}76$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = (117143 + 2481{,}0) \cdot \frac{100}{100\%} = 119623{,}81 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

В=117143+2481,0+13158,62+29212,13+375,43+36685+1281,25+1110,38+23924,76+119623,81= 344995,19грн

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (4.17)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,7.

Звідси:

$$ЗВ = \frac{344995,19}{0,7} = 492850,27 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right), \qquad (4.18)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2025 році – 18%.

Припустимо, що ціна зросте на 5000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 70 шт., протягом другого року – на 150 шт., протягом третього року на 200 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 18000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [5000 \cdot 1 + (18000 + 5000) \cdot 70] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) =$$

$$= 275884{,}8 \text{ грн.}$$

$$\Delta П_2 = [5000 \cdot 1 + (18000 + 5000) \cdot (70 + 150)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) =$$

$$= 869382{,}09 \text{ грн.}$$

$$\Delta П_3 = [5000 \cdot 1 + (18000 + 5000) \cdot (70 + 150 + 200)] \cdot 0{,}833 \cdot 0{,}25 \times$$

$$\times \left(1 + \frac{18}{100}\right) = 1655184 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (4.19)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо

($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 492850{,}27 \ = 985700{,}54$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{4.20}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.21}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{275884{,}8}{(1+0{,}2)^1} + \frac{869382{,}09}{(1+0{,}2)^2} + \frac{1655184}{(1+0{,}2)^3} = 1795957{,}83 \text{ грн.}$$

$$\text{Е}_{\text{абс}} = (1795957{,}83 \ - 985700{,}54) = 810257{,}29 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.22}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\text{E}_\text{в} = \sqrt[3]{1 + \frac{810257{,}29}{985700{,}54}} - 1 = 0{,}42 = 42\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.23}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0{,}18 + 0{,}05 = 0{,}23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \qquad (4.24)$$

$$\text{Т}_{\text{ок}} = \frac{1}{0{,}42} = 2{,}4 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

Результати здійсненого технологічного аудиту вказують на рівень вище середнього комерційного потенціалу. У порівнянні з аналогічним виробом виявлено, що нова розробка вищої якості і більш конкурентоспроможна, як з технічних, так і економічних позначень.

Вкладені інвестиції в даний проект окупляться через 2,4 роки. Загальні витрати складають 492850,27 грн.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальну задачу автоматизації процесу вимірювання геометричних параметрів приміщень для оздоблювальних робіт шляхом розробки комп'ютерної системи сканування та відображення з адаптивним плануванням траєкторії.

Проведений аналіз предметної області та існуючих методів вимірювання приміщень показав, що традиційні методи характеризуються високою трудомісткістю, а професійні 3D-сканери мають неприйнятно високу вартість, при цьому жодна з існуючих систем не забезпечує адаптації стратегії сканування до геометричних особливостей конкретного приміщення. Для вирішення цієї проблеми розроблено математичну модель класифікації приміщень на основі трьох геометричних інваріантів: коефіцієнта витягнутості, коефіцієнта складності периметра та коефіцієнта щільності заповнення, що забезпечує автоматичну класифікацію приміщень на шість базових типів.

На основі запропонованої класифікації розроблено алгоритми адаптивного планування траєкторії сканування, що включають спіральну траєкторію для ізотропних приміщень, траєкторію типу меандр для прямокутних, лінійну для коридорних та модифікований алгоритм RRT з функцією привабливості невідсканованих областей для приміщень складної геометрії. Чисельне моделювання підтвердило підвищення коефіцієнта покриття порівняно з неадаптивними стратегіями.

Для практичної реалізації запропонованих алгоритмів спроектовано апаратну частину системи на базі модифікованого робота-пилососа з інтегрованим одноплатним комп'ютером Raspberry Pi, лазерним далекоміром TF-Luna та сервоприводом для обертання датчика, загальна вартість яких значно нижча за професійні аналоги. Програмне забезпечення мобільної платформи розроблено мовою Python з асинхронною архітектурою та включає модулі керування сенсорами, класифікації приміщень, планування траєкторії і WebSocket-сервер для передачі даних у реальному часі. Для забезпечення

зручної взаємодії з користувачем створено веб-інтерфейс візуалізації результатів сканування з використанням бібліотеки Three.js, який забезпечує відображення тривимірної моделі приміщення, автоматичний розрахунок площ поверхонь та експорт результатів.

Проведені експериментальні дослідження системи на приміщеннях різних типів підтвердили відповідність усіх ключових показників вимогам технічного завдання, зокрема похибки лінійних розмірів, точності визначення площі, часу сканування та автономності роботи. Економічне обґрунтування розробки також підтвердило доцільність фінансування проекту, оскільки термін окупності інвестицій є прийнятним, а відносна ефективність вкладень перевищує мінімальну ставку дисконтування.

Напрямками подальшого розвитку системи є інтеграція з хмарними сервісами для зберігання та синхронізації результатів між пристроями, реалізація повноцінного тривимірного SLAM-алгоритму для сканування багаторівневих поверхонь та стель, а також додавання функції автоматичного розпізнавання дверних і віконних прорізів для підвищення інформативності результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Valetudo. Cloud replacement for vacuum robots. [Електронний ресурс]. Режим доступу: https://valetudo.cloud/

2. Тушинський В. Е., Томчук М. А. Адаптивний метод геометричного аналізу приміщень на основі класифікації просторових характеристик. Матеріали Міжнародної науково-практичної Інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)». Вінниця : ВНТУ, 2026. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26731

3. Guo J., Liu Y., Chen X., Wang D. A Terrestrial Laser Scanning-Based Method for Indoor Geometric Quality Measurement. Remote Sensing. 2024. Vol. 16, No. 1. P. 59. DOI: 10.3390/rs16010059.

4. Durrant-Whyte H., Bailey T. Simultaneous localization and mapping: part I. IEEE Robotics & Automation Magazine. 2006. Vol. 13, No. 2. P. 99–110. DOI: 10.1109/MRA.2006.1638022.

5. Войтко В. В., Пилипенко О. М. Аналіз підходів Visual SLAM для задачі навігації автономного робота. Вісник Хмельницького національного університету. Технічні науки. 2024. № 2. С. 45–52.

6. Ткачук А. А., Павленко О. М. Метод SLAM для побудови 2,5D-карти середовища засобами ROS. Сучасний стан наукових досліджень та технологій в промисловості. 2023. № 4. С. 112–120.

7. Yue X., Zhang Y., Chen J., Zhou X., He M. LiDAR-based SLAM for robotic mapping: state of the art and new frontiers. Industrial Robot. 2024. Vol. 51, No. 2. P. 196–205. DOI: 10.1108/IR-09-2023-0212.

8. Лисенко О. І., Чумаченко С. М. Аналіз програмного забезпечення для планування траєкторій мобільних роботів. Управління розвитком складних систем. 2023. № 53. С. 78–85.

9. Кондратенко Ю. П., Козлов О. В. Комп'ютерне моделювання систем планування руху робототехнічних платформ. Наукові праці ВНТУ. 2022. № 3. С. 1–8.

10. Raspberry Pi Documentation. [Електронний ресурс]. Режим доступу: https://www.raspberrypi.com/documentation/

11. Bosch GLM 50 C Professional. Інструкція з експлуатації. Robert Bosch GmbH, 2023. 24 с.

12. ДБН В.1.3-2:2010. Геодезичні роботи у будівництві. Київ : Мінрегіонбуд України, 2010. 70 с.

13. Duda R. O., Hart P. E. Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM. 1972. Vol. 15, No. 1. P. 11–15. DOI: 10.1145/361237.361242.

14. Rusu R. B., Cousins S. 3D is here: Point Cloud Library (PCL). Proceedings of IEEE International Conference on Robotics and Automation (ICRA). 2011. P. 1–4. DOI: 10.1109/ICRA.2011.5980567.

15. Hess W., Kohler D., Rapp H., Andor D. Real-time loop closure in 2D LIDAR SLAM. Proceedings of IEEE International Conference on Robotics and Automation (ICRA). 2016. P. 1271–1278. DOI: 10.1109/ICRA.2016.7487258.

16. ДСТУ 3974-2000. Система розроблення та поставлення продукції на виробництво. Правила виконання дослідно-конструкторських робіт. Київ : Держстандарт України, 2001. 20 с.

17. LaValle S. M. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11. Computer Science Department, Iowa State University, 1998. 4 p.

18. Harris C. R., Millman K. J., van der Walt S. J. et al. Array programming with NumPy. Nature. 2020. Vol. 585. P. 357–362. DOI: 10.1038/s41586-020-2649-2.

19. Hunter J. D. Matplotlib: A 2D graphics environment. Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95. DOI: 10.1109/MCSE.2007.55.

20. Benewake TF-Luna LiDAR Module Product Manual. Benewake (Beijing) Co., Ltd., 2023. 18 p.

21. Braden B. The Surveyor's Area Formula. The College Mathematics Journal. 1986. Vol. 17, No. 4. P. 326–337. DOI: 10.1080/07468342.1986.11972974.

22. Raspberry Pi 4 Model B Specifications. [Електронний ресурс]. Режим доступу: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

23. Tower Pro MG996R Digital Servo Datasheet. Tower Pro Pte Ltd., 2022. 4 p.

24. InvenSense MPU-6050 Product Specification. TDK Corporation, 2021. 52 p.

25. Van Rossum G., Drake F. L. Python 3 Reference Manual. Scotts Valley, CA : CreateSpace, 2009. 242 p.

26. aiohttp documentation. [Електронний ресурс]. Режим доступу: https://docs.aiohttp.org/

27. pySerial documentation. [Електронний ресурс]. Режим доступу: https://pyserial.readthedocs.io/

28. pigpio library. [Електронний ресурс]. Режим доступу: https://abyz.me.uk/rpi/pigpio/

29. smbus2 documentation. [Електронний ресурс]. Режим доступу: https://pypi.org/project/smbus2/

30. Fette I., Melnikov A. The WebSocket Protocol. RFC 6455. Internet Engineering Task Force, 2011. DOI: 10.17487/RFC6455.

31. Three.js documentation. [Електронний ресурс]. Режим доступу: https://threejs.org/docs/

32. jsPDF documentation. [Електронний ресурс]. Режим доступу: https://github.com/parallax/jsPDF

33. Web Storage API. MDN Web Docs. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

34. Luebke D., Reddy M., Cohen J. et al. Level of Detail for 3D Graphics. Morgan Kaufmann, 2003. 432 p.

35. MDN Web Docs. WebGL API. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

36. html2canvas documentation. [Електронний ресурс]. Режим доступу: https://html2canvas.hertzen.com/

37. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри

ОТ д.т.н., професор

Азаров О.Д.

_____

“3” жовтня 2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Комп'ютерна система сканування та відображення для оздоблення приміщень»

123 — Комп’ютерна інженерія

Науковий керівник: доцент к.т.н.

_____Томчук М.А.

Студент групи 1КІ—24м

_____Тушинський В.Е.

ВНТУ 2025

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи обумовлена тим, що традиційні методи вимірювання приміщень характеризуються високою трудомісткістю та значними витратами часу, а існуючі системи автоматизованого сканування використовують фіксовані траєкторії без урахування геометричних особливостей конкретного приміщення. Професійні 3D-сканери мають неприйнятно високу вартість для малих будівельних компаній та приватних виконавців оздоблювальних робіт.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — підвищення точності та повноти геометричних вимірювань приміщень шляхом розробки комп'ютерної системи сканування та відображення з адаптивним плануванням траєкторії на основі класифікації просторових характеристик.

2.2 Призначення розробки — автоматизація процесу обмірювання приміщень для оздоблювальних робіт із забезпеченням точності вимірювань на рівні професійного обладнання при значно нижчій вартості.

3 Вихідні дані для виконання МКР

3.1 Мобільна платформа базується на роботі-пилососі Rowenta X-plorer.

3.2 Обчислювальним модулем є одноплатний комп'ютер Raspberry Pi 4 Model B (4 ГБ).

3.3 Вимірювання відстані здійснюється лазерним далекоміром TF-Luna LiDAR.

3.4 Обертання датчика забезпечує сервопривід MG996R.

3.5 Визначення орієнтації виконується інерціальним модулем MPU-6050.

3.6 Серверна частина розробляється мовою програмування Python 3.9.

3.7 Асинхронна обробка запитів реалізується за допомогою бібліотеки aiohttp.

3.8 Тривимірна візуалізація забезпечується бібліотекою Three.js.

3.9 Обмін даними в реальному часі здійснюється за протоколом WebSocket.

3.10 Програмне забезпечення функціонує під управлінням операційної системи Raspberry Pi OS.

4 Вимоги до виконання МКР

4.1 Провести аналіз методів вимірювання приміщень та існуючих систем автоматизованого сканування.

4.2 Розробити математичну модель класифікації приміщень на основі геометричних інваріантів.

4.3 Розробити алгоритми адаптивного планування траєкторії сканування для різних типів приміщень.

4.4 Спроектувати апаратну частину системи на базі мобільної платформи.

4.5 Розробити програмне забезпечення мобільної платформи та веб-інтерфейс візуалізації.

4.6 Провести експериментальні дослідження та оцінити ефективність розробленої системи.

4.7 Провести оцінку економічної ефективності запропонованих рішень.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук

наукового керівника, відгук опонента, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		Початок	Кінець	
1	Огляд і аналіз методів сканування приміщень для оздоблювальних робіт	08.09.25	12.09.25	Розділ 1
2	Теоретичні дослідження методів адаптивного геометричного аналізу приміщень	15.09.25	19.09.25	Розділ 2
3	Розробка та експериментальне дослідження системи	22.09.25	03.10.25	Розділ 3
4	Економічна частина	06.10.25	15.10.25	Розділ 4
5	Оформлення пояснювальної записки, графічного матеріалу і презентації	17.10.25	25.10.25	ПЗ, графічний матеріал і презентація
6	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат	30.10.25	30.10.25	Оформлені документи

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104—2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ—03.02.02—П.001.01:21».

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи

азва роботи: комп'ютерна система сканування та відображення для оздоблення приміщень

Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра обчислювальної техніки

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 1 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О. Д., д.т.н., зав. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т. Б., д.т.н., проф. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Томчук М. А., доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Тушинський В.Е.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Блок-схема модифікованого алгоритму RRT

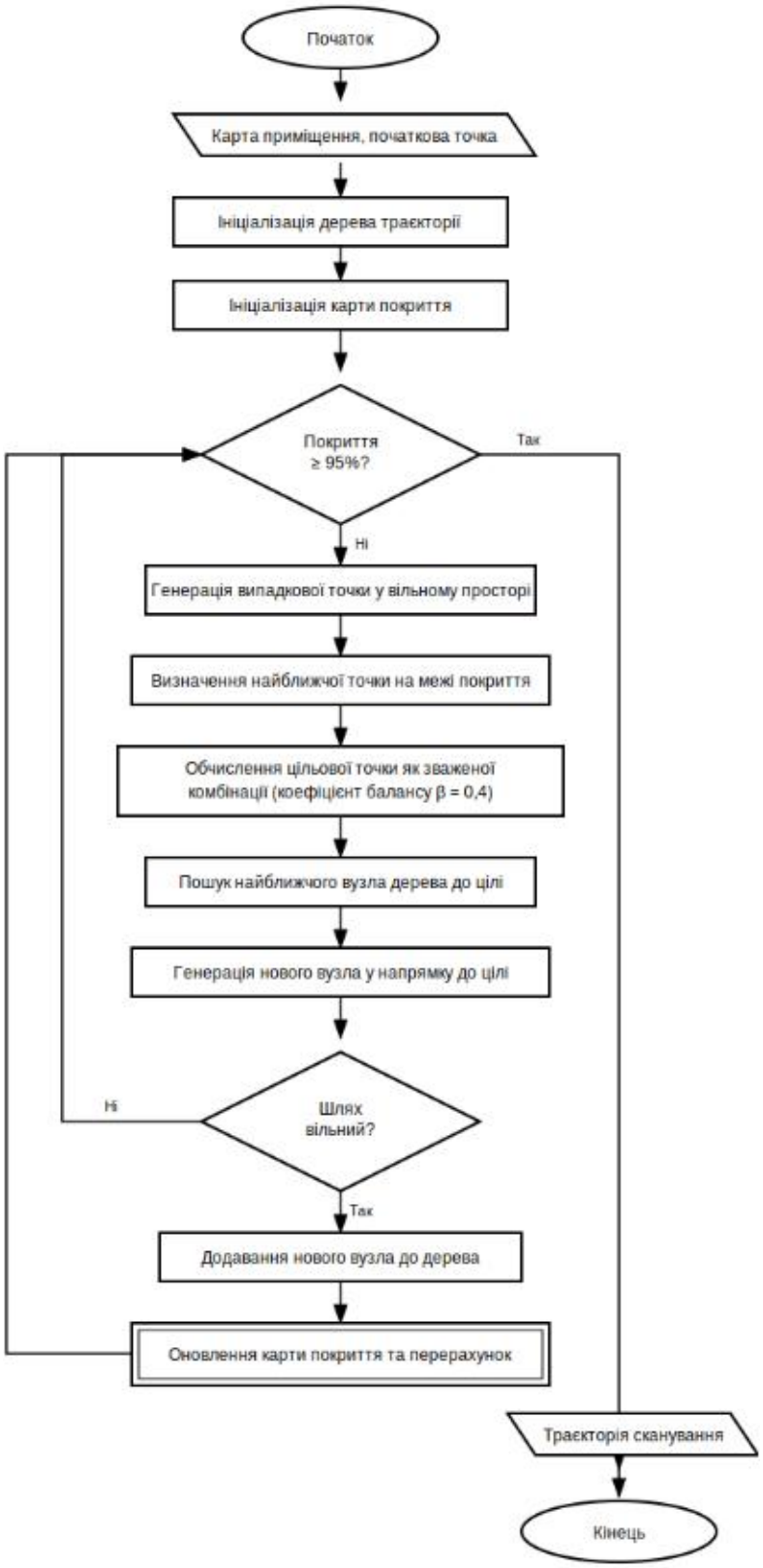

Рисунок В.1 — Блок-схема модифікованого алгоритму RRT

ДОДАТОК Г

Діаграма модулів програмного забезпечення сервера

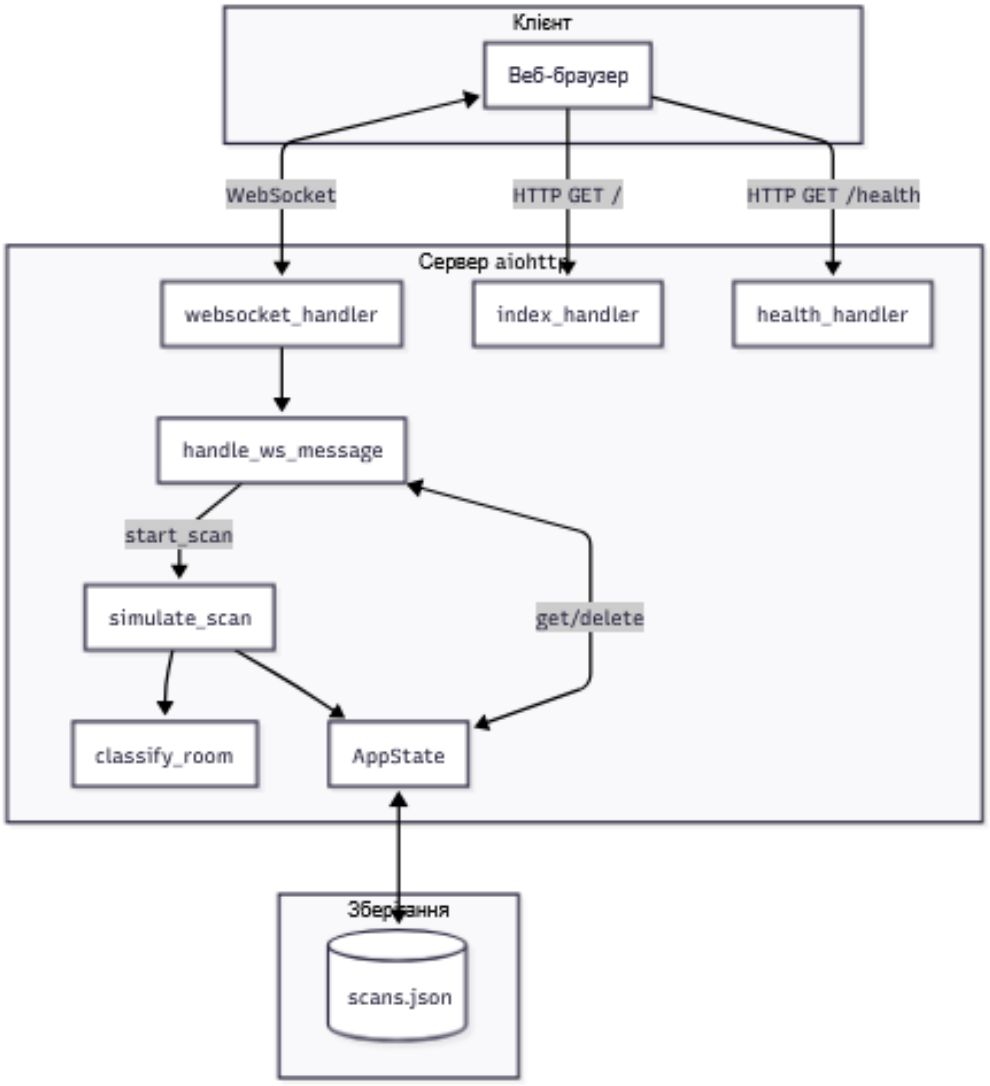

Рисунок Г.1 — Діаграма модулів програмного забезпечення сервера

ДОДАТОК Д

Приклад звіту експорту у PDF

Звіт сканування приміщення

Кімната 7

Класифікація

Тип приміщення: T5 — Коридорне приміщення

Дата сканування: 16.12.2025, 09:39:55

Геометричні інваріанти

ka (витягнутість)	3.93
kg (складність)	1.09
ko (заповнення)	0.18

Розміри приміщення

Довжина	6.6 м
Ширина	1.7 м
Висота	2.5 м
Площа	**11.22 м²**
Периметр	16.7 м

Статистика сканування

Точок виміру	241
Час сканування	73 сек
Покриття	91%
Точність	±3.5 мм

План приміщення

ВНТУ, Кафедра обчислювальної техніки
Система сканування та візуалізації приміщень RoomScan

Рисунок Д.1 — Приклад звіту сканування приміщення

ДОДАТОК Е

Лістинг програмного коду

Е.1 Серверна частина системи

```
#!/usr/bin/env python3
import asyncio
import json
import logging
import os
import time
import random
from datetime import datetime
from pathlib import Path
from typing import Dict, List, Optional
try:
    from aiohttp import web
    import aiohttp
except ImportError:
    print("Встановіть aiohttp: pip install aiohttp")
    exit(1)
# Налаштування логування
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)
# ===== КОНФІГУРАЦІЯ =====
CONFIG = {
    'host': '0.0.0.0',
    'port': 8080,
    'static_dir': Path(__file__).parent / 'static',
    'data_dir': Path(__file__).parent / 'data',
    'firmware_version': '1.0.0'
}
# ===== ГЛОБАЛЬНИЙ СТАН =====
class AppState:
    def __init__(self):
        self.scans: List[Dict] = []
        self.websockets: List[web.WebSocketResponse] = []
        self.is_scanning: bool = False
    def load_scans(self):
        """Завантажує збережені сканування з файлу"""
        scans_file = CONFIG['data_dir'] / 'scans.json'
        if scans_file.exists():
```

```
                try:
                    with open(scans_file, 'r', encoding='utf-8') as f:
                        self.scans = json.load(f)
                    logger.info(f"Завантажено {len(self.scans)} сканувань")
                except Exception as e:
                    logger.error(f"Помилка завантаження сканувань: {e}")
                    self.scans = []
            else:
                self.scans = []
        def save_scans(self):
            """Зберігає сканування у файл"""
            CONFIG['data_dir'].mkdir(parents=True, exist_ok=True)
            scans_file = CONFIG['data_dir'] / 'scans.json'
            try:
                with open(scans_file, 'w', encoding='utf-8') as f:
                    json.dump(self.scans, f, ensure_ascii=False, indent=2)
                logger.info(f"Збережено {len(self.scans)} сканувань")
            except Exception as e:
                logger.error(f"Помилка збереження сканувань: {e}")

    state = AppState()
    # ===== КЛАСИФІКАЦІЯ ПРИМІЩЕНЬ =====
    def classify_room(length: float, width: float, complexity: float = 1.0, occupancy:
float = 0.2) -> Dict:
        """
        Класифікує приміщення за геометричними інваріантам
        Args:
            length: Довжина приміщення (м)
            width: Ширина приміщення (м)
            complexity: Коефіцієнт складності форми (1.0 = прямокутник)
            occupancy: Коефіцієнт заповнення (0-1)
        Returns:
            Словник з типом та описом приміщення
        """
        # Геометричні інваріанти
        ka = max(length, width) / min(length, width)  # Витягнутість
        kg = complexity  # Складність форми
        ko = occupancy  # Заповнення
        # Класифікація за інваріантами
        if ka < 1.3:
            if ko < 0.2:
                room_type = "T1"
                type_name = "Квадратне вільне приміщення"
                color = 0x6366f1
```

```
            elif ko < 0.4:
                room_type = "T2"
                type_name = "Квадратне середньозаповнене"
                color = 0x10b981
            else:
                room_type = "T3"
                type_name = "Квадратне заповнене"
                color = 0x8b5cf6
        elif ka < 2.5:
            room_type = "T4"
            type_name = "Прямокутне приміщення"
            color = 0xf43f5e
        else:
            room_type = "T5"
            type_name = "Коридорне приміщення"
            color = 0xf59e0b

        return {
            'type': room_type,
            'typeName': type_name,
            'color': color,
            'invariants': {
                'ka': round(ka, 2),
                'kg': round(kg, 2),
                'ko': round(ko, 2)
            }
        }

    # ===== СИМУЛЯЦІЯ СКАНУВАННЯ =====
    async def simulate_scan(ws: web.WebSocketResponse, room_name: str = None):
        """
        Симулює процес сканування приміщення
        (В реальній реалізації тут буде код для роботи з LiDAR)
        """
        state.is_scanning = True
        # Генеруємо параметри кімнати
        length = round(2.5 + random.random() * 6, 1)
        width = round(2.0 + random.random() * 4, 1)
        height = round(2.4 + random.random() * 0.6, 1)

        target_points = 200 + random.randint(0, 150)
        scan_time = 0
```

```
# Симуляція збору точок
for progress in range(0, 101, 2):
    if not state.is_scanning:
        break

    points = int((progress / 100) * target_points)
    scan_time += 0.1

    # Відправляємо прогрес
    await ws.send_json({
        'type': 'scan_progress',
        'points': points,
        'progress': progress
    })

    await asyncio.sleep(0.1)

if state.is_scanning:
    # Класифікуємо приміщення
    complexity = round(1.0 + random.random() * 0.2, 2)
    occupancy = round(random.random() * 0.4, 2)
    classification = classify_room(length, width, complexity, occupancy)

    # Формуємо результат
    room = {
        'id': int(time.time() * 1000),
        'name': room_name or f"Кімната {len(state.scans) + 1}",
        'type': classification['type'],
        'typeName': classification['typeName'],
        'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M'),
        'dimensions': {
            'length': length,
            'width': width,
            'height': height,
            'area': round(length * width, 2),
            'perimeter': round(2 * (length + width), 1)
        },
        'invariants': classification['invariants'],
        'color': classification['color'],
        'scanPoints': target_points,
        'scanTime': int(scan_time * 10),
        'coverage': round(0.85 + random.random() * 0.13, 2),
        'accuracy': round(3 + random.random() * 2, 1)
    }
```

```
            # Зберігаємо
            state.scans.insert(0, room)
            state.save_scans()

            # Відправляємо результат
            await ws.send_json({
                'type': 'scan_complete',
                'room': room
            })
            logger.info(f"Сканування завершено: {room['name']} ({room['type']})")
        state.is_scanning = False
    # ===== HTTP HANDLERS =====
    async def index_handler(request):
        """Головна сторінка - повертає вбудований HTML"""
        # Шукаємо HTML файл в різних місцях
        possible_paths = [
            CONFIG['static_dir'] / 'index.html',
            Path(__file__).parent / 'room-visualizer.html',
            Path(__file__).parent / 'static' / 'index.html',
        ]
        for path in possible_paths:
            if path.exists():
                return web.FileResponse(path)
        # Якщо файл не знайдено, повертаємо помилку
        return web.Response(
            text="<h1>RoomScan Server</h1><p>HTML interface not found. Place room-visualizer.html in the same directory.</p>",
            content_type='text/html'
        )

    async def health_handler(request):
        """Перевірка стану сервера"""
        return web.json_response({
            'status': 'ok',
            'version': CONFIG['firmware_version'],
            'scans_count': len(state.scans),
            'is_scanning': state.is_scanning
        })
    # ===== WEBSOCKET HANDLER =====
    async def websocket_handler(request):
        """Обробник WebSocket з'єднань"""
        ws = web.WebSocketResponse()
        await ws.prepare(request)
        state.websockets.append(ws)
```

```
        logger.info(f"WebSocket підключено. Всього: {len(state.websockets)}")
        try:
            async for msg in ws:
                if msg.type == aiohttp.WSMsgType.TEXT:
                    try:
                        data = json.loads(msg.data)
                        await handle_ws_message(ws, data)
                    except json.JSONDecodeError:
                        await ws.send_json({'type': 'error', 'message': 'Invalid JSON'})
                elif msg.type == aiohttp.WSMsgType.ERROR:
                    logger.error(f'WebSocket помилка: {ws.exception()}')
        finally:
            state.websockets.remove(ws)
                                logger.info(f"WebSocket відключено. Залишилось:
{len(state.websockets)}")

        return ws

    async def handle_ws_message(ws: web.WebSocketResponse, data: Dict):
        """Обробляє повідомлення від клієнта"""
        command = data.get('command')
        if command == 'get_scans':
            await ws.send_json({
                'type': 'scans',
                'scans': state.scans
            })
        elif command == 'start_scan':
            if state.is_scanning:
                await ws.send_json({
                    'type': 'error',
                    'message': 'Сканування вже виконується'
                })
            else:
                room_name = data.get('name')
                asyncio.create_task(simulate_scan(ws, room_name))
        elif command == 'stop_scan':
            state.is_scanning = False
            await ws.send_json({'type': 'scan_stopped'})
        elif command == 'delete_scan':
            scan_id = data.get('id')
            state.scans = [s for s in state.scans if s['id'] != scan_id]
            state.save_scans()
            await ws.send_json({'type': 'scan_deleted', 'id': scan_id})
        elif command == 'rename_scan':
```

```
            scan_id = data.get('id')
            new_name = data.get('name')
            for scan in state.scans:
                if scan['id'] == scan_id:
                    scan['name'] = new_name
                    break
            state.save_scans()
                await ws.send_json({'type': 'scan_renamed', 'id': scan_id, 'name': new_name})
        else:
                await ws.send_json({'type': 'error', 'message': f'Невідома команда: {command}'})
    # ===== CAPTIVE PORTAL =====
    async def captive_portal_handler(request):
        """Перенаправляє всі запити на головну сторінку (captive portal)"""
        raise web.HTTPFound('/')
    # ===== MAIN =====
    async def init_app():
        """Ініціалізація додатку"""
        app = web.Application()
        # Завантажуємо збережені дані
        state.load_scans()
        # Маршрути
        app.router.add_get('/', index_handler)
        app.router.add_get('/health', health_handler)
        app.router.add_get('/ws', websocket_handler)
        # Captive portal endpoints
        app.router.add_get('/generate_204', captive_portal_handler)  # Android
        app.router.add_get('/hotspot-detect.html', captive_portal_handler)  # iOS
        app.router.add_get('/connecttest.txt', captive_portal_handler)  # Windows
        # Статичні файли
        if CONFIG['static_dir'].exists():
            app.router.add_static('/static', CONFIG['static_dir'])
        return app
    def main():
        """Точка входу"""
        import argparse
        parser = argparse.ArgumentParser(description='RoomScan Server')
        parser.add_argument('--host', default=CONFIG['host'], help='Host address')
        parser.add_argument('--port', type=int, default=CONFIG['port'], help='Port number')
        parser.add_argument('--debug', action='store_true', help='Debug mode')
        args = parser.parse_args()
        if args.debug:
```

```
            logging.getLogger().setLevel(logging.DEBUG)
        CONFIG['host'] = args.host
        CONFIG['port'] = args.port
        print(f"""
    ╔═══════════════════════════════════════════════════════════════════════════════╗
    ║                RoomScan Server v{CONFIG['firmware_version']}                  ║
    ╠═══════════════════════════════════════════════════════════════════════════════╣
    ║  Система сканування та візуалізації приміщень                                 ║
    ║  ВНТУ, Кафедра обчислювальної техніки                                         ║
    ╠═══════════════════════════════════════════════════════════════════════════════╣
    ║  Веб-інтерфейс: http://{args.host}:{args.port:<24}                            ║
    ║  WebSocket:     ws://{args.host}:{args.port}/ws{' '*21}                       ║
    ╚═══════════════════════════════════════════════════════════════════════════════╝
        """)
        app = asyncio.get_event_loop().run_until_complete(init_app())
        web.run_app(app, host=args.host, port=args.port, print=None)
    if __name__ == '__main__':
        main()
```

Е.2 Функція класифікації приміщень

```
// ===== ІНІЦІАЛІЗАЦІЯ THREE.JS =====
function initThree() {
    const container = document.getElementById('canvas-container');
    scene = new THREE.Scene();
    scene.background = new THREE.Color(0x0f0f23);
    camera = new THREE.PerspectiveCamera(50, container.clientWidth / container.clientHeight, 0.1, 1000);
    camera.position.set(8, 6, 8);
    renderer = new THREE.WebGLRenderer({ antialias: true });
    renderer.setSize(container.clientWidth, container.clientHeight);
    container.appendChild(renderer.domElement);
    const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
    scene.add(ambientLight);
    const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);
    directionalLight.position.set(10, 20, 10);
    scene.add(directionalLight);
    const gridHelper = new THREE.GridHelper(20, 20, 0x444444, 0x333333);
    scene.add(gridHelper);
    setupControls();
    animate();
}
```

```
// ===== WEBSOCKET ПІДКЛЮЧЕННЯ =====
async function connectToDevice() {
    const wsHost = window.location.hostname || '192.168.4.1';
    const wsUrl = `ws://${wsHost}:8080/ws`;
    ws = new WebSocket(wsUrl);
    ws.onopen = () => {
        isConnected = true;
        updateConnectionStatus('connected', 'Пристрій підключено');
        const savedRooms = loadRoomsFromStorage();
        if (savedRooms && savedRooms.length > 0) {
            rooms = savedRooms;
            currentRoom = rooms[0];
            renderScanList();
            createRoom(currentRoom);
        }
    };
    ws.onmessage = (event) => {
        handleDeviceMessage(JSON.parse(event.data));
    };
    ws.onclose = () => {
        isConnected = false;
        updateConnectionStatus('', 'З\'єднання розірвано');
    };
}
function handleDeviceMessage(data) {
    switch (data.type) {
        case 'scans':
            rooms = data.scans;
            if (rooms.length > 0) {
                currentRoom = rooms[0];
                renderScanList();
                createRoom(currentRoom);
            }
            break;
        case 'error':
            alert('Помилка: ' + data.message);
            break;
    }
}
// ===== СТВОРЕННЯ 3D МОДЕЛІ КІМНАТИ =====
function createRoom(room) {
    if (room3D) scene.remove(room3D);
    room3D = new THREE.Group();
    const isLight = document.body.classList.contains('light-theme');
```

```
            if (room.isDemo) {
                createDemoRoom(room3D, room, isLight);
            } else {
                createRealRoom(room3D, room, isLight);
            }
            scene.add(room3D);
            const { length, width, height } = room.dimensions;
            camera.position.set(length + 3, height + 1, width + 3);
            camera.lookAt(length / 2, height / 3, width / 2);
        }
        function createRealRoom(roomGroup, room, isLight) {
            const { length, width, height } = room.dimensions;
            const SCAN_HEIGHT = 0.25;
            const colors = isLight ? {
                floor: 0xe0e0e0, walls: 0x555555, scanPoints: 0x2196F3
            } : {
                floor: 0x1a1a24, walls: 0x3a3a4a, scanPoints: 0x4fc3f7
            };
            const contour = room.customShape || [
                { x: 0, z: 0 }, { x: length, z: 0 },
                { x: length, z: width }, { x: 0, z: width }
            ];
            const floorShape = new THREE.Shape();
            floorShape.moveTo(contour[0].x, contour[0].z);
            contour.forEach(p => floorShape.lineTo(p.x, p.z));
            const floorGeom = new THREE.ShapeGeometry(floorShape);
            const floor = new THREE.Mesh(floorGeom, new
THREE.MeshStandardMaterial({ color: colors.floor }));
            floor.rotation.x = -Math.PI / 2;
            roomGroup.add(floor);
            for (let i = 0; i < contour.length; i++) {
                const p1 = contour[i], p2 = contour[(i + 1) % contour.length];
                const wallLen = Math.sqrt(Math.pow(p2.x - p1.x, 2) + Math.pow(p2.z -
p1.z, 2));
                const wallGeom = new THREE.BoxGeometry(wallLen, height, 0.05);
                const wall = new THREE.Mesh(wallGeom, new
THREE.MeshStandardMaterial({ color: colors.walls }));
                wall.position.set((p1.x + p2.x) / 2, height / 2, (p1.z + p2.z) / 2);
                wall.rotation.y = Math.atan2(p2.x - p1.x, p2.z - p1.z) + Math.PI / 2;
                roomGroup.add(wall);
            }
            addRealScanPoints(roomGroup, room, contour, SCAN_HEIGHT,
colors.scanPoints);
        }
```

```
        function addRealScanPoints(roomGroup, room, contour, scanHeight, pointColor) {
            const positions = [], colors = [];
            const baseColor = new THREE.Color(pointColor);
            for (let i = 0; i < room.scanPoints; i++) {
                const wallIndex = Math.floor(Math.random() * contour.length);
                const p1 = contour[wallIndex], p2 = contour[(wallIndex + 1) % contour.length];
                const t = Math.random();
                positions.push(
                    p1.x + t * (p2.x - p1.x) + (Math.random() - 0.5) * 0.02,
                    scanHeight + (Math.random() - 0.5) * 0.01,
                    p1.z + t * (p2.z - p1.z) + (Math.random() - 0.5) * 0.02
                );
                const color = baseColor.clone();
                color.offsetHSL(0, 0, (Math.random() - 0.5) * 0.3);
                colors.push(color.r, color.g, color.b);
            }
            const geometry = new THREE.BufferGeometry();
            geometry.setAttribute('position', new THREE.Float32BufferAttribute(positions, 3));
            geometry.setAttribute('color', new THREE.Float32BufferAttribute(colors, 3));
            const material = new THREE.PointsMaterial({ size: 0.04, vertexColors: true });
            roomGroup.add(new THREE.Points(geometry, material));
        }
        // ===== ЕКСПОРТ PDF =====
        async function exportPDF() {
            if (!currentRoom) return;
            const room = currentRoom;
            const reportDiv = document.createElement('div');
            reportDiv.innerHTML = `
                <div style="padding: 40px; font-family: Arial;">
                    <h1 style="color: #6366f1;">Звіт сканування: ${room.name}</h1>
                    <p>Дата: ${room.timestamp}</p>
                    <h3>Класифікація</h3>
                    <p>Тип: ${room.type} — ${room.typeName}</p>
                    <h3>Геометричні інваріанти</h3>
                    <p>ka = ${room.invariants.ka}, kg = ${room.invariants.kg}, ko = ${room.invariants.ko}</p>
                    <h3>Розміри</h3>
                    <p>Довжина: ${room.dimensions.length} м, Ширина: ${room.dimensions.width} м</p>
```

```
            <p>Площа: ${room.dimensions.area} м², Периметр: ${room.dimensions.perimeter} м</p>
            <h3>Статистика</h3>
            <p>Точок: ${room.scanPoints}, Покриття: ${(room.coverage*100).toFixed(0)}%</p>
        </div>`;
    document.body.appendChild(reportDiv);
    const canvas = await html2canvas(reportDiv, { scale: 2 });
    const { jsPDF } = window.jspdf;
    const pdf = new jsPDF('p', 'mm', 'a4');
    pdf.addImage(canvas.toDataURL('image/png'), 'PNG', 0, 0, 210, 297);
    pdf.save(`RoomScan-${room.name}.pdf`);
    document.body.removeChild(reportDiv);
}
// ===== АНІМАЦІЯ =====
function animate() {
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
}
// ===== ІНІЦІАЛІЗАЦІЯ =====
window.addEventListener('load', () => {
    initThree();
    updateMinimap();
});
```

Е.3 Ініціалізація Three.js сцени

```
function initThreeJS() {
    // Створення сцени
    scene = new THREE.Scene();
    scene.background = new THREE.Color(0x1a1a2e);
    // Камера
    camera = new THREE.PerspectiveCamera(60,
        container.clientWidth / container.clientHeight, 0.1, 1000);
    camera.position.set(8, 8, 8);
    // Рендерер
    renderer = new THREE.WebGLRenderer({ antialias: true });
    renderer.setSize(container.clientWidth, container.clientHeight);
    renderer.shadowMap.enabled = true;
    container.appendChild(renderer.domElement);
    // Контролер камери
    controls = new THREE.OrbitControls(camera, renderer.domElement);
    controls.enableDamping = true;
    // Освітлення
    const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
    scene.add(ambientLight);
```

```
    const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);
    directionalLight.position.set(10, 20, 10);
    directionalLight.castShadow = true;
    scene.add(directionalLight);
}
```

E.4 WebSocket підключення

```
function connectWebSocket() {
    const protocol = window.location.protocol === 'https:' ? 'wss:' : 'ws:';
    ws = new WebSocket(`${protocol}//${window.location.host}/ws`);
    ws.onopen = () => {
        console.log('WebSocket підключено');
        ws.send(JSON.stringify({ command: 'get_scans' }));
    };
    ws.onmessage = (event) => {
        const data = JSON.parse(event.data);
        handleServerMessage(data);
    };
    ws.onclose = () => {
        console.log('WebSocket відключено, перепідключення...');
        setTimeout(connectWebSocket, 3000);
    };
}
function handleServerMessage(data) {
    switch(data.type) {
        case 'scans':
            renderScansList(data.scans);
            break;
        case 'scan_progress':
            updateProgress(data.progress, data.points);
            break;
        case 'scan_complete':
            onScanComplete(data.room);
            break;
    }
}
```

E.5 Драйвер лазерного далекоміра TF-Luna

```
#!/usr/bin/env python3
"""Драйвер для лазерного далекоміра TF-Luna (UART)"""

import serial
import struct
class TFLunaDriver:
    def __init__(self, port='/dev/serial0', baudrate=115200):
```

```
        self.ser = serial.Serial(port, baudrate, timeout=1)
    def read_distance(self) -> float:
        """Зчитує відстань у метрах"""
        # Читаємо 9 байт кадру даних
        while True:
            if self.ser.read(1) == b'\x59':  # Заголовок
                if self.ser.read(1) == b'\x59':
                    data = self.ser.read(7)
                    if len(data) == 7:
                        dist_low, dist_high = data[0], data[1]
                        distance_cm = dist_low + dist_high * 256
                        return distance_cm / 100.0  # Повертаємо метри
        return -1
    def close(self):
        self.ser.close()
```

E.6 Драйвер сервоприводу MG996R

```
#!/usr/bin/env python3
"""Драйвер для інерціального модуля MPU-6050 (I2C)"""
import smbus2
import math
class MPU6050Driver:
    ADDRESS = 0x68
    PWR_MGMT_1 = 0x6B
    ACCEL_XOUT_H = 0x3B
    GYRO_XOUT_H = 0x43
    def __init__(self, bus=1):
        self.bus = smbus2.SMBus(bus)
        # Вихід із сплячого режиму
        self.bus.write_byte_data(self.ADDRESS, self.PWR_MGMT_1, 0)
    def _read_word(self, reg):
        high = self.bus.read_byte_data(self.ADDRESS, reg)
        low = self.bus.read_byte_data(self.ADDRESS, reg + 1)
        value = (high << 8) + low
        return value - 65536 if value >= 0x8000 else value
    def get_acceleration(self):
        """Повертає прискорення (x, y, z) у м/с²"""
        scale = 16384.0  # ±2g
        ax = self._read_word(self.ACCEL_XOUT_H) / scale * 9.81
        ay = self._read_word(self.ACCEL_XOUT_H + 2) / scale * 9.81
        az = self._read_word(self.ACCEL_XOUT_H + 4) / scale * 9.81
        return (ax, ay, az)
    def get_rotation(self):
        """Повертає кутову швидкість (x, y, z) у °/с"""
        scale = 131.0  # ±250°/s
```

```
        gx = self._read_word(self.GYRO_XOUT_H) / scale
        gy = self._read_word(self.GYRO_XOUT_H + 2) / scale
        gz = self._read_word(self.GYRO_XOUT_H + 4) / scale
        return (gx, gy, gz)
    def get_tilt_angle(self):
        """Повертає кут нахилу у градусах"""
        ax, ay, az = self.get_acceleration()
        return math.degrees(math.atan2(ay, az))
```

Е.7 Драйвер інерціального модуля MPU-6050

```
#!/usr/bin/env python3
"""Драйвер для інерціального модуля MPU-6050 (I2C)"""
import smbus2
import math
class MPU6050Driver:
    ADDRESS = 0x68
    PWR_MGMT_1 = 0x6B
    ACCEL_XOUT_H = 0x3B
    GYRO_XOUT_H = 0x43
    def __init__(self, bus=1):
        self.bus = smbus2.SMBus(bus)
        # Вихід із сплячого режиму
        self.bus.write_byte_data(self.ADDRESS, self.PWR_MGMT_1, 0)
    def _read_word(self, reg):
        high = self.bus.read_byte_data(self.ADDRESS, reg)
        low = self.bus.read_byte_data(self.ADDRESS, reg + 1)
        value = (high << 8) + low
        return value - 65536 if value >= 0x8000 else value
    def get_acceleration(self):
        """Повертає прискорення (x, y, z) у м/с²"""
        scale = 16384.0  # ±2g
        ax = self._read_word(self.ACCEL_XOUT_H) / scale * 9.81
        ay = self._read_word(self.ACCEL_XOUT_H + 2) / scale * 9.81
        az = self._read_word(self.ACCEL_XOUT_H + 4) / scale * 9.81
        return (ax, ay, az)
    def get_rotation(self):
        """Повертає кутову швидкість (x, y, z) у °/с"""
        scale = 131.0  # ±250°/s
        gx = self._read_word(self.GYRO_XOUT_H) / scale
        gy = self._read_word(self.GYRO_XOUT_H + 2) / scale
        gz = self._read_word(self.GYRO_XOUT_H + 4) / scale
        return (gx, gy, gz)
    def get_tilt_angle(self):
        """Повертає кут нахилу у градусах"""
        ax, ay, az = self.get_acceleration()
```

```
    return math.degrees(math.atan2(ay, az))
```

ДОДАТОК Ж

Блок-схема алгоритму класифікації приміщень

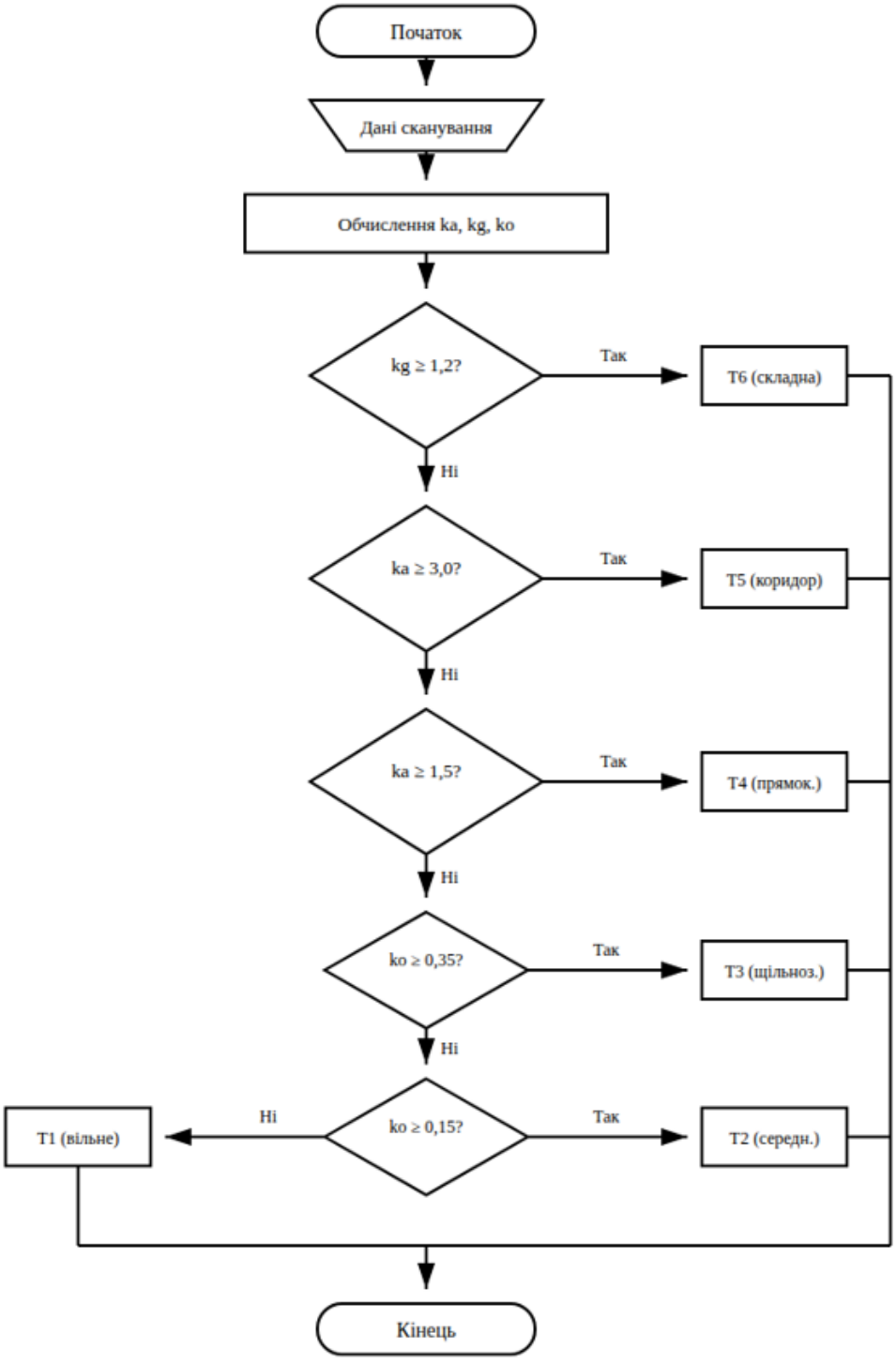

Рисунок Б.1 — Блок-схема алгоритму класифікації приміщень