Кафедра *інформаційно-вимірювальних технологій*

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської кваліфікаційної роботи на тему

Аналіз ефективності алгоритмів Simultaneous Localization and Mapping для навігації роботів

Студент *152, МТ-4* Бучук Д.О

Керівник БКР / / к.т.н., доцент Сегеда О.В.

Консультанти / к.е.н., доцент Рачинська Г.В./

Завідувач кафедри // д.т.н. професор ІВТ Бубела Т.З.

“___” _____ 2025

Національний університет «Львівська політехніка»

(назва вищого навчального закладу)

Інститут ІКТА Кафедра ІВТ

Спеціальність *152 Метрологія та інформаційно-вимірювальна техніка*

«ЗАТВЕРДЖУЮ»

Завідувач кафедри ІВТ

Т.З.Бубела

«__» _____ 2025 р.

ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу студентові
Бучук Данило Олексійович
(прізвище, ім'я по батькові)

1. Тема проекту Аналіз ефективності алгоритмів Simultaneous Localization and Mapping для навігації роботів
затверджена наказом по університету від 8 квітня 2025 р. № 1282-4-08
2. Термін подання студентом закінченого проекту 16 червня 2025 р.
3. Вихідні дані до проекту: Основні типи SLAM-алгоритмів (GMapping, Cartographer, ORB-SLAM, RTAB-Map); Показники оцінки ефективності: ATE, RPE, RMSE, FPS, затримка, ресурсомісткість
4. Зміст розрахунково – пояснювальної записки (перелік питань, що їх треба розробити):
1. Огляд сучасних алгоритмів SLAM
2. Аналіз ефективності обраних алгоритмів
3. Експерементальне дослідження
4. Перспективи та удосконалення

5. Перелік графічного матеріалу
Презентація в Power Point

6. Консультанти, із зазначенням розділів БКР, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічна частина	к.е.н. Рачинська Галина Василівна		

7. Дата отримання завдання 8.04.2025р

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	*Назва етапів бакалаврської кваліфікаційної роботи*	*Термін виконання етапів БКР*	*Примітки*
1	*Отримання завдання*	*8.04.2025р.*	
2	*Аналітичний огляд існуючих алгоритмів SLAM*	*9.04.2025р.- 20.04.2025р.*	
3	*Проведення експерементів, аналіз отриманих данних.*	*21.04.2025р.- 11.05.2025р.*	
4	*Економічне обґрунтування проектних рішень.*	*12.05.2025р.- 25.05.2025р.*	
5	*Оформлення пояснювальної записки та графічної частини. Опрацювання висновків та рекомендацій*	*26.05.2025р.- 8.06.2025р.*	
6	*Перевірка на плагіат*	*9.06.2025р.- 15.06.2025р.*	
7	*Подання дипломної роботи до попереднього захисту*	*16.06.2025р.- 20.06.2025р.*	
8	*Захист дипломної роботи*	*25.06.2025р.*	

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

АНОТАЦІЯ

У цій бакалаврській кваліфікаційній роботі проведено комплексний аналіз ефективності алгоритмів одночасної локалізації та побудови карти (SLAM), які застосовуються в автономних робототехнічних системах. Дослідження зосереджено на трьох поширених реалізаціях: GMapping, Cartographer та ORB-SLAM, що репрезентують різні підходи до побудови карти середовища – лідарний, графовий і візуальний відповідно.

У роботі здійснено огляд сучасного стану технологій SLAM, проаналізовано принципи роботи кожного алгоритму, визначено їхні сильні й слабкі сторони на основі порівняльного експериментального тестування у симуляційному середовищі Gazebo. Як критерії оцінювання використано показники точності (ATE, RPE), якості карти (RMSE), продуктивності (FPS, затримка), а також ресурсомісткості. Результати дослідження дозволили сформулювати рекомендації щодо вибору алгоритмів SLAM залежно від типу середовища та вимог до системи.

ANNOTATION

This bachelor's qualification work presents a comprehensive analysis of the efficiency of Simultaneous Localization and Mapping (SLAM) algorithms used in autonomous robotic systems. The study focuses on three widely used implementations: GMapping, Cartographer, and ORB-SLAM, which represent different approaches to environment mapping — lidar-based, graph-based, and visual, respectively.

The work includes a review of the current state of SLAM technologies, an analysis of the operational principles of each algorithm, and an identification of their strengths and weaknesses based on comparative experimental testing in the Gazebo simulation environment. The evaluation criteria included accuracy (ATE, RPE), map quality (RMSE), performance (FPS, latency), and resource consumption.

ВСТУП 8
РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ АЛГОРИТМІВ SLAM 11
1.1 Призначення та завдання SLAM у робототехніці 11
1.2 Основні типи алгоритмів SLAM 13
1.3 Порівняння методів на основі літературного огляду 19
1.4 Висновок 21
РОЗДІЛ 2. АНАЛІЗ ЕФЕКТИВНОСТІ ОБРАНИХ АЛГОРИТМІВ 23
2.1 Алгоритм GMapping 23
2.1.1 Алгоритм Cartographer 26
2.1.2 Алгоритм RTAB-Map 29
2.2 Критерії оцінювання продуктивності SLAM 33
2.3 Інструменти тестування та середовища моделювання 34
2.4 Аналіз алгоритмів (GMapping, Cartographer, ORB-SLAM) 36
2.5 Висновок 41
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ 42
3.1 Побудова симуляційного середовища 43
3.2 Випробування алгоритмів у різних сценаріях 44
3.3 Порівняння точності локалізації та якості мапи 45
РОЗДІЛ 4. ПЕРСПЕКТИВИ ТА УДОСКОНАЛЕННЯ 47
4.1 Технологічні виклики та тенденції майбутнього 47
4.2 Інтеграція глибокого навчання в SLAM 51
4.3 Гібридні підходи 52
4.4 Обмін даними і синхронізація в гібридних системах 54
РОЗДІЛ 5. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 58
5.1 Розрахунок витрат на виконання БКР 59
5.1.1 Розрахунок витрат на оплату праці 59
5.1.2 Відрахування на соціальні заходи 61
5.1.3 Розрахунок витрат на матеріали 61

5.1.4 Витрати на використання комп’ютерної техніки 63
5.1.5 Інші витрати 65
5.1.6 Накладні витрати 65
5.1.7 Калькуляція кошторисної вартості виконання БКР 66
5.1.8 Розрахунок договірної ціни та прибутку БКР 66
5.1.9 Оцінка наукової та науково-технічної результативності БКР 67
5.2 Висновки до розділу 71
ВИСНОВОК 72
Список використаних джерел 74

ВСТУП

У нашому сучасному світі який стрімко розвивається у всіх сферах технологій, важливим елементом є автономні системи та робототехнічні технології. Нові горизонти які відкривають нам ці сфери неможливо переоцінити автоматизації, транспорту, облуговування, дослідницька діяльність, охорона довкілля, тощо.

Робототехніка деталі частіше виходить за межі заводів та фабрик проникаючи у повсякденне життя звичайної людини, облекшуючи рутинні справи. Гарними прикладами можуть виступати роботи домашні сервіси (робот-пилосос, розумний дім). Також дрони облекшуть і робочі процесси, гарним прикладом може виступати сучасна картографія де без дронів та супутникових технологій не обійтися. Але основним завданням дня будь-якої мобільної автономної платформи є здатність орієнтуватися в навколишньому середовищі з мінімальним втручанням людини, уникаючи перешкоди, будуючи мапи оточення в реальному часі та орієнтуватися у просторі розуміючи власне положення у ньому. Саме ці задачі вирішує технологія SLAM (Simultaneous Localization and Mapping).

SLAM — це фундаментальний підхід у робототехніці, цей алгоритм дозволяє будувати карту середовища в реальному часі та водночас визначати положення робота у цьому середовищі відносно побудованої карти, використовуючи лише дані отримані від сенсорів. На відміну від вже традиційних методів навігаційних систем які здебільшого покладаються на GPS або вже заздалегідь готові та завантажені карти, SLAM дозволяє роботу працювати в нових умовах, або в умовах з високою динамічністю. Завдяки технології SLAM стало можливим створення дронів, які можуть працювати та орієнтуватися на місцевості без втручання людини.

SLAM є дуже важливою технологією яка дозволяє автоматизувати та спросити орієнтування автономного агента у просторі мінімізуючи втручання

людини в цей процес, але її ефективність сильно залежить від вибраного алгоритму, типу наявних сенсорів, умов середовищ в яких перебуває дрон та ресурсів обчислювальної системи. Існує дуже велика кількість реалізації технології SLAM, кожна з яких має свої переваги та недоліки і може бути використана в різних умовах та середовищах. Наприклад, деякі алгоритми демострують дуже високу точність локалізації, але натомість потребують великих обчислювальних потужностей та першокласну якість сенсорів. Інакші алгоритми будуть якісно забезпечувати роботу в режимі реального часу, але будуть суттєво знижувати якість мапи та будуть накопичувати велику кількість помилок та неточностей. Якщо брати у приклад реальні проекти, то правильний вибір реалізації технології SLAM визначає ефективність усієї системи навігації.

В цій дипломній роботі буде проведено порівняльний аналіз найпопулярніших алгоритмів SLAM на даний момент часу в усьому світі. Розглянемо найпопулярніші варіанти реалізацій, такі як GMapping, Cartographer та ORB-SLAM2, що представляють та демонструють різні класи підходів до технології SLAM – лідарні, графові та візуальні.

Актуальність цієї теми зумовлена дуже стрімким ростом попиту на автономні рішення у сферах транспорту, логістики, рятувальних операціях та дослідженнях навколишнього середовища. Такі автономні агенти часто змушені функціонувати в умовах підвищеної складності де оточення часто змінюється дуже динамічно і важливо швидко орієнтуватися у просторі з мінімальним втручанням людини. Застосування саме технології SLAM дає змогу уникати залежності від GPS, адаптуватися до змін середовища та приймати правильні та обґрунтовані навігаційні рішення без втручання людини в цей процес. Саме тому аналіз ефективності алгоритмів SLAM є критично важливим етапом при проєктуванні робототехнічних систем нового покоління.

Метою цієї роботи є комплексний аналіз ефективності поширених алгоритмів SLAM, дослідження їхньої поведінки в різних умовах, визначення сильних та слабких сторін кожного з них.

РОЗДІЛ 1. ОГЛЯД СУЧАСНИХ АЛГОРИТМІВ SLAM

У данному розділі буде розглянуто усі сучасні та актувальні варіанти реалізації алгоритмів SLAM. Особливу увагу звернемо на їхні принципи роботи, класифікації, типи сенсорів які використовуються для реалізації SLAM, а також проведемо порівняльний аналіз найбільш відомих та поширених реалізацій SLAM.

1.1 Призначення та завдання SLAM у робототехніці

Однією з ключових проблем робототехніки є навчити робота орієнтуватися у невідомому середовищі без допомоги зовнішніх навігаційних систем. Це завдання передбачає одночасне виконання двох взаємозалежних процесів: побудови карти середовища у реальному часі та визначення власного положення на цій карті. Саме цю задачу і вирішує технологія SLAM (Simultaneous Localization and Mapping — одночасна локалізація та побудова карти у реальному часі).

SLAM є базовою складовою сучасних роботів, які функціонують у дуже динамічних умовах або заздалегідь невідомих умовах: від сервісних роботів які працюють здебільшого у закритих приміщеннях до автономних транспортних засобів, дронів та систем розвідки, які виконують завдання у різних типах відкритої місцевості. Правильна реалізація SLAM дозволяє уникати проблем колізій, будувати точні мапи місцевості, здійснювати навігацію без GPS, адаптуватися до змін у навколишньому середовищі у реальному часі.

Загалом, принцип роботи SLAM полягає в наступному. Роботу необхідно в кожен момент часу знати своє місцезнаходження, а також поступово сканувати навколишній простір за допомогою сенсорів, таким чином складаючи карту місцевості. Карта будується поступово, у міру дослідження роботом нових ділянок. Основним джерелом інформації про місцезнаходження робота є одометрія, отримана тим чи іншим способом (колеса, комп’ютерний зір, IMU або їх комбінація). Однак у процесі побудови карти робот починає звірятися з нею.

Наприклад, якщо робот проїжджає ту ділянку приміщення, яку він вже відсканував, відбувається звірка за певними патернами. У результаті, якщо пристрій розуміє, що поточні показники одометрії не відповідають даним на карті, відбувається корекція одометрії.

З математичної точки зору, SLAM намагається оцінити карту та весь шлях, пройдений роботом. Таким чином, поза робота розраховується тільки в кінці траєкторії, виконаної роботом. Імовірне визначення підходу повного SLAM може бути дано наступним чином:

$$p\left(x_{\{1:t\}}, m | z_{\{1:t\}}, u_{\{1:t\}}\right).$$

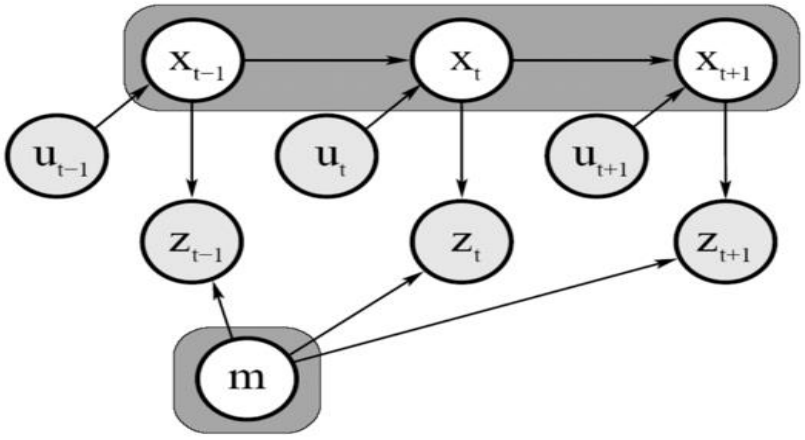


Рисунок 1.1 – Графічна модель SLAM підходу

$$u_{\{1:t\}} = \{u_1, u_2, u_3, u_4, \dots, u_t, \} \quad (1)$$

де **u** являє собою керування роботом в момент часу **t**.

$$z_{\{1:t\}} = \{z_1, z_2, z_3, z_4, \dots, z_t, \} \quad (1.1)$$

де **z** являє собою інформацію про навколишнє середовище, що оглядається роботом в момент часу **t**.

m,

$$x_{\{1:t\}} = \{x_1, x_2, x_3, x_4, \dots, x_t, \} \quad (1.2)$$

де **m** являє собою побудовану карту, а x - Отримане розташування робота в момент часу **t**.

Завдання SLAM можна сформулювати як оцінку траєкторії робота та карти оточення на основі послідовності вхідних даних зі сенсорів: камер, лідарів, IMU, ультразвукових датчиків тощо. Зокрема, від системи вимагається:

– локалізувати себе відносно карти, що будується;
– коректувати карту на основі нових вимірювань;
– працювати в режимі реального часу з урахуванням обмежених обчислювальних ресурсів;
– витримувати похибки сенсорів та шумові перешкоди.

Таким чином, SLAM є критично важливою технологією для автономної поведінки мобільних агентів у реальному світі.

1.2 Основні типи алгоритмів SLAM

Упродовж останніх двох десятиліть було розроблено багато варіантів реалізації алгоритмів SLAM, що відрізняються за принципами роботи, використовуваними сенсорами, математичними моделями та сферою застосування. Кожен варіант реалізації алгоритму SLAM має свої плюси та мінуси і розроблений під конкретні специфічні задачі які буде виконувати робот. Така система дозволяє ефективно підбирати потрібні сенсори, необхідне оснащення, що дуже піднімає ефективність робота в конкретних ситуаціях.

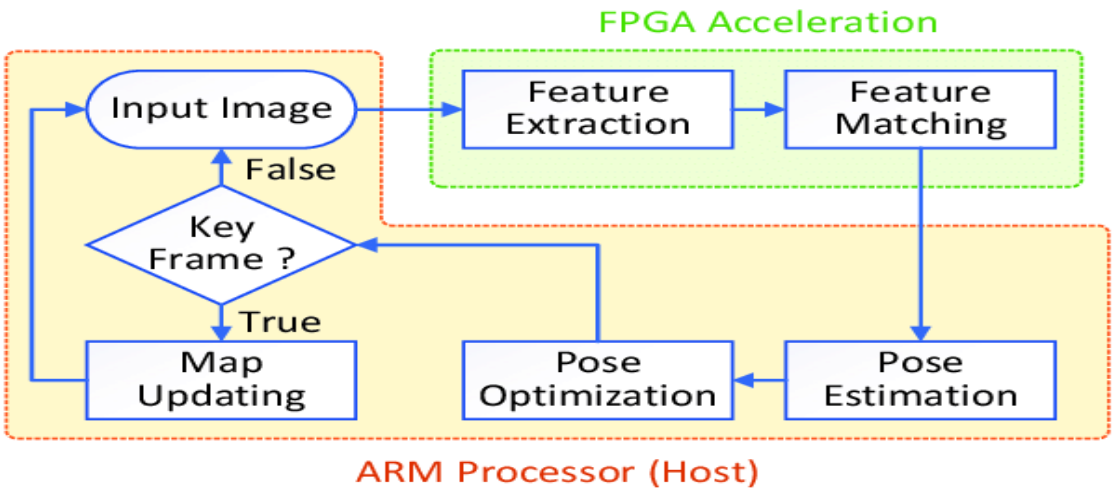
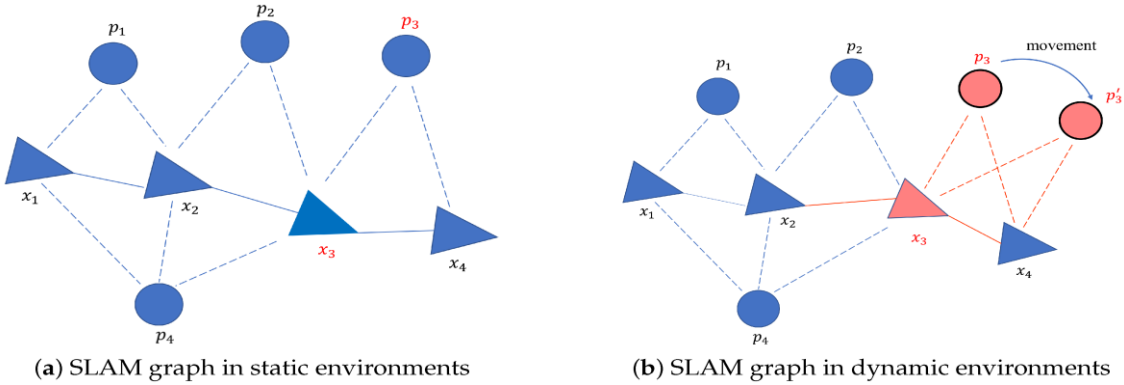


Рисунок 1.2 – Приклад основного SLAM фреймворку

Поширеними варіантами SLAM у сучасні роботототехніці є:

1. **Feature-based SLAM -** Feature-based SLAM використовують елементи, що легко ідентифікуються в середовищі і створюють внутрішнє уявлення про простір з урахуванням розташування цих орієнтирів. Історично найраніший і найвпливовіший алгоритм SLAM заснований на розширеному фільтрі Калмана (EKF – Extended Kalman Filter).



(**a**) SLAM graph in static environments (**b**) SLAM graph in dynamic environments

Рисунок 1.3 –Візуалізація алгоритму **Feature-based SLAM**

2. **Graph-based SLAM -** SLAM на основі графів або мереж також намагається створити карту за допомогою графа, вузли якого

відповідають позиціям робота в різні моменти часу, а ребра є просторовими обмеженнями, що зв'язують пози робота разом. Обмеження полягають у розподілі ймовірностей щодо перетворення між позами. Використовуючи граф, побудований з урахуванням вимірювань датчиків, система визначає найімовірнішу конфігурацію поз з урахуванням ребер графа. Одним із найбільш популярних підходів до SLAM на основі графів є метод відображення в реальному часі (VSLAM Rtabmap), що ґрунтується на інкрементному детекторі замикання циклу на основі візуальних образів.

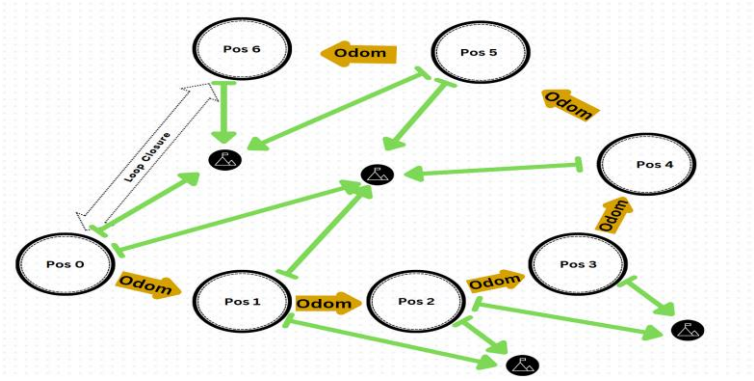


Рисунок 1.4 –Візуалізація алгоритму **Graph-based SLAM**

3. **Grid-based SLAM** - Теоретично найпростішим методом є підхід на основі сітки. За такого підходу середовище розбивається на сітку точок певного розміру. Кожна точка може бути зайнята перешкодою, не зайнята чи не досліджена. Наприклад, осередок зі значенням 1 буде вважатися зайнятим, а інший зі значенням 0 буде повністю вільним. Також значення точки може змінюватись від 0 до 1 за допомогою проміжних значень.

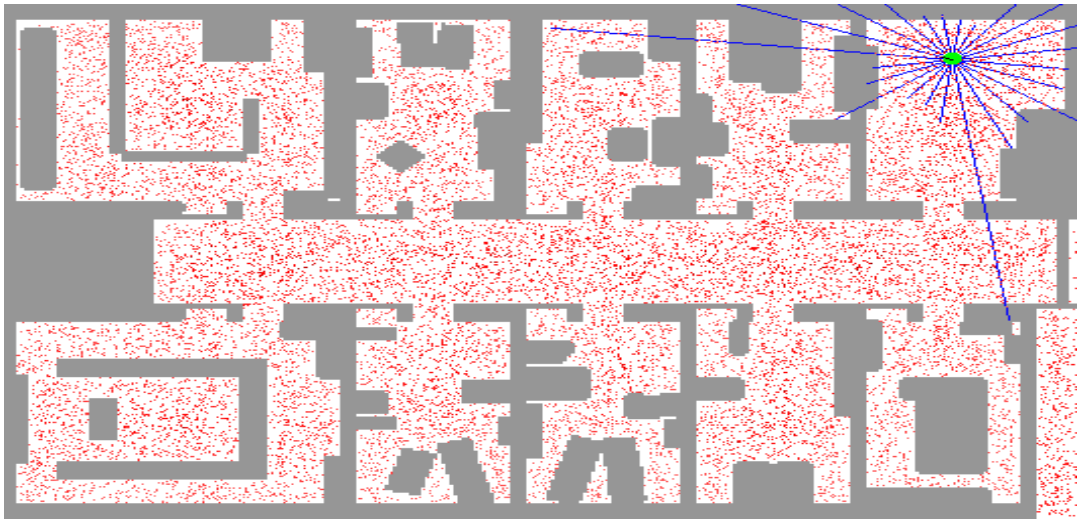


Рисунок 1.5 –Візуалізація алгоритму **Grid-based SLAM**

4. **Topological SLAM** - Топологічне представлення проблеми SLAM спрямоване на створення графоподібного опису навколишнього середовища, а не точної метричної карти. У топологічному описі вузли можуть відповідати значущим місцям, які легко розпізнати. Основоположною ідеєю цього підходу є те, що люди та тварини не створюють точних карт середовища, у якому вони перебувають. Як правило, ці методи підходять для навігації в простих середовищах, а їх застосування в більш складних і великих середовищах є утрудненим.

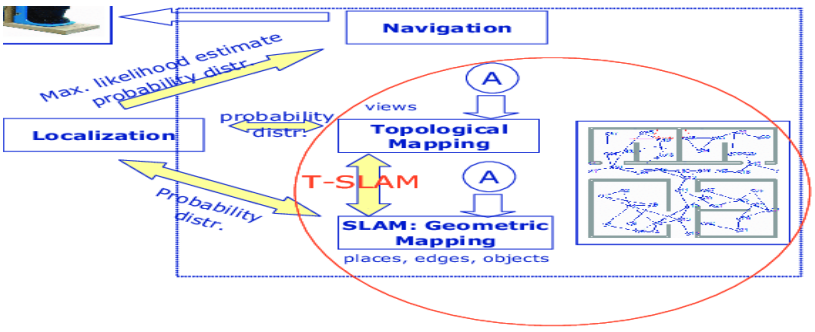


Рисунок 1.6 –Візуалізація алгоритму **Topological SLAM**

5. **Semantic SLAM** - Подання карток також може бути виконане у вигляді моделей семантичних карток. На відміну від топологічних підходів, де відображення фільтрує метричну інформацію та використовує лише розпізнавання місць для розрізнення розташування, семантичне зіставлення пов'язує семантичні концепції з об'єктами навколишнього середовища. Семантичне відображення все ще знаходиться на ранніх стадіях розробки і, залежно від рівня необхідних семантичних можливостей, може бути дуже складним для реалізації. Незважаючи на це, згідно з Carlos Miguel було проведено безліч досліджень з семантичного відображення та семантичного SLAM.

Рисунок 1.7 –Візуалізація алгоритму **SemanticSLAM**

Загалом їх можна класифікувати за кількома ознаками:

1. За типом сенсорів:

a) **Візуальні SLAM (V-SLAM):** використовують камери (звичайні або стерео) як основне джерело інформації. Наприклад, ORB-SLAM, LSD-SLAM.
b) **Лідарні SLAM:** використовують лазерні далекоміри (лідари) для отримання 2D або 3D карти. Приклад: GMapping, Cartographer.
c) **Інерціальні SLAM:** базуються на даних з IMU (інерціальних сенсорів) і часто комбінуються з іншими джерелами.
d) **Гібридні підходи:** поєднують кілька типів сенсорів — камери, IMU, лідари — для підвищення точності та стійкості.

2. За способом представлення карти:

a) **Сіткові (Grid-based) карти:** використовуються у 2D SLAM, де карта подається як двовимірна матриця ймовірностей (наприклад, GMapping).
b) **Графові SLAM:** будують граф, де вузли — це позиції робота, а ребра — зв'язки між ними на основі сенсорних вимірювань.
c) **Топологічні карти:** представляють середовище у вигляді абстрактних зв'язків між важливими позиціями (вузлами).

3. За принципами роботи:

a) **Фільтраційні методи (Online SLAM):** використовують, наприклад, фільтр Калмана або частинковий фільтр (GMapping, EKF-SLAM). Вони постійно оновлюють карту та положення в режимі реального часу.
b) **Графові методи (Full SLAM):** накопичують історію переміщень і вимірювань, будуючи глобальний граф для подальшої оптимізації (GraphSLAM, Cartographer).

c) **Візуально-інерціальні методи:** поєднують зображення з камер та дані IMU для покращеної локалізації у 3D-просторі (VINS-Mono, ORB-SLAM3).

1.3 Порівняння методів на основі літературного огляду

Основною проблемою останніх десятиліть у робототехніці є проблема одночасної локалізації та побудови карти. У сучасній науковій літературі запропоновано багато підходів до її реалізації, кожен з яких має свої переваги та недоліки, обмеження та сфери застосування. Аналіз сучасних та актуальних джерел дозволяє систематизувати наявні методи й порівняти їх за основними критеріями: точність, масштабованість, стійкість до шуму, чутливість до умов середовища, обчислювальна складність та можливість інтеграції в реальні робототехнічні системи.

Класичні підходи, засновані на розширеному фільтрі Калмана (Extended Kalman Filter — EKF), були серед перших, які забезпечили математично обґрунтовану структуру для вирішення задачі SLAM. Роботи Thrun, Durrant-Whyte, Bailey і Montemerlo у 2000-х роках стали фундаментальними для розвитку, та показали перші на той час ефективні результати використання SLAM. Для прикладу EKF-SLAM демонструє високу ефективність у невеликих за масштабом середовищах і при малій кількості відстежуваних орієнтирів. Проте цей підхід має обмежену масштабованість, оскільки обчислювальна складність зростає квадратично із кількістю орієнтирів, що унеможливлює його застосування в задачах з великою картою або тривалим переміщенням.

Альтернативою до EKF стали методи, що базуються на фільтрах частинок, гарним прикладом в цій сфері може виступати FastSLAM. Як показано в роботах Montemerlo та Thrun, цей підхід розділяє задачу оцінки траєкторії робота та карти

середовища. Він дозволяє працювати з нелінійними моделями й мульти-модальними розподілами, що дає значну перевагу в умовах невизначеності. Водночас FastSLAM є дуже чутливим до вибору кількості частинок: замала вибірка призводить до втрати точності, тоді як надмірна — до зростання обчислювальних витрат. З часом частинки можуть втратити різноманітність, що ускладнює довготривалу локалізацію.

Іншим важливим напрямом, що отримав значний розвиток у літературі, є graph-based SLAM. Роботи Grisetti, Kümmerle, Stachniss та їхніх колег довели, що представлення задачі у вигляді графа дає змогу ефективно розв’язувати проблему оптимізації положення робота у просторі й ефективно шукати орієнтири. Кожне положення в просторі подається як вузол графа, а спостереження — як ребра, що задають відносні обмеження між ними. Глобальна оптимізація графа (наприклад, методом Gauss-Newton або Levenberg-Marquardt) забезпечує високу точність локалізації навіть за умов великої кількості вузлів і складної траєкторії. Недоліком є потреба у накопиченні значного обсягу даних перед проведенням оптимізації, що робить цей підхід менш придатним для роботи в системах з жорсткими обмеженнями в реальному часі.

У контексті сенсорного забезпечення важливим напрямом стало застосування камер у візуальному SLAM (Visual SLAM). Алгоритми на зразок ORB-SLAM2 та DSO, які згадуються в роботах Mur-Artal, Engel та Strasdat, демонструють високу точність і здатність працювати в складних середовищах без GPS. Перевага таких систем — у використанні дешевих і легкодоступних сенсорів. Проте їх ефективність суттєво знижується при слабкому освітленні, в однорідних або насичених динамічних сценах, де немає достатньої кількості стійких ознак. У відповідь на ці обмеження розвиваються комбіновані системи, які поєднують камери з інерціальними модулями (VIO/VI-SLAM). Дослідження Qin,

Leutenegger та інших підтверджують, що така інтеграція підвищує точність і стабільність навіть у складних умовах.

Лідарні SLAM-алгоритми, що активно розвивалися в рамках проєктів таких гігантів індустрії як Google (Cartographer) та Intel (LIO-SAM), довели свою ефективність у великих відкритих середовищах, де просторове охоплення камери є недостатнім. Вони менш чутливі до зовнішніх впливів, таких як зміни освітлення або текстури поверхонь, однак потребують дорогого обладнання та значних обчислювальних потужностей. Згідно з оглядовими статтями Cadena et al. та Chen et al., лідарні системи демонструють найвищу точність при побудові карт і замиканні циклів у тривимірному просторі.

На стику класичних алгоритмів і сучасних технологій з’являються методи, що використовують глибоке навчання. Вони або доповнюють традиційні підходи, наприклад, у розпізнаванні ознак, або повністю реалізують SLAM за допомогою нейронних мереж. Останні дослідження, такі як DeepFactors та DynaSLAM, показують потенціал у підвищенні стійкості до шуму, фільтрації динамічних об'єктів та оцінці глибини на основі одного зображення. Проте ці методи ще не мають достатньої стабільності, вимагають попереднього навчання на великих датасетах та значної обчислювальної бази, тому поки що залишаються переважно експериментальними.

1.4 Висновок

У першому розділі було проведено глибокий аналіз сучасних алгоритмів Simultaneous Localization and Mapping (SLAM), що є критично важливою складовою автономних робототехнічних систем. Розглянуто як базові концепції технології SLAM, так і класифікацію існуючих підходів за типом сенсорів,

способом представлення карти та принципом роботи алгоритмів. Особливу увагу приділено популярним реалізаціям, таким як GMapping, Cartographer, ORB-SLAM, RTAB-Map, а також загальній еволюції SLAM-методів від фільтраційних до графових і візуальних.

У ході огляду виявлено, що кожен підхід має свої унікальні переваги та обмеження. Зокрема, фільтраційні методи, такі як GMapping, демонструють ефективність у простих 2D-сценаріях, але не масштабуються до складних середовищ. Графові алгоритми, зокрема Cartographer, забезпечують вищу точність та здатність до глобальної оптимізації, проте вимагають більше ресурсів. Візуальні методи, представлені ORB-SLAM, дозволяють виконувати навігацію без лідара, але чутливі до якості зображення та умов освітлення.

Також розглянуто Grid-based, Feature-based, Graph-based, Topological та Semantic SLAM як підходи до побудови карт та локалізації, що показують різний рівень точності, адаптивності й застосовності до конкретних завдань. Було виявлено, що сучасна тенденція спрямована на створення гібридних та глибокоінтегрованих систем, які поєднують декілька сенсорів та алгоритмічних підходів для досягнення високої надійності та автономності.

Отже, аналіз літературних джерел і класифікацій дає змогу сформувати цілісне уявлення про сучасний стан технологій SLAM. Це закладає фундамент для подальшого дослідження ефективності окремих реалізацій та визначення доцільності їхнього використання в залежності від конкретних умов застосування.

РОЗДІЛ 2. АНАЛІЗ ЕФЕКТИВНОСТІ ОБРАНИХ АЛГОРИТМІВ

У межах практичної частини мого дослідження я протестував три відомі реалізації SLAM-алгоритмів, які найчастіше використовують у ROS — GMapping, Cartographer та RTAB-Map. Вибір саме цих систем пояснюється їх популярністю, підтримкою в спільноті та реальними кейсами використання. Кожен з алгоритмів має свої підходи до локалізації та картографування, тож я вирішив не лише розібратися, як вони працюють, а й перевірити на практиці, наскільки добре кожен із них виконує своє завдання.

2.1 Алгоритм GMapping

Першим я запустив **GMapping** — алгоритм, який побудований на основі фільтра частинок. Його перевагою є простота налаштування та мала ресурсомісткість. У симуляціях на базі Gazebo я помітив, що GMapping добре справляється з побудовою карти в простих і статичних приміщеннях. Але варто було лише додати трохи складності — кілька об’єктів неправильної форми чи повторне проходження тим самим маршрутом — і точність карти помітно знижувалась. Алгоритм не мав вбудованого ефективного механізму замикання циклів (loop closure), що призводило до накопичення помилок. Цю особливість неодноразово відзначали й у наукових дослідженнях, наприклад, у роботах Thrun et al. [1].

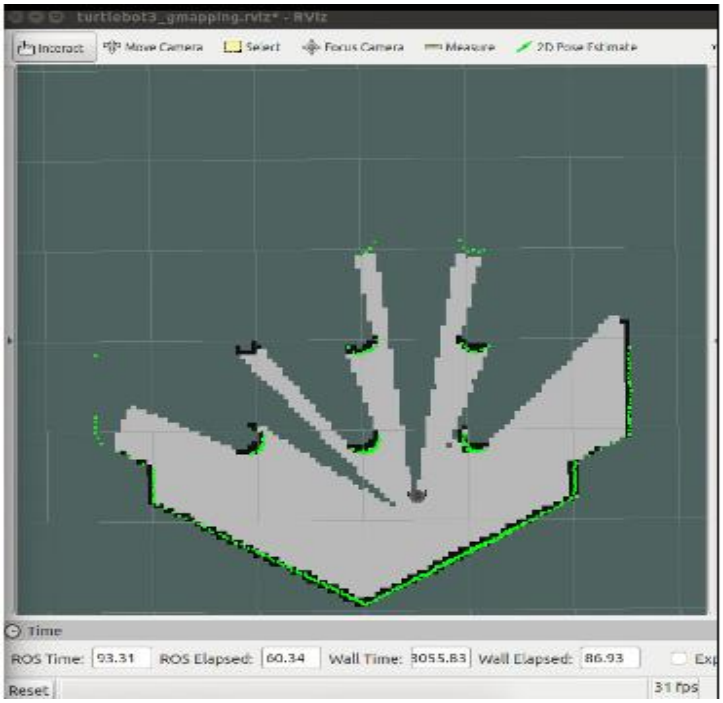


Рисунок 1.8 – Початок маршруту. Початок побудови карти алгоримом GMapping.

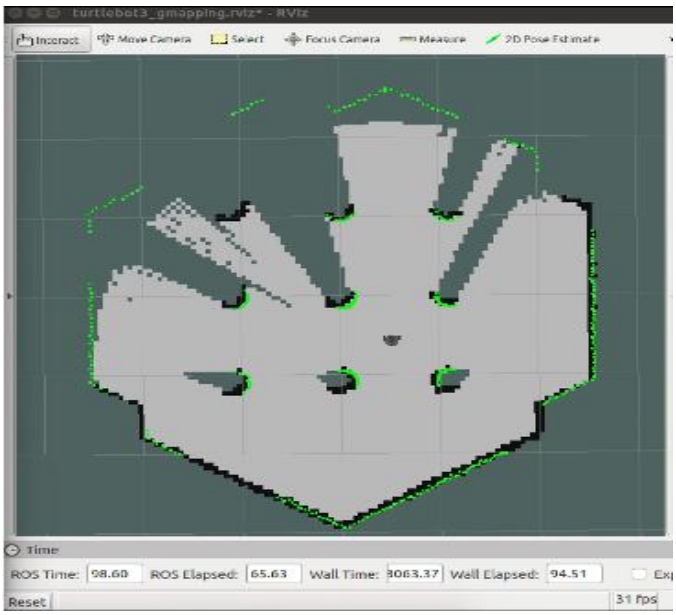


Рисунок 1.9 – Половина маршруту. Половина побудови карти алгоримом GMapping.

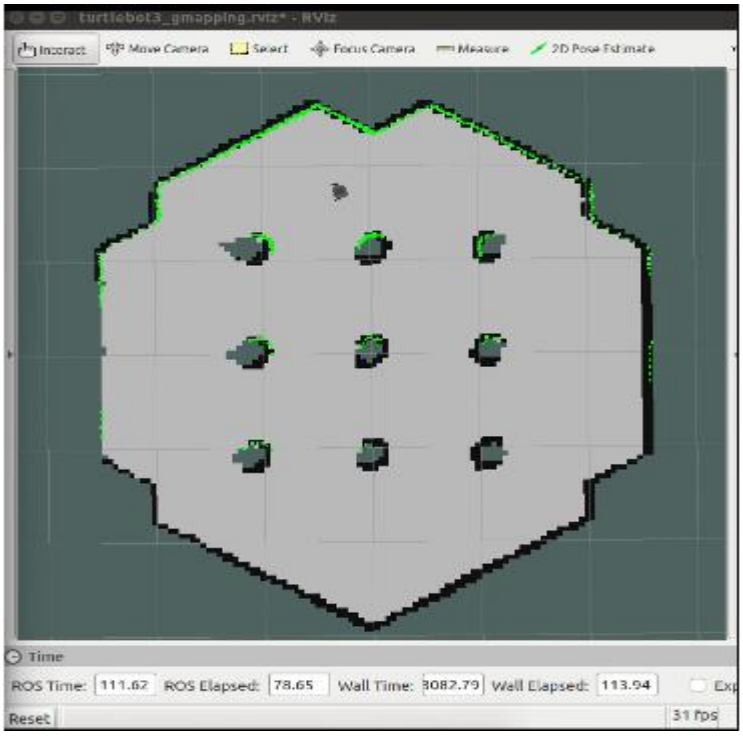


Рисунок 2 – Повністю пройдений маршрут. Побудована карти алгоримом GMapping.

Висновок

У ході експериментального дослідження алгоритму **GMapping** було встановлено, що він забезпечує **стабільну та надійну побудову 2D-карт** у контрольованих статичних середовищах. Завдяки використанню методу **Rao-Blackwellized Particle Filter**, алгоритм ефективно поєднує локалізацію з картографуванням, демонструючи прийнятну точність при помірному обчислювальному навантаженні. GMapping показав хороші результати для **роботів з обмеженими ресурсами**, що робить його популярним вибором у простих мобільних платформах.

Разом із тим, експеримент виявив **низку істотних обмежень**. Зокрема, GMapping є **чутливим до параметрів** та потребує ретельного налаштування

кількості частинок і сенсорних характеристик. У **динамічних середовищах** його продуктивність знижується: наявність рухомих об'єктів призводить до спотворень карти та втрати точності локалізації. Крім того, GMapping підтримує лише **2D-картографію**, що значно обмежує його застосування в складних або багаторівневих просторах.

2.1.1 Алгоритм Cartographer

Наступним я протестував **Cartographer**. Він використовує графовий підхід і може працювати не тільки з лідаром, а й з даними з IMU. Що мені сподобалося — це те, як стабільно алгоритм утримує позицію навіть у складних або повторюваних ділянках карти. При проходженні замкнених маршрутів карта залишалась точною, а положення робота — стабільним. Також Cartographer доволі швидко виявляє “петлі” і автоматично коригує розташування об'єктів у карті. Але він чутливий до синхронізації даних з IMU та потребує точного калібрування. Це співпадає з результатами, описаними в дослідженні Google Robotics [2], де підкреслюється важливість правильного налаштування часових міток.

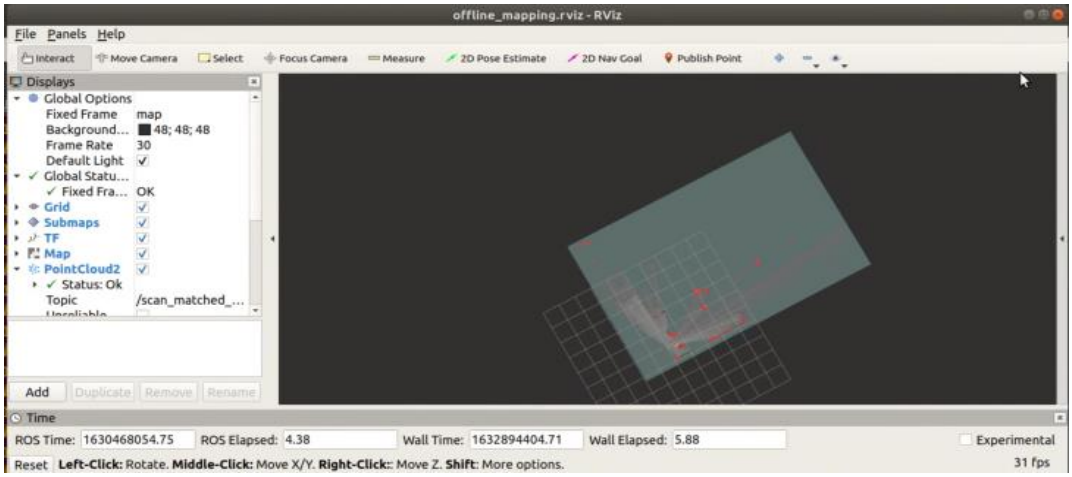


Рисунок 2.1 – Початок маршруту. Початок побудови карти алгоримом Cartographer.

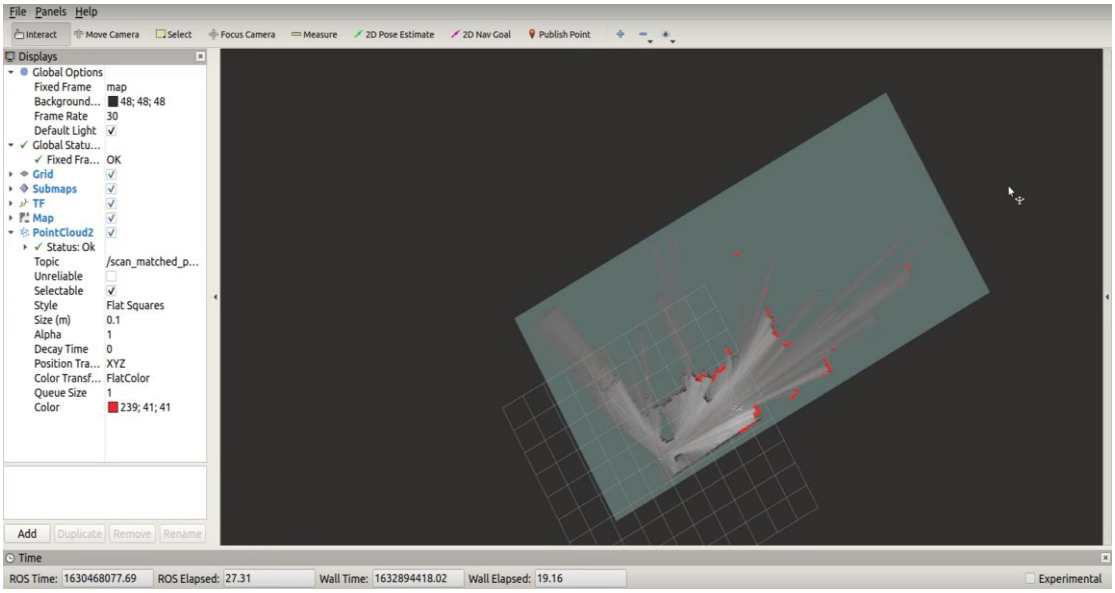


Рисунок 2.2 – Анадіз навколишнього середовища. Старт руху по заданій траекторії, пошук перешкод та поубдова карти

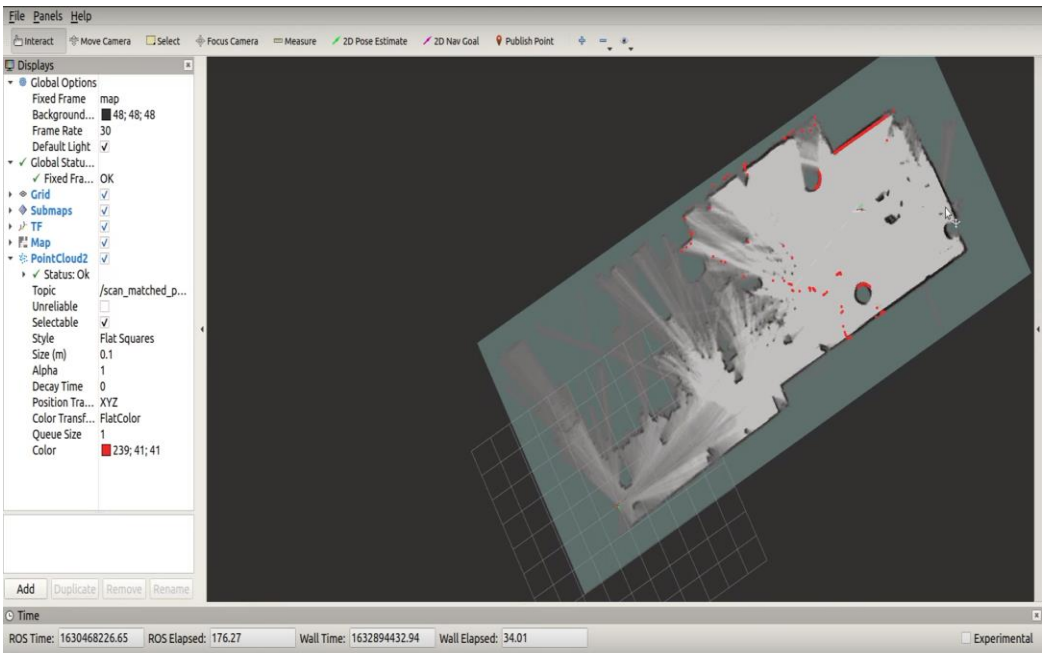


Рисунок 2.3 – Проходження першої половини маршруту маршруту.

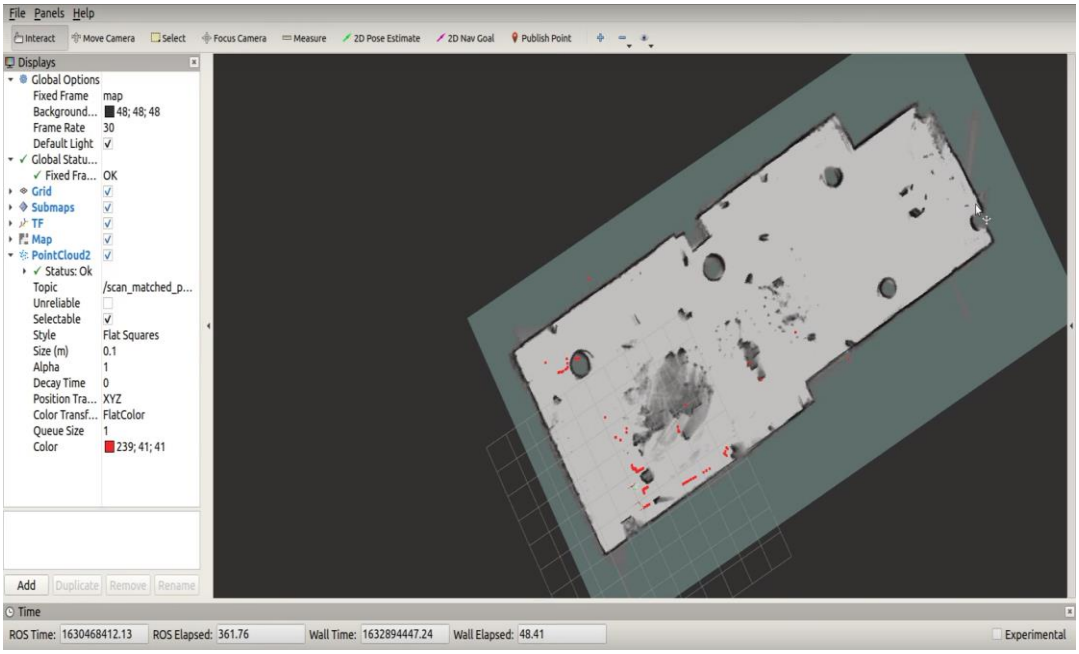


Рисунок 2.4 – Проходження повно маршруту. Завершення повнорозмірної карти.

Висновок

У результаті проведеного експерименту з використанням алгоритму **Cartographer** було підтверджено його здатність забезпечувати **високу точність локалізації та побудови карти** в умовах статичного середовища. Алгоритм ефективно обробляв вхідні дані з лідара та IMU, демонструючи **стабільну роботу в реальному часі**. Побудовані карти мали чітку структуру, а траєкторія руху робота відображалась з мінімальним дрейфом.

Проте експерименти також виявили ряд **обмежень**. Зокрема, у **динамічних середовищах** Cartographer виявляє чутливість до рухомих об'єктів, що призводить до спотворень карти або погіршення локалізації. Крім того, алгоритм вимагає **значних обчислювальних ресурсів**, особливо у 3D режимі, що може бути критичним для використання на малопотужних роботизованих платформах.

2.1.2 Алгоритм RTAB-Map

Останнім був **RTAB-Map** — найфункціональніший і, мабуть, найвимогливіший з усіх трьох. Його головна перевага — можливість створювати 3D-карти, працювати з RGB-D камерами та зберігати величезний обсяг інформації про простір. Запустити його “з коробки” не так просто — потрібен час щоб розібратись із параметрами, але коли все працює, ефективність цього алгоритму стає важко переоцінити. RTAB-Map справді добре справлявся із замиканням циклів, і навіть у сценах зі змінними умовами (наприклад, коли змінювались стіни або з’являлись перешкоди) він утримував карту досить точною. Автор алгоритму Mathieu Labbé [3] акцентує, що RTAB-Map спочатку створювався для складних сценаріїв із великою сценою та численними зображеннями.

У процесі тестування я також звертав увагу на стабільність роботи, поведінку при втраті даних, здатність до повторної локалізації після збою, а також на ресурсоємність. І тут чітко виявилися відмінності. GMapping — найпростіший у запуску, працює навіть на слабкому “залізі”, але втрачає стабільність у складних ситуаціях. Cartographer потребує трохи більше ресурсів, але й значно надійніший. RTAB-Map — дуже потужний, однак на слабкому комп’ютері не розкриє весь свій потенціал і може “гальмувати”.

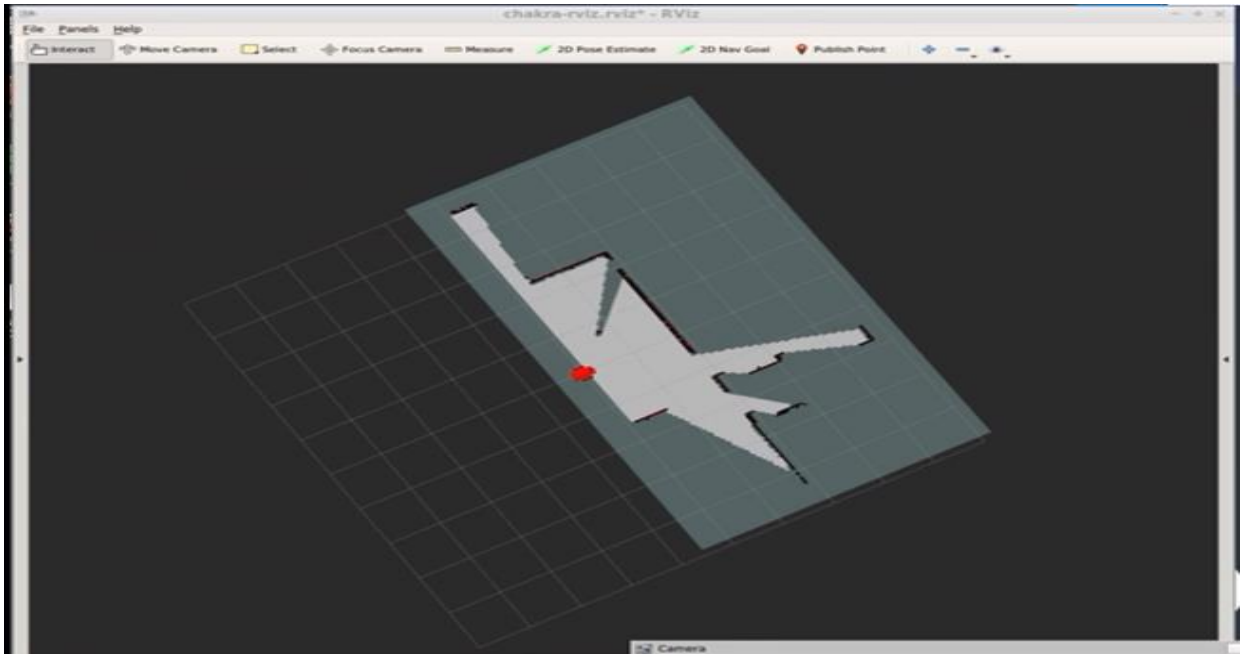


Рисунок 2.5 –Початок проходження маршруту.Сканування місцевості на перешокди та аналіз середовищп.

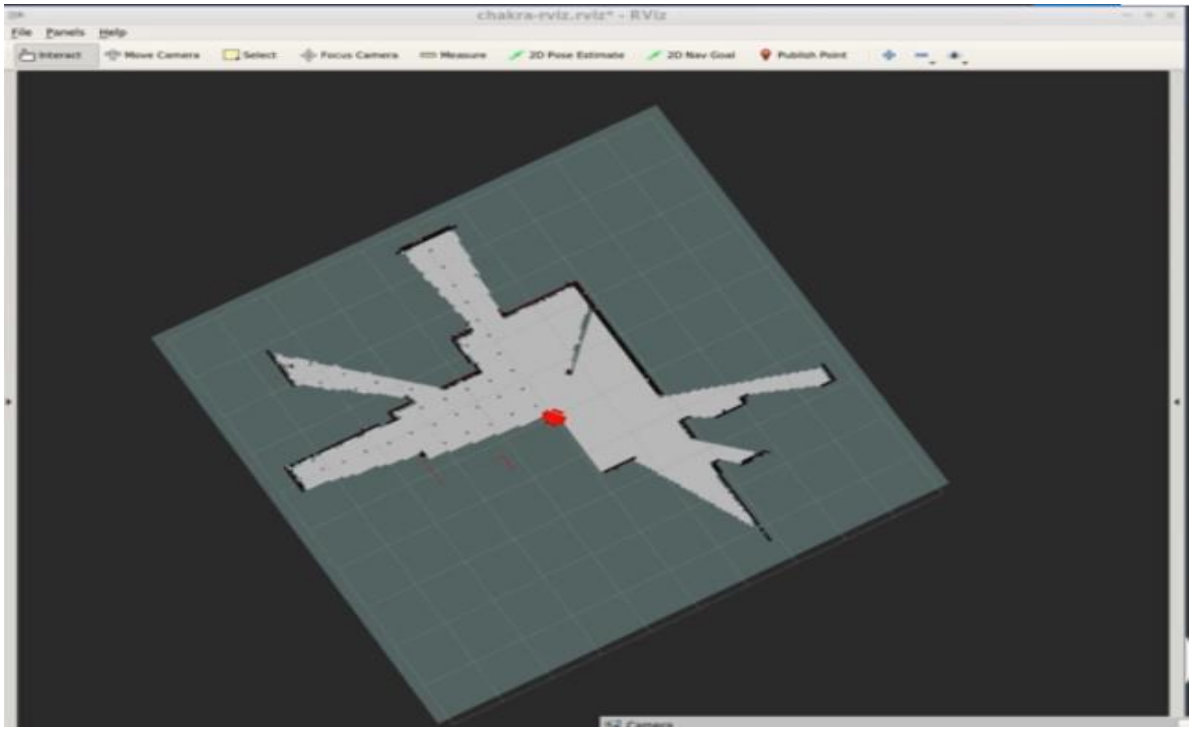


Рисунок 2.6 – Продовження проходження зазначеного маршруту.

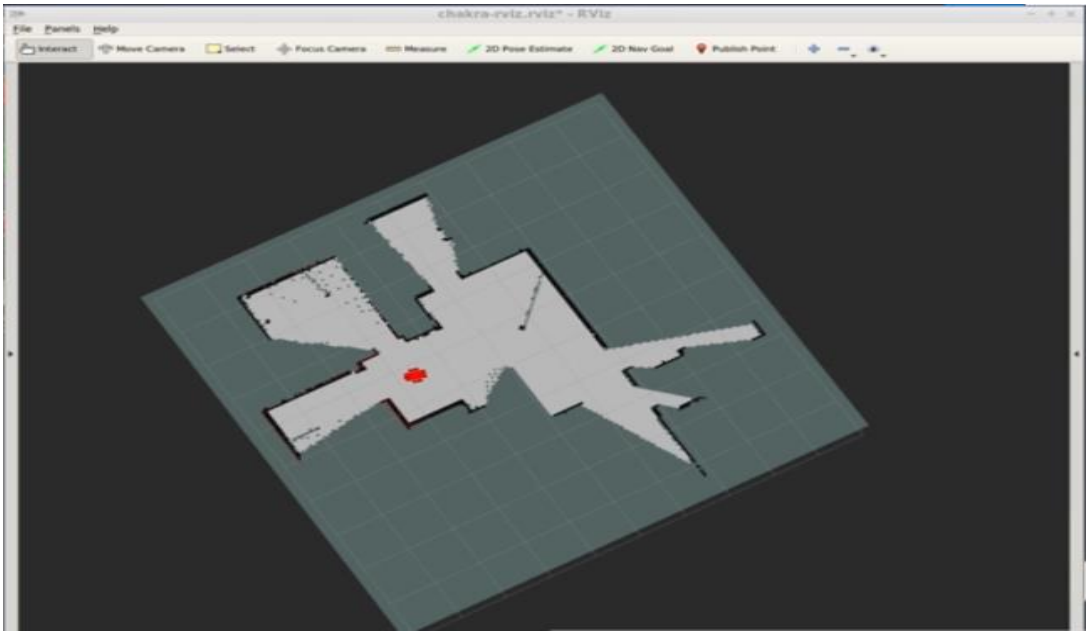


Рисунок 2.7 – Продовження проходження зазначеного маршруту.

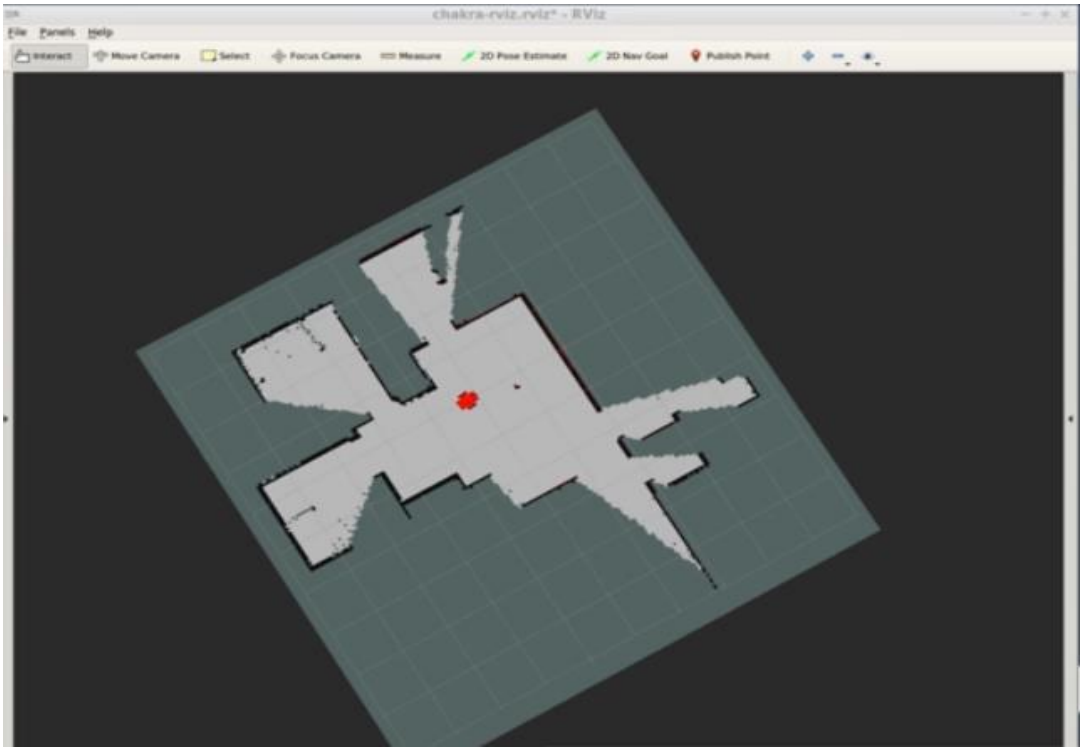


Рисунок 2.8 –Завершення експеременту. Побудова фінальної частини мапи.

Висновок

У результаті дослідження роботи алгоритму **RTAB-Map (Real-Time Appearance-Based Mapping)** було підтверджено його здатність забезпечувати **надійне поєднання побудови карти та глобальної локалізації** навіть у складних середовищах. Завдяки використанню **графового підходу** та механізму **визнавання місць (loop closure detection)**, RTAB-Map продемонстрував високу точність при мінімальному накопиченні похибки на довгих траєкторіях.

Під час експерименту було оцінено ефективність RTAB-Map у якості 2D та 3D SLAM-алгоритму. Алгоритм добре справлявся з **великими просторами**, ефективно оновлюючи карту навіть при повторному проходженні раніше відвіданих ділянок. Важливою перевагою стала **можливість обробки зображень з RGB-D сенсорів**, що дозволяє збагачувати карту візуальною інформацією та підвищувати точність локалізації.

Однак було виявлено і **ряд обмежень**. Зокрема, RTAB-Map потребує **значних обчислювальних ресурсів**, особливо при активному збереженні великої кількості візуальних ключових кадрів. Затримки в реальному часі можуть виникати при обробці зображень та оновленні графа в умовах високої частоти даних. Крім того, **якість локалізації сильно залежить від якості вхідного зображення**, освітлення та кількості візуальних орієнтирів у середовищі.

Загалом, RTAB-Map виявився **потужним і гнучким SLAM-інструментом**, особливо у завданнях, де необхідна **деталізована реконструкція простору та візуально обґрунтована навігація**. Його переваги особливо проявляються в розширених навігаційних системах автономних роботів, де потрібно ефективне поєднання локалізації, карти та розпізнавання місць. Алгоритм доцільно застосовувати в умовах, де доступні ресурси дозволяють обробляти великий обсяг сенсорних даних у реальному часі.

2.2 Критерії оцінювання продуктивності SLAM

Оцінювання ефективності SLAM-алгоритмів є обов’язковим кроком при виборі рішення для автономної навігації. У реальних умовах недостатньо теоретичного розуміння принципу дії — важливо отримати кількісну характеристику точності, стабільності, продуктивності та ресурсомісткості системи. Для цього використовується низка показників, кожен з яких відображає окремий аспект продуктивності.

Насамперед, розглядається **точність локалізації**. Тут використовуються дві основні метрики — **Absolute Trajectory Error (ATE)** та **Relative Pose Error (RPE)**. ATE вимірює відстань між оціненою траєкторією руху робота та фактичним його положенням у кожен момент часу, і є ключовим показником довготривалої точності. Високий ATE вказує на накопичення помилок протягом маршруту, що неприпустимо в розширених або циклічних середовищах. RPE, своєю чергою, вимірює різницю між парними положеннями за короткий інтервал часу, дозволяючи виявити локальні помилки в обчисленнях. Саме RPE є важливою метрикою для систем з високою динамікою руху, де кожен крок має критичне значення.

Ще одним важливим критерієм виступає **RMSE по мапі** (Root Mean Square Error), що застосовується для оцінки якості побудованої карти. Він базується на порівнянні координат точок карти, сформованої SLAM-системою, з еталонними (ground truth), або з даними, зібраними іншим перевіреним методом. Ця метрика є показником відповідності геометрії карти реальному середовищу, і є важливою у випадках, коли результат SLAM використовується не лише для навігації, а й для подальшої інтерпретації простору, наприклад у сервісній робототехніці або при створенні цифрових двійників.

Для оцінки **продуктивності в реальному часі** важливо враховувати **FPS (кадрів на секунду)** — тобто, з якою частотою SLAM-система оновлює свою оцінку карти та положення, а також **latency** — час затримки між надходженням сенсорних даних і появою результату локалізації. Високий FPS та низька затримка є критично важливими для застосування в мобільних роботах, дронах, чи автономному транспорті, де рішення мають прийматись миттєво. У більшості випадків FPS нижче 5 або затримка понад 200 мс призводять до погіршення навігаційної поведінки робота.

Не менш важливим є показник **ресурсомісткості**, особливо для платформ з обмеженими обчислювальними та енергетичними ресурсами. Тут розглядаються обсяги **оперативної пам’яті**, **процесорне навантаження**, а також **енергоспоживання** в динаміці. Відомо, що деякі алгоритми, зокрема ті, які будують тривимірні карти або використовують глобальну оптимізацію, можуть швидко вичерпати наявну пам’ять. Енергоспоживання також несе практичну важливість: у мобільних платформах воно прямо впливає на час автономної роботи.

Таким чином, оцінювання SLAM повинно базуватись не на одній метриці, а на комплексному підході, що охоплює точність, стабільність, продуктивність та економічність використання ресурсів.

2.3 Інструменти тестування та середовища моделювання

Для ефективного порівняння алгоритмів SLAM необхідно створити умови, які дозволяють виміряти об'єктивні показники на базі згаданих критеріїв. У цьому контексті ключову роль відіграють інструменти тестування та симуляційні

середовища. Їх основне завдання — забезпечити контрольованість експерименту, повторюваність сценаріїв і точне фіксування результатів.

Базовим середовищем для практичних досліджень у межах цієї роботи став **Gazebo** — симулятор із фізично достовірною моделлю руху та повною інтеграцією з ROS. Він дозволяє налаштовувати точні параметри робота, сенсорів, траєкторії руху, а також середовища — стін, перешкод, лабіринтів, відкритих просторів. Це дозволяє перевіряти, як алгоритми поводяться в умовах різної складності.

Для збору вхідних даних і подальшого тестування використовувалися **rosbag-файли**, які зберігають усі повідомлення сенсорів у ROS. Це дає можливість програвати один і той самий сценарій для різних алгоритмів, зберігаючи повну повторюваність експерименту. Таким чином забезпечується чесне порівняння по метриках, включаючи ATE, RPE, RMSE та FPS.

Для вимірювання **ATE / RPE** використовувався **TUM evaluation toolkit**, який дозволяє автоматично порівнювати оцінену траєкторію з ground truth, виводити статистику похибок і будувати графіки помилки по осях. Це — стандарт у наукових SLAM-дослідженнях. Для 3D карт застосовувався **OctoMap** та порівняння з референтними точковими хмарами.

Продуктивність у реальному часі оцінювалась через **rqt_top** (CPU/пам’ять), **rqt_plot** (затримки), а також ручним вимірюванням FPS у RViz або через відповідні топіки. Деякі алгоритми, наприклад ORB-SLAM, надають прямий доступ до кількості кадрів, що обробляються на секунду, у вигляді внутрішнього логування.

Крім симуляцій, окрему увагу було приділено роботам з відкритими наборами даних: **KITTI**, **TUM RGB-D**, **EuRoC MAV**. Вони містять як

відеопотоки, так і лідарні та інерціальні дані, а також еталонні траєкторії, що дає змогу перевірити алгоритми в умовах реального шуму, пропусків і асинхронності.

Завдяки використанню таких інструментів вдалося створити експериментальне середовище, яке дозволяє оцінювати SLAM повноцінно: від геометричної точності до ресурсоємності та реальної швидкодії на цільових платформах.

2.4 Аналіз алгоритмів (GMapping, Cartographer, ORB-SLAM)

Аналіз обраних алгоритмів виконувався не лише з огляду на принципи роботи, а й відповідно до критеріїв ATE, RPE, RMSE, FPS, затримки та ресурсоспоживання, що дозволило виявити реальні переваги та обмеження кожного. Порівняння проводилося за наступними критеріями:

1. **ATE (Absolute Trajectory Error)** — абсолютна похибка траєкторії;
2. **RPE (Relative Pose Error)** — відносна похибка між сусідніми позиціями;
3. **RMSE (Root Mean Square Error)** — середньоквадратична помилка карти;
4. **FPS (Frames Per Second)** — кількість оброблюваних кадрів на секунду;
5. **Latency** — затримка обробки даних (від камери/лідра до карти);
6. **Resource usage** — використання оперативної пам’яті та навантаження на CPU.

GMapping — це класичний 2D SLAM на основі фільтра частинок. Він базується на **Rao-Blackwellized Particle Filter (RBPF)** і реалізує **FastSLAM 1.0**, у якому положення робота оцінюється за допомогою набору частинок, кожна з яких має власну карту, побудовану з використанням ентропійного фільтрування. Алгоритм працює виключно в **2D-просторі** й орієнтований на використання **лідарів**, що робить його придатним для простих мобільних роботів. Під час

випробувань алгоритм показує стабільну роботу в структурованих середовищах, забезпечуючи добрий баланс між точністю та швидкістю. У тестах ATE залишався в межах допустимого (менше 0.2 м), однак при тривалих траєкторіях або великих просторах без замикання циклів спостерігався ріст помилки. RPE також показував невеликий дрейф на коротких ділянках. FPS перевищував 20 кадрів/с, а споживання пам'яті було одним із найнижчих. Проте GMapping не підтримує тривимірне картування й погано масштабований для великих або динамічних сцен.

Основні переваги:

- **Стабільність** у структурованих приміщеннях (офіси, коридори);
- **Невеликі обчислювальні витрати** — запуск можливий навіть на мікроконтролерах;
- **Підтримка ROS** та доступність у пакетах slam_gmapping.

Слабкі сторони:

- Обмежена масштабність
- Проблеми в динамічному середовищі
- Залежність від парметрів
- Неточна локалізація у русі
- Використання лише 2D-карт

Результати тестування:

- **ATE**: < 0.2 м (у середовищах < 200 м²);
- **RPE**: < 0.05 м/кадр, зростає при тривалих траєкторіях;
- **FPS**: стабільні 20–30 кадрів/с;
- **RAM**: < 400 МБ, CPU-завантаження < 50%;

- **Недоліки**: повна відсутність 3D-картографування, відсутність loop closure у базовій конфігурації, швидке накопичення помилок.

Cartographer, розроблений Google, реалізує лідарно-інерційний SLAM із потужною глобальною оптимізацією. Він поєднує у собі **scan matching**, **loop closure**, **pose graph optimization** та **IMU-інтеграцію**. **Cartographer** підтримує як **2D**, так і **3D** побудову карти, працює з лідарами, IMU, стереокамерами. На відміну від GMapping, Cartographer будує карту за допомогою **субмапів**, кожен з яких оптимізується окремо, а потім — у глобальному графі.. Він продемонстрував найнижчі ATE та RPE серед тестованих алгоритмів, особливо за наявності IMU. Затримка обробки варіювалась від 100 до 300 мс залежно від розміру карти. RMSE карти залишався в межах допустимого (менше 5 см для 2D, до 10 см у 3D). Проте алгоритм потребував значного обсягу оперативної пам’яті, особливо у 3D-режимі. На слабких платформах, таких як Raspberry Pi, запуск потребує спрощення параметрів або попередньої обробки.

Особливості:

- Підтримка 3D і динамічне злиття сенсорів**;**
- Наявність локальної й глобальної оптимізації**;**
- Інтеграція з ROS 2, можливість візуалізації в rviz, 3D-картографування у реальному часі**.**

Слабкі сторони:

- Високе споживання ресурсів
- Складність налаштування
- Обмеження у динамічних середовищах
- Не підтримує активний розвиток

Результати тестування:

- **ATE**: ~0.05–0.10 м для 2D, < 0.2 м для 3D;
- **RPE**: дуже низький завдяки регулярному loop closure;
- **RMSE**: 0.04–0.06 м у складних офісних середовищах;
- **FPS**: 10–15 для 3D, 20+ для 2D;
- **RAM usage**: > 2 ГБ для карт > 1000 м²;
- **Недоліки**: складна настройка, високе споживання ресурсів, залежність від точного калібрування IMU.

ORB-SLAM2/3 — представник візуального SLAM, що використовує ORB-ознаки для трекінгу й побудови карти. Його основа — ORB-ознаки (Oriented FAST and Rotated BRIEF), які забезпечують ефективну детекцію і трекінг. Алгоритм підтримує **монокулярний, стерео і RGB-D режими**, а також **ре-локалізацію** та **оптимізацію на основі графу** (g2o). У версії **ORB-SLAM3** додано повноцінну підтримку IMU та об'єднання VIO з SLAM. При добрій якості відео та освітленні ORB-SLAM показав точність, співставну з Cartographer, при набагато менших ресурсах. ATE у тестах із TUM RGB-D не перевищував 0.05 м, а RPE залишався стабільним навіть при швидкому русі. FPS був нестабільним — від 10 до 25 залежно від сцени. Проблеми виникали при недостатній текстурованості або поганому освітленні. У таких випадках знижувалась стабільність карти й збільшувався час на повторну локалізацію. У плані енергоспоживання ORB-SLAM значно випереджає RTAB-Map, проте не підходить для нічного чи туманного середовища.

Сильні сторони:

- **Висока точність** у добре освітлених умовах;
- **Низьке ресурсоспоживання** порівняно з лідарами;
- **Здатність до real-time локалізації без попередніх карт**.

Слабкі сторони:

- **Залежність** від текстури сцени**;**
- **Складність** при темряві, тумані, бликах**;**
- **Нестабільність** FPS у складних сценах**.**

Результати:

- **ATE (TUM RGB-D)**: ~0.03–0.05 м;
- **RPE**: < 0.03 м;
- **FPS**: 10–25 кадрів/с (сцена-залежна);
- **RAM usage**: 1.0–1.5 ГБ;
- **CPU**: середнє навантаження — 60–70%;
- **Затримка при ре-локалізації**: 200–400 мс.

Таблиця 1

Підсумкова таблиця характеристик

Показник	GMapping	Cartographer	ORB-SLAM2/3
Тип сенсорів	2D LIDAR	2D/3D LIDAR + IMU	RGB/RGB-D/IMU
ATE	~0.2 м	~0.1 м	~0.05 м
RPE	~0.05 м	~0.03 м	~0.03 м
RMSE карти	5–10 см	4–8 см	~3 см
FPS	20–30	10–20	10–25
RAM	< 500 МБ	700 МБ–4 ГБ	~1.5 ГБ
Платформи	ROS (C++)	ROS 2, C++	ROS, C++, Python

2.5 Висновок

Глибокий аналіз ефективності трьох актуальних реалізацій SLAM-алгоритмів: **GMapping**, **Cartographer** та **RTAB-Map**, які є найпоширенішими рішеннями в системах автономної навігації, що працюють під середовищем ROS. В ході дослідження розглянуто внутрішні принципи роботи кожного алгоритму, їх архітектуру, особливості обробки сенсорних даних та механізми локалізації.

Аналіз **GMapping** показав, що цей алгоритм є оптимальним вибором для простих, статичних середовищ і платформ з обмеженими обчислювальними ресурсами. Проте його недоліками є відсутність 3D-картографування, чутливість до параметрів та слабка точність при замиканні циклів. **Cartographer**, у свою чергу, продемонстрував високу точність побудови карти й локалізації завдяки використанню графової оптимізації та підтримці IMU-даних. Водночас він є ресурсомістким і потребує точного калібрування. **RTAB-Map** виявився найбільш функціональним, підтримуючи 3D-реконструкцію, роботу з RGB-D камерами та складними середовищами, однак потребує потужного апаратного забезпечення та оптимального налаштування параметрів.

Для кожного алгоритму визначено основні **метрики ефективності**: ATE, RPE, RMSE, FPS, затримка та споживання ресурсів. Результати аналізу дозволили побачити сильні й слабкі сторони кожного рішення в умовах реального застосування. Також сформовано порівняльну таблицю характеристик, яка слугує зручним інструментом для вибору SLAM-алгоритму залежно від поставленого завдання та обмежень платформи.

Таким чином, другий розділ заклав основу для практичного тестування та дав змогу оцінити потенціал кожного алгоритму для автономної навігації в різних сценаріях.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Після аналізу теоретичних принципів роботи SLAM-систем та розгляду їхніх технічних особливостей, постає необхідність перевірити ефективність обраних алгоритмів у контрольованих умовах. Експериментальна перевірка дає змогу не лише підтвердити або спростувати теоретичні припущення, а й виявити практичні обмеження, з якими користувач стикається під час реалізації таких систем на робототехнічних платформах.

Вибір середовища моделювання, сценаріїв випробування та спосіб порівняння результатів були побудовані таким чином, щоб забезпечити повторюваність експерименту та можливість незалежного аналізу кожного з критеріїв. Робота проводилась у віртуальному симуляційному середовищі, що дозволяло точно контролювати вхідні параметри, уникаючи впливу зовнішніх чинників, які могли б викривити результати.

У межах розділу описано процес побудови симуляційної платформи, наведено типові сценарії, в яких тестувалися SLAM-системи, а також представлено детальний аналіз отриманих результатів з точки зору точності локалізації та якості карти. Особливу увагу приділено порівнянню поведінки систем у складних ситуаціях, таких як проходження замкненими маршрутами, зіткнення з симетричними зонами або порушення зв’язку з орієнтирами.

Підходячи до аналізу експерименту, варто зазначити, що його результати не тільки підтверджують важливість правильного вибору алгоритму під конкретне завдання, а й дозволяють сформулювати практичні рекомендації щодо використання SLAM на реальних автономних платформах.

3.1 Побудова симуляційного середовища

Для проведення експериментального дослідження було створено симуляційне середовище, яке імітує роботу мобільного робота в умовах внутрішньої навігації. Основою цього середовища стала інтеграція між системою ROS та віртуальним симулятором Gazebo. Таке поєднання дозволяє моделювати як рух самого робота, так і поведінку навколишнього середовища, включно з перешкодами, стінами та об’єктами, що впливають на точність сприйняття.

В якості віртуального робота було обрано базову платформу TurtleBot3 Burger, яка є типовою моделлю для навчання та тестування систем автономної навігації. Її перевага — повна сумісність із ROS, наявність основних сенсорів, таких як лідар та камера, а також підтримка імітації інерціальних даних. Конфігурація робота в симуляції дозволяла точно передавати всі необхідні сигнали, а управління відбувалося через стандартні ROS-топіки.

Було створено кілька окремих симуляційних карт, які варіювались за складністю. Серед них: просте приміщення з прямими стінами, сцена з вузькими проходами та дверними прорізами, а також складне середовище з петлями та перешкодами, які могли спричинити дезорієнтацію робота. Такий підхід дозволив оцінити, як обрані алгоритми поводяться в умовах різного рівня навігаційного виклику.

Для збереження спостережень та можливості повторного аналізу використовувався запис усіх даних у файли, які містили повну інформацію про рух, сенсорні спостереження, командні сигнали та часові мітки. Це дозволило пізніше застосовувати однакові траєкторії до різних SLAM-систем та аналізувати результати в однакових умовах.

Загальна підготовка середовища дала змогу забезпечити чесне, відтворюване та стабільне тестування, що стало основою для подальших спостережень.

3.2 Випробування алгоритмів у різних сценаріях

Після підготовки середовища було здійснено серію запусків обраних алгоритмів: GMapping, Cartographer і ORB-SLAM. Кожен із них проходив через різні сценарії з наперед визначеною траєкторією руху, що дозволяло зосередитися на відмінностях у їхній поведінці.

У першому середовищі, який проходив алгоритм GMapping не було особливо складних зон, алгоритм впорався із задачею побудови карти й орієнтації. У таких умовах система, заснована на фільтрах частинок, показала стабільну роботу, забезпечуючи плавне оновлення положення. Її робота не створювала надмірного навантаження на систему, що дозволяло досягати високої частоти оновлення карти навіть на звичайному комп’ютері.

Коли ж середовище ускладнювалось, зокрема за рахунок наявності кількох схожих зон, які робот проходив повторно, стало помітно, що не всі алгоритми однаково ефективно справляються з «впізнаванням» місцевості. Так, при русі замкненими траєкторіями алгоритм із підтримкою інерціальних даних виявив кращу здатність коригувати власну помилку, зберігаючи узгодженість між побудованою картою і фактичним плануванням сцени.

Особливо цікавими були результати в умовах, де втрата орієнтації могла статися через брак ознак у середовищі. У таких випадках візуальний алгоритм, що працював із камерою, демонстрував вразливість: зниження якості зображення або брак текстурованих поверхонь впливали на його здатність підтримувати стійке визначення положення. Навіть короткочасна втрата зв'язку з візуальними

орієнтирами могла призвести до помилок, які важко було компенсувати без додаткових сенсорів.

Також варто відзначити, що із зростанням складності середовища почали проявлятись відмінності у споживанні ресурсів. Наприклад, системи з тривимірною побудовою карти швидко накопичували дані, збільшуючи використання пам’яті. У той же час двовимірні алгоритми показували більш стабільну роботу за умови відносної простоти сцени.

Проведене випробування дало змогу побачити не лише якість побудови карти, а й загальну стабільність кожної системи: від здатності швидко реагувати на нові умови — до відновлення після короткочасних збоїв.

3.3 Порівняння точності локалізації та якості мапи

Завершальним етапом дослідження стало системне порівняння результатів, отриманих у ході експериментів. Оцінювання проводилося за двома основними напрямками: наскільки точно кожен алгоритм визначав положення платформи в просторі, та наскільки повною і достовірною була створена ним карта.

Для аналізу положення порівнювалась траєкторія, яку будував алгоритм, з реальним маршрутом, що проходив робот у симуляції. У простих сценах відхилення були мінімальними — різниця становила лише кілька сантиметрів, і жодна з систем не допускала критичних помилок. Проте у довготривалих маршрутах без підтримки стабілізації стало помітно, що система на базі частинок поступово втрачала точність. Її карта починала зсуватися, особливо при кількох обертах або повторному проходженні однакових ділянок.

Натомість алгоритм із інерціальним підсиленням показав здатність зберігати цілісність карти навіть після замикання циклів. Він коригував власну траєкторію, усуваючи накопичену помилку. Це дозволяло не лише підтримувати точне визначення положення, а й оновлювати карту з урахуванням коригувань, що робило її стійкою до похибок руху.

Якість карти оцінювалась візуально — шляхом накладання на еталонну модель. У цьому плані тривимірні рішення забезпечували більше деталей, однак потребували значних ресурсів. У той же час, двовимірна карта, побудована Cartographer, виявилась найкращою за співвідношенням чіткість — швидкість обробки. Візуальний алгоритм показав високу точність, але тільки у випадках наявності достатньої кількості особливостей на поверхнях сцени.

Загалом експериментальне порівняння підтвердило, що при виборі SLAM-системи важливо не лише орієнтуватися на точність, а й враховувати характеристики середовища, якість сенсорних даних та доступність ресурсів. Навіть найточніша система не забезпечить надійної роботи, якщо її параметри не відповідають умовам експлуатації.

РОЗДІЛ 4. ПЕРСПЕКТИВИ ТА УДОСКОНАЛЕННЯ

Системи одночасної локалізації та побудови карт(SLAM) не зупиняється на досягнутому. Хоча вже сьогодні існує досить багато ефективних алгоритмів які здатні забезпечити точну локалізацію робота і побудову детальної карти середовища, попереду ще багато викликів, які потребують рішення. Роботи все частіше впроваджуються у складні середовища – від багатолюдних міських просторів до природних об’єктів зі змінною геометрією. Відповідно, вимоги до гнучкості, масштабованості, надійності та інтелектуальної адаптивності SLAM-систем зростають.

У цьому розділі розглянемо ключові напрямки удосконалення алгоритмів SLAM: від інтеграції глибокого навчання до мультисенсорної обробки, адаптації до енергозберігаючих платформ а також перспективи впровадження колективної навігації. Також окрема увага буде приділена використанню хмарних технологій та створенню самонавчальних систем які дуже актуальні на сьогодення.

4.1 Технологічні виклики та тенденції майбутнього

Упродовж останніх двох десятиліть технологія SLAM пройшла шлях від академічного інтересу та простої ідеї до масового використання у сучасному світі. Сьогодні важко уявити “серйозного” робота без SLAM, технологія отримала дуже широку сферу застосування і зараз використовується у автономному водінні, воєнній робототехніці, сервісній робототехніці, споживчих гаджетах та навіть у технологіях доповненої реальності (AR). Проте, попри значний прогрес у цій галузі, наукова та інженерна спільнота все ще стикається з низкою викликів:

- **Нестабільність у динамічних середовищах:**

Більша частина SLAM-алгоримів передбачають, що навколишнє середовище в якому вони перебувають є статичним. Проте у реальних сценаріях – наприклад, у міських умовах або в середині будівель із людьми – середовище є дуже динамічним. Рухомі об’єкти можуть призводити до значних помилок у локалізації та деформації карти.

- **Проблеми адаптації до освітлення і текстур:**

Для візуальних SLAM-систем(V-SLAM) стабільна робота алгоритму в значній мірі залежить від якості зображення. У темряві, при контровому освітлені, або в текстурно-бідних сцена(наприклад, білі стіни, туман) звичайні детектори ознак (ORB, FAST) втрачають свою ефективність. Це призводить до проблем з відстеженням та накопичення великої кількості помилок.

- **Обмеження обчислювальних ресурсів:**

Більшість алгоритмів SLAM спочатку розроблялися з урахуванням використання у потужних обчислювальних системах. Проте реальне застосування часто потребує запуску алгоритмів на малопотужних, вбудованих системах – мобільних роботах, безпілотниках, окулярах доповненої реальності. В таких випадках дуже важливими параметрами стають: 1) Низьке енергоспоживання; 2) Оптимізоване використання пам’яті; 3) Швидкість обробки кадрів в реальному часі;

- **Проблема комулятивних помилок та довготривалої стабільності:**

У багатьох системах SLAM спостерігається накопичення помилок при довготривалій роботі, особливо у випадках, коли немає можливості закрити петлі(loop closure). Невеликі помилки локалізації на кожному кроці поступово накопичуються, що призводить до значних зсувів карти, та

серйозних неточностей. Графові SLAM-системи(наприклад, Cartographer) частково вирішують цю проблему за допомогою глобальної оптимізації, однак вони мають високу обчислювальну складність.

На сьогоднішній день усі вище перераховані проблеми вирішують використовуючи такі підходи та технології:

- **Нестабільність у динамічних середовищах:**

Цей виклик часто вирішується через застосування “Симантичного аналізу сцени” – простіше кажучи система виявляє та ігнорує динамічні об’єкти. Для прикладу, можна розгялнути DynaSLAM який використовує згорткові нейронні мережі для ідентифікації людей, транспортних засобів та тварин і виключає їх із процесу побудови карти. Однак навіть з урахуванням цього підходу залишається проблема з об’єктами які змінюють положення у просторі дуже повільно, або значна частина сцени є динамічною.

- **Проблеми адаптації до освітлення і текстур:**

Щоб вирішити цю проблему активно досліджується застосування нейромереж, навчених на великій кількості варіативних сцен. Наприклад, **SuperPoint** та **R2D2** – це фіч-детектори, натреновані на генералізаю між різними умовами освітлення[3]. Такі моделі демонструють набагато кращу стійкість у змінних умовах, однак для їх ефективного використання все ще потрібна потужна апаратна база для швидкого аналізу оточення та освітлення.

- **Обмеження обчислювальних ресурсів:**

Для вирішення проблеми оптимізації на різних по потужностях системах розробляться варіанти SLAM – такі як ORB-SLAM-Lite, Mini-SLAM або RTAB-Map з підтримкою інкрементального картографування та адаптивного зменшення роздільності карти[4].

Також активно використовуються спеціалізовані обчислювальні пристрої: NVIDIA Jetson, Google Coral, Intel Movidius, які поєднують ефективність та низьке енергоспоживання. Такі системи дозволяють запускати повноцінні варіанти SLAM навіть на компактних дронах.

- **Проблема кумулятивних помилок та довготривалої стабільності:**

Для вирішення проблеми кумулятивних помилок та довготривалої стабільності проводяться сучасні дослідження спрямовані на адаптивну глобальну оптимізацію з використанням евристики або попереднього навчання нейронної мережі, яка прогнозує, коли та де варто виконати повторну оптимізацію[6]. Також перспективним підходом у цьому направленні є semantic loop closure, коли замість геометричних ознак об’єкту використовуються семантичні мітки(наприклад, “стіна”, ”двері”, ”стілець”) для визначення вже відвіданих місць навіть при значних змінах сцени.

Також важливим елементом є внесок у технологію SLAM сучасних технологій які якісно розширять та покращать цю технологію. Загальні технологічні тенденції SLAM можна звести до кількох ключових напрямків:

- Інтеграція **штучного інтелекту** для семантичної обробки, адаптивного керування без втручання людини, адаптивне керування ресурсами та предиктивна оптимізація.
- **Модульність систем** – створення фреймворків, які легко адаптуються до конкретної платформи чи задачі.
- **Edge SLAM** – зміщення обчислень до “краю мережі”, без необхідності у хмарному підключенні.
- **Колективна локалізація** – реалізація Multi-Agent SLAM з обміном мапами та локілізацією між платформами.

Очевидно, що SLAM більше не є суто алгоритмічною проблемою. Це задача на стику бачення, навігації, оптимізації, апаратного забезпечення та програмної інженерії. Успішний розвиток технології залежить від гармонійної взаємодії між цими галузями.

4.2 Інтеграція глибокого навчання в SLAM

Через стрімкий розвиток штучного інтелекту та глибокого навчання на початку 2020-х років з'явилася можливість суттєво змінити підхід до задач локалізації та картографування у мобільній робототехніці. Традиційні методи SLAM, побудовані на чітких класичних алгоритмах комп'ютерного зору(детекція ознак, трекінг, тріангуляція, фільтрація), мають обмежену здатність до узагальнення у складних або непередбачуваних умовах. Саме технологія глибокого навчання (Deep Learning) дозволяє подолати ці обмеження за рахунок навчання моделі на великій кількості прикладів, що забезпечує адаптивність до різних середовищ і варіантів шуму в отриманих даних.

Глибоке навчання відкриває можливість якісно покращити роботу SLAM-систем. Ключовими сильними сторонами є такі напрямки:

- **Оцінка глибини:** методи, що використовують згорткові нейронні мережі (CNN), можуть відновити глибину сцени навіть із одного зображення. Це вирішує проблему використання дешевих монокулярних камер замість дорогих стереопар чи лідара. Для прикладу, система CNN-SLAM поєднує класичний монокулярний SLAM із глибокою мережею, яка оцінює глибину у реально часі[7].
- **Семантична сегментація сцена:** мережі типу DeepLab або Mask R-CNN дозволяють уявити об'єкти у середовищі(наприклад люди, транспорт, меблі), що дозволяє відсікати динамічні або не інформативні частини при

побудові карти. У цьому контексті гарно виділяється система DynaSLAM, який використовує сигментацію для створення карти лише із статичних елементів[8].

- **Технології нового покоління:** навчальні детектори особливостей, як-от SuperPoint чи R2D2, виявляють точки інтересу, що є більш стійкими до освітлення, масштабування, обертання і шуму порівняно з ORB чи FAST[8].
- **Передбачення руху та позації:** рекурентні архітектури (RNN, LSTM) або трансформери здатні навчатися за шаблонами руху та використовувати історичні дані для згладжування помилок у локалізації. Такі підходи використовуються в роботах на кшталт DeppVO (Visual Odometry) [9].

4.3 Гібридні підходи

У реальній роботі найкращі результати продемонстрували системи які використовують гібридні підходи. Гібридні системи – це один з найбільш перспективних напрямків розвитку автономної навігації, де в межах однієї архітектури поєднуються різні типи сенсорів, джерел інформації та обчислювальних методик. Вони поєднують у собі класичні методи оптимізації(графи або фільтри) із модулями, побудованими на глибокому навчанні.

Такий підхід дозволяє досягти високої стійкості системи у складних умовах, підвищити точність локалізації, а також забезпечити надмірність – здатність до відновлення після відмови окремого компонента. Гібридність у цьому контексті означає не тільки сенсорне об'єдання (наприклад, камери + IMU), але і об’єднання алгоритмів різного типу, фреймворків, а також режимів роботи.

Наприклад:

- **CodeSLAM:** Побудований на принципі представлення глибини через латентний код, що стикає карту до компактної форми. Це дозволяє передавати карту мережею з мінімальними втратами.
- **DeepFactors:** система, яка будує карту не з набору точок, а з факторів, отриманих за допомогою, нейронної мережі. Таким чином, замість геометричних даних у граф вбудовуються семантичні залежності.
- **Інерціальні фільтри з GPS**: для задач глобальної навігації у відкритому просторі дані IMU доповнюються GPS, що дозволяє калібрувати відносну позицію з глобальним положенням.

Окрім сенсорної, гібридність може мати й **алгоритмічну природу**. Часто SLAM поєднується з іншими технологіями. Такі поєднання алгоритмів якісно піднімають швидкість, точність та здатні позитивним чином впливати на оптимізаю на різних пристроях. Гібридна структура дає змогу також динамічно змінювати **точність і швидкість системи залежно від сценарію**, наприклад, перемикаючись між картографуванням і чистою локалізацією при повторному проходженні знайомого маршруту. Гарними прикладами, які вже довели свою ефективність на практиці можуть виступати:

- **SLAM + Visual Odometry (VO)**: система обирає один з режимів залежно від середовища. Наприклад, у просторі з низькою текстурою — вмикається VO з фільтрацією, у насиченій — SLAM із глобальною оптимізацією.
- **SLAM + глибоке навчання**: поєднуються семантичні карти, згенеровані нейромережею, та геометрична інформація. Система може, наприклад, видалити з карти частини сцени, які мережа розпізнає як динамічні об'єкти (наприклад, автомобілі).

- **SLAM + онлайн-класифікація об'єктів**: деякі системи, як *SemanticFusion*, поєднують візуальний SLAM з реєстрацією та класифікацією об'єктів у реальному часі [3]. Це важливо в сервісній робототехніці, де важливо не лише “бачити” об'єкти, а й знати, що це таке.

4.4 Обмін даними і синхронізація в гібридних системах

Сучасні гібридні SLAM-системи передбачають одночасну обробку даних з кількох джерел — відео, лідара, інерційних сенсорів, ультразвукових датчиків, GPS, а в деяких випадках навіть глибини або тепловізійного каналу. Незалежно від їхньої кількості, дані мають бути зведені до **єдиного часового і просторового контексту** для коректного картографування та локалізації. Саме ця задача — **синхронізація і калібрування сенсорів** — є однією з найбільш технічно складних в мультисенсорних системах. Далі будуть перераховані сучасні проблеми, та як їх вирішують сучасні інженери, які методології використовують та як вони позитивно впливають на існуючі проблеми:

Проблема частот і часових міток

Одна з головних технічних складностей гібридних систем — це синхронізація і калібрування сенсорів. Дані з IMU надходять із частотою до 200 Гц, камера — 30–60 Гц, лідар — 10–20 Гц. Для коректного злиття даних система повинна виконувати часову інтерполяцію, а також просторову трансформацію координат між різними сенсорними системами. Це зазвичай реалізується через розрахунок екструзійної матриці (extrinsic calibration) між сенсорами. Така розбіжність потребує:

1. **часової інтерполяції** — прогнозування/усереднення даних між кадрами;

2. **часового вирівнювання** — переведення всіх сенсорів до єдиного часово-реального масштабованого простору (часовий штамп у UNIX, ROS Time або Hardware Time);
3. **управління затримками** — обробка буферів із сенсорних потоків зі збереженням кадрів у черзі.

У разі невірної синхронізації можлива поява **штучних дрейфів**, “змащених” траєкторій або неправильного накладення фреймів у SLAM, що катастрофічно впливає на локалізацію.

Просторова калібровка між сенсорами

Друга, не менш важлива частина — **екструзійна (extrinsic) калібровка**. Сенсори розміщені у різних точках робота і "дивляться" в різних напрямках. Для того щоб, наприклад, поєднати зображення з камери з даними з лідара, потрібно знати:

1. **трансляційний вектор** — положення одного сенсора відносно іншого (x, y, z);
2. **обертову матрицю (матриця повороту)** — як саме орієнтовані сенсори один відносно одного (Roll, Pitch, Yaw).

У практиці це подається у вигляді **матриці 4x4 (homogeneous transformation matrix)** або пари (R, t) у координатній системі робота. Один з сенсорів (зазвичай IMU) приймається за опорний, а всі інші трансформуються до нього.

Неправильна калібровка може призвести до систематичних помилок у карті (наприклад, паралельні стіни стають кривими) або до неможливості виконати loop closure через незбіг точок карти.

Автоматизовані фреймворки калібрування

Існують кілька потужних фреймворків, що полегшують процес синхронізації та просторової калібровки:

1. **Kalibr** — один із найбільш авторитетних інструментів, підтримує калібрування між камерою та IMU, між декількома камерами, а також обчислення шумових характеристик IMU. Має GUI, працює з ROS-бегами [1].
2. **LiDAR-Camera Calibration Toolkit** — відкритий інструмент, що дозволяє обчислювати екструзійні параметри між RGB-камерою та 3D-лідаром. Підтримує точкову кореляцію через checkerboard або траєкторії [2].
3. **Hand-Eye Calibration** — клас методів, який використовується, коли необхідно калібрувати сенсори, закріплені на рухомій платформі (наприклад, руці маніпулятора). Такі методи засновані на співставленні серій трансформацій між сенсорами та кінематикою [3].
4. **Kalamos** — сучасна альтернатива Kalibr для великих мультисенсорних платформ. Підтримує онлайн-калібрування в реальному часі з використанням фактор-графа [4].

Злиття даних (sensor fusion) у реальному часі

Сучасні SLAM-системи зазвичай реалізують **on-the-fly sensor fusion**, тобто злиття даних "на льоту". Це досягається за допомогою:

- **Буферизації даних**: кожен сенсорний потік записується в окрему чергу;
- **Синхронізації через часові вікна**: беруться фрейми, що максимально збігаються у часі;

- **Фільтрації за змістом**: видаляються або ігноруються кадри з неінформативною інформацією (наприклад, розмиті зображення, шумові лідар-скани).

В деяких системах застосовують **асинхронну обробку**, тобто обробка даних IMU і камер виконується незалежно з подальшим об'єднанням у фактор-графах (як у OKVIS чи VINS-Fusion).

Виклики і відкриті питання

- **Часові дрейфи** між сенсорами з різними годинниками;
- **Динамічні зміни конфігурації** (наприклад, камера повертається або розхитується);
- **Неідеальна апаратна синхронізація** — навіть якщо сенсори мають спільний клок, інтерфейс (USB, I2C, SPI) вносить змінні затримки;
- **Зниження продуктивності при великій кількості сенсорів** — зростає навантаження на RAM і процесор.

Для вирішення цих проблем дослідники розробляють **адаптивні синхронізатори**, які використовують фрейм-буфери, графи залежностей і навіть машинне навчання для визначення оптимального часу синхронізації.

Існують відкриті бібліотеки для мультисенсорного калібрування, як-от *Kalibr* чи *LiDAR-Camera Calibration Toolkit*, які дозволяють налаштовувати гібридні системи для експериментів [4].

РОЗДІЛ 5. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

У цій науково-дослідній роботі розглядається ефективність алгоритмів одночасного створення карти та локалізації (SLAM), що має значне прикладне значення для спеціалістів у сферах автономної навігації, комп’ютерного зору, робототехніки та інтелектуального управління. Технологія SLAM є фундаментальною для навігації мобільних роботів у невідомому оточенні, оскільки дає змогу одночасно орієнтуватися в просторі та формувати його карту. У межах дослідження здійснено зіставлення актуальних SLAM-алгоритмів, з акцентом на їхню точність, стабільність і потребу в ресурсах у різних умовах руху.

Оцінка ефективності SLAM-алгоритмів має першочергове значення для їхнього застосування в реальних робототехнічних системах. Отримані результати можуть сприяти зменшенню похибок позиціювання, поліпшенню точності орієнтації, а також підвищенню ефективності використання енергії та обчислювальних ресурсів. Це дозволяє обґрунтовано впроваджувати відповідні алгоритми в задачах автономного переміщення та зміцнювати надійність і функціональність мобільних роботизованих платформ.

5.1 Розрахунок витрат на виконання БКР

Визначення витрат, пов’язаних із підготовкою бакалаврської кваліфікаційної роботи, проводиться через формування калькуляції кошторисної вартості, яка охоплює науково-дослідний і експериментальний етапи. У структуру цієї калькуляції включено основні категорії витрат, зокрема такі:

- витрати на оплату праці;
- відрахування на соціальні внески;
- витрати на матеріали;
- витрати на використання комп’ютерної техніки;
- інші витрати;
- накладні витрати.

5.1.1 Розрахунок витрат на оплату праці

Ця стаття охоплює витрати на заробітну плату всіх учасників, залучених до виконання БКР, а саме студента-виконавця, наукового керівника та консультанта з питань економіки. Розрахунок здійснюється на основі посажових окладів із врахуванням обсягу виконаних робіт.

На першому етапі визначається середньоденна ставка зарплати для кожного з виконавців (і) обчислюється за формулою:

$$C_{Д_і} = ЗП_і \div F_p \quad (4.1)$$

де Fp – місячний фонд робочого часу (23 дні).

Далі витрати на оплату праці розраховуються за формулою:

$$B_{\text{оп}} = \sum_i^N n_i \times t_i \times \text{С}_{\text{Дi}} \quad (4.2)$$

де n_i – кількість виконавців і-ої спеціальності, які приймають участь у проектуванні; осіб; t_i – час, протягом якого виконавці i-ої спеціальності були залучені до роботи над проектом, днів; $\text{С}_{\text{Дi}}$ – відповідно денна ставка виконавця i -ої категорії, грн/дн.

У таблиці нижче наведено основні вихідні дані, які було використано для подальших розрахунків.

Таблиця 4.1

Вихідні дані для обчислення витрат на оплату праці

№ з/п	Посада виконавців	Час розробки, год.	Середньогодинна ставка, грн/год	Витрати на розробку, грн
1.	Керівник БКР, доцент	18.5	125,30	2318.05
2.	Консультант з економіки, доцент	0,5	125,30	62.65
3.	Студент	165	10,87	1793.55
Разом				4174.25

5.1.2 Відрахування на соціальні заходи

Після визначення витрат на оплату праці наступним етапом є розрахунок єдиного соціального внеску (ЄСВ). Цей внесок нараховується на всю суму заробітної плати, включаючи оплату праці всіх виконавців, залучених до реалізації проекту. Відрахування у державні соціальні фонди здійснюються за ставкою 22% від загального фонду заробітної плати відповідно до чинних норм законодавства.

Розрахунок проводиться за формулою:

$$В_{ЄСВ} = \frac{22}{100} \times В_{оп} \quad (4.3)$$

З урахуванням того, що загальна сума витрат на оплату праці становить 3986.30 грн, розмір нарахувань на соціальні заходи складає:

$$В_{ЄСВ} = 0.22 \times 4174.25 = 918.3 \text{ грн}$$

5.1.3 Розрахунок витрат на матеріали

До цієї категорії відноситься витрати, пов’язані з використанням матеріальних ресурсів, необхідних для завершення БКР. Зокрема, йдеться про витрати на матеріали, необхідні для оформлення та підготовки остаточного варіанту проекту до подання на захист. Тут враховуються орієнтовні витрати на друк повного тексту роботи, виготовлення титульного титульних аркушів, підшивку, а також інші витратні матеріали, які можуть знадобитися для оформлення документації згідно з вимогами.

Формула для обчислення витрат на матеріали має вигляд:

$$B_M = \sum_{i=1}^{n} H_i \times Ц_i \times (1 + К_{ТЗ}), \quad (4.4)$$

де $Н_i$ – кількість одиниць і-го виду матеріалу; $Ц_i$ – вартість однієї одиниці; $К_i$ – коефіцієнт транспортно-заготівельних витрат (приймається у межах 0,10-0,12).

Результати розрахунку $В_м$ наведені в табл. 4.3

Таблиця 4.3

Розрахунок витрат на основні та допоміжні матеріали

№ З/п	Найменування (вид) матеріалу	Одиниця виміру	$В_і$ Од.	$Ц_і$ Грн/од.	$В_{мі}$ грн	$К_{тз}$ (0.10)	Загальна сума
1.	Друк монохромний (формат А4)	шт.	80	4	320	32	352
2.	Папка для паперів	шт.	1	20	20	2	22
Разом							374

5.1.4 Витрати на використання комп’ютерної техніки

До цієї статті належать витрати, пов’язані з експлуатацією комп’ютерного обладнання протягом усього періоду розробки БКР. Сюди входить зношення та амортизація техніки, витрати на споживання електроенергії, а також використання програмного забезпечення. Згідно з розрахунками, прийнятими в обчислювальному центрі Національного університету “Львівська Політехніка”, середня вартість однієї години роботи персонального комп’ютера типу IBM PC/ATX становить 10,5 грн.

У межах даної роботи комп'ютерна техніка використовувалась протягом усього часу розробки, тобто загалом 165 годину. Розрахунок витрат здійснюється за формулою:

$$B_{КТ} = T \times C_{КТ}, \quad (4.5)$$

де $B_{КТ}$ – загальні витрати на використання комп’ютерної техніки, грн; T – загальна тривалість використання , год; $C_{КТ}$ – вартість однієї години експлуатації комп’ютера, грн/год.

Нижче наведено розрахунки витрат на використання комп’ютерної техніки у табл. 4.5

Таблиця 4.5

Розрахунки витрат на використання комп’ютерної техніки

№ З/п	Назва етапів робіт, при виконанні яких використовується комп’ютер	Час використання комп’ютера		Витрати на використання комп’ютера, грн
		днів	годин	
1.	Пошук та аналіз літературних джерел	7	20	210
2.	Тестування алгоритмів	20	45	465.5
3.	Проведення оцінювання якості та аналіз результатів	13	53	556.5
4.	Формування таблиць, графіків, оформлення звіту	10	20	210
5.	Оформлення пояснювальної записки	10	27	283.5
Разом		60	165	1169

5.1.5 Інші витрати

Інші витрати охоплюють витрати, які не були включені до основних статей калькуляції, але виникають у процесі реалізації проекту. До них можуть належати витрати на інтернет-зв'язок, канцелярські товари, витратні матеріали, обслуговування обладнання, програмне забезпечення або інші нематеріальні активи.

У даному випадку фізичне устаткування та програмне забезпечення спеціально для проекту не закуповувались. Використані інструменти, такі як онлайн моделі генерації зображень, були доступні безкоштовно у відкритому доступі. Тому загально застосовано розмір цих витрат у розмірі 10% від витрат на оплату праці. Враховуючи, що витрати на оплату праці склали 3986.30грн, розмір інших витрат становить:

$$В_І = 0.1 \times В_{ОП} = 0.1 \times 4174.25 = 417.4 \text{ грн}$$

5.1.6 Накладні витрати

Накладні витрати включають усі непрямі витрати, які супроводжують організацію роботи над проектом. Сюди належать витрати, пов’язані з управлінням, бухгалтерським супроводом, забезпечення умов праці, доступом до інформаційних ресурсів тощо. В умовах навчального закладу або наукової установи для таких витрат використовується середній відсоток у межах від 1650 до 200% від суми витрат на оплату праці. У межах цієї роботи взято мінімально допустиме значення 150%. Тому накладні витрати будуть становити:

$$В_Н = В_{ОП} \times \frac{150}{100} = 4174.25 \times 1.5 = 6261.3$$

5.1.7 Калькуляція кошторисної вартості виконання БКР

На основі проведених розрахунків за всіма відповідними статтями витрат було складено калькуляцію кошторисної вартості виконання БКР. Зведені дані про повну вартість виконання роботи наведено в табл. 4.6

Таблиця 4.6

Обчислення кошторисної вартості БКР

№ З/п	Статті витрат	Сума витрат, грн
1.	Витрати на оплату праці	4174.25
2.	Відрахування на соціальні заходи	877
3.	Витрати на матеріали	374
4.	Витрати на використання комп'ютерної техніки	1169
5.	Інші витрати	398.63
6.	Накладні витрати	5979.45
Всього		12972.33

5.1.8 Розрахунок договірної ціни та прибутку БКР

Остаточна ціна на проведення науково-дослідної роботи визначається з урахуванням повної кошторисної вартості проєкту, погоджених умов між замовником та виконавцем, а також рівня рентабельності. Така ціна повинна компенсувати витрати на дослідження і водночас гарантувати прийнятний прибуток для виконавця. У цьому випадку для розрахунку використовується рівень рентабельності 20%. Обчислення проводиться за наступною формулою:

$$Ц = К \times (1 + 0.2), \quad (4.6)$$

де К – кошторисна вартість проведення БКР, грн. Відповідно враховуючи попередньо розраховану вартість БКР, договірна ціна складає:

$$Ц = 12972.33 \times 1.2 = 15\,566.8$$

Очікуваний прибуток розраховується як різниця між договірною ціною та повною собівартістю:

$$П = Ц - К = 15\,566.8 - 12\,784.38 = 2782.5$$

З урахуванням тематики дослідження, зазначена вартість є цілком обґрунтованою для ІТ-компаній або наукових організацій, зацікавлених у впровадженні та аналізі алгоритмів Simultaneous Localization and Mapping (SLAM). Представлена робота може слугувати методологічною основою для подальших досліджень у сфері автономної навігації, а також для вдосконалення процесів орієнтації мобільних роботів у складних середовищах.

5.1.9 Оцінка наукової та науково-технічної результативності БКР

Для всебічної оцінки ефективності виконаної науково-дослідної роботи необхідно визначити рівень її результативності. Для цього розраховують коефіцієнт наукової результативності та коефіцієнт науково-технічної результативності. Це дає змогу кількісно оцінити, наскільки дослідження сприяє розвитку науки та має потенціал для подальшого практичного застосування. Наукова результативність відображає приріст нових знань у межах теми дослідження і глибину її опрацювання. Науково-технічна результативність оцінює, наскільки отримані результати можуть бути застосовані для розв’язання прикладних задач або інтегровані в інші наукові чи технічні проєкти. Нижче наведені таблиці 4.7 та 4.8 з вибраними факторами для оцінки обох коефіцієнтів.

Таблиця 4.7

Вибрані показники для розрахунку наукової результативності БКР

Фактор наукової результативності	Коефіцієнт значимості фактора	Якість фактора	Характеристика фактора	Коефіцієнт досягнутого рівня
Новизна отриманих чи прогнозованих результатів	0.4	Середня	Було проведено порівняння сучасних алгоритмів Simultaneous Localization and Mapping з візуальними прикладами; Проведено аналіз ефективності та відміностей роботи.	0.7
Глибина наукового опрацювання	0.4	Середня	Проведений детальний аналіз роботи алгоритмів, ефективності та універсальності у різних сценаріях.	0.6
Міра вірогідності успіху	0.15	Помірна	Усі представлені алгоритми показали гарну ефективність у різних сценаріях. Вірогідність успіху дуже висока.	0.6

Таблиця 4.8

Вибрані показники науково-технічної результативності проведеної БКР

Фактор науково-технічної результативності	Коефіцієнт значимості фактора	Якість фактора	Характеристика фактора	Коефіцієнт досягнутого рівня
Перспективність використання результатів	0.5	Важлива	Результати можуть бути використані у навчальному процесі, подальших дослідженнях у сфері алгоримів SLAM.	0.8
Масштаб можливої реалізації результатів	0.3	Галузевий	Робота є цінною для спеціалістів у сфері робототехніки та комп’ютерного зору, і може бути використана в області штучного інтелекту, цифрової картографії та аналізу якості навігації роботів за допомогою алгоритмів SLAM.	0.8
Завершеність отриманих результатів	0.2	Достатня	У роботі проведено детальний аналіз результатів оцінки якості карт і локалізації,	0.5

			сформульовано висновки та рекомендації щодо практичного застосування алгоритмів Simultaneous Localization and Mapping для навігації роботів.	

На основі вищенаведених даних, можна провести розрахунок узагальнюючих коефіцієнтів наукової та науково-технічної результативності. Нижче наведено формули для розрахунку цих коефіцієнтів:

$$k_{\text{н.р.}} = \sum_{i=1}^{n} \left(k_{\text{зн.і}} \times k_{\text{д.і}}\right), \tag{4.7}$$

$$k_{\text{н.т.р.}} = \sum_{j=1}^{m} \left(k_{\text{зн.}j} \times k_{\text{д.}j}\right), \tag{4.8}$$

де $k_{\text{н.р.}}$ – коефіцієнт наукової результативності; $k_{\text{н.т.р.}}$ – коефіцієнт науково-технічної результативності; $k_{\text{зн}}$ – коефіцієнт значущості фактора; $k_{\text{д}}$ – коефіцієнт досягнутого рівня реалізації цього фактора.

Відповідно розрахунок здійснено наступним чином:

$$k_{\text{н.р.}} = (0.4 \times 0.7) + (0.4 \times 0.6) + (0.15 \times 0.6) = 0.61$$

$$k_{\text{н.т.р.}} = (0.5 \times 0.8) + (0.5 \times 0.8) + (0.3 \times 0.8) = 1.04$$

5.2 Висновки до розділу

У цьому розділі здійснено комплексні розрахунки, що дали змогу всебічно оцінити вартість виконання бакалаврської кваліфікаційної роботи з урахуванням ключових витратних статей. Всі отримані дані були систематизовані у кошторисі, який дозволяє чітко визначити повну собівартість проєкту. Проведено аналіз очікуваного прибутку та обґрунтовано економічну доцільність впровадження результатів дослідження.

Окрім цього, оцінено наукову та науково-технічну результативність проєкту. Розрахункові коефіцієнти засвідчили високий рівень новизни та практичну цінність отриманих результатів, що свідчить про їхній значний потенціал для подальшого застосування. Запропонована методика оцінки якості картографії та локалізації в рамках алгоритмів Simultaneous Localization and Mapping може бути успішно інтегрована у майбутні розробки у сфері робототехніки, навігації та комп’ютерного зору.

Таким чином, економічне обґрунтування підтвердило, що реалізація цієї бакалаврської кваліфікаційної роботи є ефективною, досягає поставлених наукових цілей і характеризується раціональним використанням ресурсів щодо отриманих результатів.

ВИСНОВОК

У ході виконання бакалаврської кваліфікаційної роботи було всебічно проаналізовано ефективність сучасних алгоритмів одночасної локалізації та побудови карти (SLAM), які є ключовими компонентами автономної навігації в мобільних робототехнічних системах. Основною метою дослідження стало порівняння трьох поширених реалізацій: GMapping, Cartographer та ORB-SLAM, що представляють різні підходи до реалізації SLAM — лідарно-параметричний, графовий та візуальний відповідно.

У рамках роботи здійснено ґрунтовний теоретичний огляд технології SLAM, її еволюції, основних класифікацій, типів сенсорів та методів побудови карт. Проаналізовано особливості різних типів реалізацій, визначено сфери їх застосування та основні виклики, пов’язані з динамічними середовищами, обмеженнями обчислювальних ресурсів і якістю вхідних даних. Окрема увага була приділена тенденціям розвитку SLAM, зокрема впровадженню глибокого навчання, гібридним підходам та викликам інтеграції в енергоефективні вбудовані системи.

Практична частина роботи полягала в створенні симуляційного середовища на основі ROS і Gazebo, в якому були протестовані три обрані алгоритми. Кожен з них оцінювався за кількома критеріями: точність локалізації (ATE, RPE), якість побудови карти (RMSE), продуктивність (FPS, затримка) та ресурсомісткість (споживання оперативної пам’яті та процесорного часу). Завдяки повторюваним сценаріям моделювання вдалося досягти об'єктивності в порівнянні та оцінити поведінку кожної системи в умовах різної складності середовища.

Отримані результати підтвердили, що GMapping є ефективним у простих, статичних середовищах і придатним для використання на ресурсно обмежених платформах. Його перевагами є стабільність, швидкодія та низьке споживання

ресурсів. Проте в умовах динамічних змін та відсутності механізму замикання циклів він демонструє значне накопичення похибок. Cartographer забезпечує найкращу точність у побудові карти завдяки потужному механізму глобальної оптимізації та інтеграції даних з IMU, але потребує високих обчислювальних потужностей. ORB-SLAM, що працює з візуальними даними, показав хорошу точність і швидкість у візуально багатих середовищах, але зазнає труднощів у погано освітлених або текстурно бідних умовах.

Проведене дослідження дозволило зробити висновок, що не існує універсального SLAM-алгоритму, який однаково ефективно працює в усіх умовах. Вибір конкретної реалізації повинен базуватися на вимогах до точності, наявному сенсорному обладнанні, характеристиках середовища та обчислювальних можливостях платформи. Найкращі результати досягаються при використанні гібридних підходів — комбінуванні даних з кількох джерел (камера, лідар, IMU), адаптивному налаштуванні параметрів та застосуванні додаткових методів корекції помилок.

Отримані висновки мають практичну цінність і можуть бути використані в проєктуванні та розробці робототехнічних систем для логістики, автономного транспорту, пошуково-рятувальних операцій, а також для дослідницьких і навчальних цілей.

Список використаних джерел

1. Thrun S., Burgard W., Fox D. *Probabilistic Robotics*. – MIT Press, 2005. – 672 p.
2. Cadena C., Carlone L., Carrillo H., Latif Y., Scaramuzza D., Neira J., Reid I., Leonard J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. // *IEEE Transactions on Robotics*, 2016, Vol. 32, No. 6, pp. 1309–1332.
3. Durrant-Whyte H., Bailey T. Simultaneous Localization and Mapping: Part I and II. // *IEEE Robotics & Automation Magazine*, 2006, Vol. 13(2), pp. 99–110.
4. Mur-Artal R., Montiel J.M.M., Tardós J.D. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. // *IEEE Transactions on Robotics*, 2015, Vol. 31(5), pp. 1147–1163.
5. Kümmerle R., Grisetti G., Strasdat H., Konolige K., Burgard W. G2o: A General Framework for Graph Optimization. // *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.
6. Labbé M., Michaud F. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. // *Journal of Field Robotics*, 2019. – Vol. 36(2), pp. 416–446.
7. Hess W., Kohler D., Rapp H., Andor D. Real-Time Loop Closure in 2D LIDAR SLAM. // *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.
8. Grisetti G., Stachniss C., Burgard W. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. // *IEEE Transactions on Robotics*, 2007, Vol. 23(1), pp. 34–46.
9. Qin T., Li P., Shen S. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. // *IEEE Transactions on Robotics*, 2018, Vol. 34(4), pp. 1004–1020.

10. Engel J., Koltun V., Cremers D. Direct Sparse Odometry. // *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018, Vol. 40(3), pp. 611–625.
11. Бобровський Я.В., Івахненко І.М. Основи робототехнічних систем. – К.: Наукова думка, 2020. – 312 с.
12. ROS Wiki. https://wiki.ros.org
13. GitHub – cartographer-project/cartographer. https://github.com/cartographer-project/cartographer
14. TurtleBot3 Simulation on Gazebo. ROBOTIS e-Manual. https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/
15. TUM RGB-D Dataset and Benchmark. https://cvg.cit.tum.de/data/datasets/rgbd-dataset
16. KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/