Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп’ютеризовані та робототехнічні системи
(повна назва)

Кафедра Кафедра комп’ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Другий (магістерський)
(рівень вищої освіти)

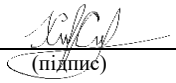
Розробка методу одночасної локалізації і картографування на пересіченій місцевості в режимі реального часу
(тема)

Виконав:
студент 2 курсу, групи КТРСм-22-1
Сухачов К. І.
(прізвище, ініціали)
Спеціальності 151 Автоматизація та комп`ютерно-інтегровані технології
(код і повна назва спеціальності)
Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Комп’ютерні та робототехнічні системи
(повна назва освітньої програми)
Керівник проф. Новоселов С. П.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

2024 р.

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

« 09 » Січня 2024 р.  Сухачов К. І.

(підпис) (прізвище, ініціали)

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет	КТРС
Кафедра	КІТАР
Рівень вищої освіти	другий (магістерський)
Спеціальність	151 Автоматизація та комп`ютерно-інтегровані технології
	(шифр і назва)
Тип програми	Освітньо-професійна
Освітня програма	Комп'ютерні та робототехнічні системи
	(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Сухачову Костянтину Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка методу одночасної локалізації і картографування на пересіченій місцевості в режимі реального часу
Затверджена наказом по університету від 03.11.2023 р. № 1288 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 11.01.2024
3. Вихідні дані до роботи
3.1 Операційна система Ubuntu 22.04.3 (Jammy Jellyfish)
3.2 Операційна система для роботів ROS 2 (Humble Hawksbill)
3.3 Середовища моделювання Gazebo і Rviz
3.4 Розширений фільтр Калмана (EKF)
3.5 Адаптивна локалізація Монте-Карло (AMCL)
3.6 Основна мова програмування Python 3
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз сучасних методів одночасної локалізації і картографування
4.3 Синтез методу гібридного фільтрування AMCL-EKF
4.4 Опис концепції та інструментів фреймворку ROS 2
4.5 Створення віртуального середовища з використанням фреймворку ROS 2
4.6 Результати експериментальних досліджень
4.7 Охорона праці
4.8 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій Презентаційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 34 с. формату А4

6. Консультанти розділів роботи

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз технічного завдання	06.09.23 – 25.09.23	Виконано
2	Аналіз предметної області та останніх досліджень	26.09.23 – 15.10.23	Виконано
3	Розробка програмного забезпечення побудови карт навколишнього середовища	20.10.23 – 27.12.23	Виконано
4	Проведення симуляцій та моделювання	28.12.23 – 07.01.24	Виконано
5	Оформлення пояснювальної записки	08.01.24 – 10.01.24	Виконано
6	Проходження перевірок атестаційної роботи	09.01.24 – 11.01.24	
7	Подання роботи до ЕК	11.01.2024	

Дата видачі завдання 05.09.2023

Студент _____ Сухачов К. І.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Новоселов С. П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 78 с., 39 рис., 2 дод., 32 джерело.

SLAM, ROS2, LIDAR, 2D LRF, NAV2, AMCL, EKF, ІНТЕЛЕКТУАЛЬНИЙ РОБОТ, МЕТОДИ, АЛГОРИТМИ, ЛОКАЛІЗАЦІЯ, КАРТОГРАФУВАННЯ, ВИЗНАЧЕННЯ МІСЦЕЗНАХОДЖЕННЯ, МОДЕЛЮВАННЯ, СИМУЛЯЦІЯ.

Мета роботи – симуляція та моделювання роботи алгоритму побудови карт навколишнього середовища інтелектуального робота і визначення його місцезнаходження та орієнтації у просторі за допомогою пакетів ROS 2 для покрашення точності і стабільності позиціонування, розпізнавання та контролю у невідомій пересіченій місцевості у режимі реального часу.

Об’єкт дослідження – процес одночасної локалізації і картографування в режимі реального часу.

Предмет дослідження – гібридне фільтрування AMCL-EKF для оцінки положення на основі SLAM на ускладненому рельєфі пересіченій місцевості.

Методи дослідження – розширений фільтр Калмана, адаптивна локалізація Монте-Карло, одночасна локалізація та картографування.

Завданнями цієї роботи є синтез поєднання методів SLAM та розробка нових підходів для розв'язання проблем одночасної локалізації і картографування. Для досягнення поставлених завдань був використаний підхід поєднання локальних гібридних частинок-Калманівського фільтрування та глобального відстеження положення на основі SLAM, базуючись на можливостях адаптивної локалізації методу Монте-Карло. Такий підхід розширює технології, не замінюючи існуючі робочі пропозиції, і дозволяє використовувати сучасні методи для всебічного виявлення та розпізнавання навколишнього середовища за допомогою ефективного локалізаційного та картографічного підходу надаючи більш точні результати з використанням менших ресурсів.

ABSTRACT

Explanatory note: 78 pp., 39 figs., 2 app., 32 sources.

SLAM, ROS2, LIDAR, 2D LRF, NAV2, AMCL, EKF, INTELLECTUAL ROBOT, METHODS, ALGORITHMS, LOCALIZATION, MAPPING, POSITION DETERMINATION, MODELING, SIMULATION.

The goal of the work – to simulate and model the operation of an algorithm for constructing maps of the surrounding environment of an intelligent robot and determining its position and orientation in space using ROS 2 packages to enhance the accuracy and stability of positioning, recognition, and control in unknown, challenging terrains in real-time.

The object of study – the process of simultaneous localization and mapping (SLAM) in real-time mode.

The subject of study – the hybrid AMCL-EKF filtering for position estimation based on SLAM in complex, intersected terrain.

Research methods include extended Kalman filter, adaptive Monte Carlo localization, and simultaneous localization and mapping.

The tasks of this work involve synthesizing a combination of SLAM methods and developing new approaches to address simultaneous localization and mapping challenges. To achieve these tasks, a hybrid approach was employed, combining local particle-Kalman filtering and global position tracking based on SLAM, leveraging the capabilities of adaptive localization through the Monte Carlo method. This approach extends existing technologies without replacing current working propositions, allowing for the utilization of modern methods for comprehensive environmental detection and recognition. It employs an efficient localization and mapping approach, providing more accurate results with fewer resources.

ЗМІСТ

Перелік скорочень 9
Вступ 10
1 Аналіз сучасних методів одночасної локалізації і картографування 12
1.1 Концепція SLAM 12
1.2 Математична модель проблеми SLAM 14
1.3 Імовірнісне рішення SLAM фреймворку 15
1.4 Метод лазерного сканування 19
1.5 Методи SLAM одометрії 21
1.6 Об’єднання інформації з декількох сенсорів 22
1.7 Проблематика застосування 23
1.8 Висновки по першому розділу 24
2 Синтез методу гібридного фільтрування AMCL-EKF 25
2.1 Гібридна оцінка положення за допомогою AMCL-EKF 25
2.2 Висновки по другому розділу 29
3 Опис концепції та інструментів фреймворку ROS 2 30
3.1 Обґрунтування вибору фреймворку 30
3.2 Архітектура ROS 31
3.3 Файлова система ROS 32
3.4 Рівень обчислювального графа 34
3.5 Висновки по третьому розділу 37
4 Створення віртуального середовища з використанням фреймворку ROS 2 38
4.1 Створення моделі інтелектуального робота 38
4.2 Налаштування одометрії для інтелектуального робота в ROS 2 42
4.3 Налаштування пакету robot_localization 50
4.4 Налаштування LiDAR для симуляції інтелектуального робота в ROS 2 52
4.5 Висновки по четвертому розділу 55

5 Результати експериментальних досліджень 56
5.1 Обґрунтування вибору Navigation Stack 2 56
5.2 Навігація та SLAM за допомогою навігаційного стеку ROS 2 63
5.3 Висновки по п’ятому розділу 67
6 Охорона праці 68
6.1 Розрахунок освітленості і рівня шуму 69
6.2 Висновки по шостому розділу 73
Висновки 74
Перелік джерел посилань 75
Додаток А Апробації результатів наукових досліджень 79
Додаток Б Програмне забезпечення 102
Додаток В Демонстраційний матеріал 103

ПЕРЕЛІК СКОРОЧЕНЬ

БПЛА – безпілотний літальний апарат;

ПЗ – програмне забезпечення;

ТЗ – технічне завдання;

AMCL – Adaptive Monte Carlo Localization;

EKF – Extended Kalman Filter;

ICP – iterative corresponding point;

IMU – Inertial Measurement Unit;

LiDAR – Light Detection and Ranging;

LO – laser odometry;

NAV 2 – Navigation Stack 2;

9DOF – 9 Degrees of Freedom;

PL-ICP – point-to-line iterative corresponding point;

PSM – Polar Scan Matching;

RBPF – Rao-Blackwellized Particle Filter;

ROS – Robot Operating System;

SLAM – Simultaneous Localization And Mapping;

2D LRF – 2D Laser Range Finder.

ВСТУП

Роботи є передовою технологією, що має великий вплив на різні аспекти нашого суспільства. Вони є універсальними пристроями для відтворення рухових та інтелектуальних функцій людини, що дозволяє передавати їм види діяльності, які для людини трудомісткі, тяжкі, монотонні, шкідливі для здоров'я і життя.

Незважаючи на те, що більшість роботів досі керуються людиною або слідують певній чіткій програмі, з'являється все більше інтелектуальних пристроїв, які можуть виконувати різноманітні складні операції без втручання людини [1].

Особливо актуальною на сьогоднішній день є розробка інтелектуальних роботів з розвиненими можливостями одночасно локалізувати та картографувати навколишнє середовище в режимі реального часу. Це дозволяє здійснювати автономне функціонування інтелектуальних мобільних роботів, зокрема пересування в незнайомій місцевості, в невідомих умов та при наявності певної міри невизначеності даних, які отримуються від навколишнього середовища [2].

Виходячи з вищесказаного, для забезпечення автономного функціонування роботів, зокрема їх пересування в незнайомих місцевостях та у складних умовах, надзвичайно важливе значення набувають системи локації та їх методи локалізації і картографування.

Сьогодні більшість автономних роботів використовують відображення свого навколишнього середовища у 2D або 3D мапах для того, щоб точно визначити своє місцезнаходження та безпечно переміщуватися у фізичному просторі. Однак, однією з основних проблем картографування інтелектуальних роботів полягає в тому, що 2D мапа не містить достатньої інформативності про перешкоди на різних рівнях висот, а для створення 3D мап потрібні значні обчислювальні потужності, багато часу на обробку інформації та великий обсяг пам'яті [3].

У даній роботі пропонується метод гібридного фільтрування AMCL-EKF для локалізації інтелектуального робота, який об'єднує локальні гібридні частинки-Калманівського фільтрування та глобальне відстеження положення на основі

SLAM, базуючись на можливостях адаптивної локалізації методу Монте-Карло (AMCL), який виявляє добру стійкість до непередбачуваних помилок [4]. Метод гібридного фільтрування частинок використовується разом із лазерною одометрією (2D LRF Odom) та інерційною навігацією (IMU) для локальної оцінки положення, яка здійснюється за допомогою розширеного фільтру Калмана (EKF).

В запропонованому методі використовується алгоритм MCL як одне з джерел даних для фільтра Калмана, замість його звичайного використання для глобальної локалізації. Глобальна локалізація базується на техніці Rao-Blackwellized SLAM з використанням інформації про рух, оціненої спостереженнями EKF з лідарного сенсора. Ефективність запропонованого методу була перевірена за допомогою симуляція та моделюванню в програмному середовище ROS 2.

У результаті симуляції та моделюванню експерименту, було виявлено, що запропонований підхід має більш високу точність і стабільність позиціонування, розпізнавання та контролю у невідомій пересіченій місцевості порівняно з іншими методами 2D LRF SLAM. Багато існуючих автономних систем стикаються з такими сценаріями, що обмежують їх можливості для розгортання та збільшують витрати, оскільки середовище має бути ретельно підготовлено, щоб вони могли функціонувати. Досягнуті результати підтверджують надійність розробленого підходу при виконанні завдань в умовах ускладненого рельєфу пересіченій місцевості.

Пояснювальна записка кваліфікаційної роботи виконана з використанням опублікованих автором тез доповіді [2] і публікації статті в науковому журналі «Innovative Technologies and Scientific Solutions for Industries» [3] відповідно до теми кваліфікаційної роботи, згідно державного стандарту України (ДСТУ) 3008-15 [5].

Автор кваліфікаційної роботи був нагороджений дипломом І ступня І туру Всеукраїнського конкурсу студентських наукових робіт з галузей знань і спеціальностей у 2022/2023 навчальному році за міжгалузевим напрямом «Інженерія вбудованих систем» за роботу «Дослідження сучасних методів одночасної локалізації і картографування в режимі реального часу».

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ОДНОЧАСНОЇ ЛОКАЛІЗАЦІЇ І КАРТОГРАФУВАННЯ

Інтелектуальний мобільний робот може визначати своє місце розташування за допомогою апріорно наявної карти простору або шляхом аналізу своїх спостережень [6]. В ідеальному випадку, карту навколишнього середовища можна завантажити у робота завчасно, але на практиці це не завжди можливо.

Тому, виникає задача навчання робота одночасно будувати карту навколишнього середовища та визначати своє місцезнаходження у цьому просторі, розробляючи можливий шлях траєкторії переміщення.

Область знань, що описує методи розв’язання цієї задачі, отримала назву SLAM (Simultaneous Localization And Mapping). SLAM – це задача в області робототехніки, яка вимагає від робота одночасно визначати своє місце в середовищі і побудувати карту цього середовища [7].

1.1 Концепція SLAM

SLAM (одночасна локалізація та картографування) є актуальною темою досліджень і розвитку в галузі робототехніки та комп’ютерного зору. SLAM знаходить широке застосування у різних областях, таких як автономна навігація інтелектуальних роботів, вирішення проблем у розширеній та віртуальній реальності, БПЛА та інших систем. За останні роки SLAM отримав значні досягнення завдяки поступовому розвитку його алгоритмів, використанню новітніх датчиків, а також покращенню обчислювальної потужності комп’ютерів.

Методи одночасної локалізації і картографування (SLAM) – це концепція, яка сполучає два взаємно залежних процеси навігації та побудови карт в єдиний цикл обчислень. Це дозволяє інтелектуальним роботам отримувати дані про навколишнє середовище, створювати карту, визначати своє місцезнаходження та орієнтацію в просторі.

Суть методу полягає в тому, що робот використовує різні датчики (наприклад, камеру, лазерний далекомір, гіроскоп та акселерометр) для збору даних про навколишнє середовище та своє місцезнаходження. Потім він обробляє ці дані, щоб зрозуміти, як він знаходиться у просторі та як навколишнє середовище виглядає.

Варто зазначити, що SLAM це не якийсь конкретний алгоритм, а набір методів та різноманітних засобів, які дозволяють вирішувати задачу щодо визначення місця розташування робота та побудови карти місцевості. На даний момент існує значна кількість методів, що використовують як апаратні, так і програмні можливості. Вибір методу залежить від конкретної задачі та характеристик використовуваних датчиків. Однак, загальною метою всіх цих методів є забезпечення високої точності та швидкості обробки даних з датчиків, щоб дати можливість роботу зорієнтуватися в оточуючому просторі та рухатися в ньому з безпекою та ефективністю.

Основна ідея більшості методів SLAM та їх алгоритмів проста. Перебуваючи у певному положенні, робот починає виявляти об’єкти навколо. Зробивши перші заміри відстаней, він їх запам’ятовує і рухається у напрямку інших об’єктів. Після того, як усі об’єкти в деякому просторі знайдені і відстані до них виміряні, робот повертається на вихідну позицію. Швидше за все, він потрапить в інше місце, а не в те, в якому знаходився спочатку. Відбувається це через помилки одометрії. Далі робот починає друге коло вимірювань, у якому помилка скорочується.

На даний момент найбільш актуальними є методи, засновані на розширеному фільтрі Калмана (EKF) [8] та на фільтрі частинок (метод Монте-Карло) [9]. Основним недоліком розширеного фільтра Калмана є його квадратична обчислювальна складність від кількості об’єктів на карті. Фільтр частинок має логарифмічну складність (залежить від кількості осередків на карті та числа частинок).

Існують дві головні проблеми, які виникають під час вирішення задачі SLAM. Перша – проблема збіжності. Вона безпосередньо пов’язана з точністю обчислень. Будь-які датчики та системи одометрії мають певну модель помилки. Проте, точно

визначити цю модель найчастіше неможливо, тому користуються різноманітних спрощеннями, які тягнуть у себе неточності у побудові карти.

Друга проблема – обчислювальна складність алгоритмів. Частково цю проблему вирішено і на даний момент існують алгоритми, які асимптотично вирішують це завдання за логарифмічний час. Тим не менш, структурна складність навколишнього середовища така, що навіть за такої складності обчислень не завжди вдається вирішувати завдання у прийнятний час.

Додатковою проблемою є те, що завдання SLAM найчастіше постає перед інтелектуальними роботами, для яких питання споживання електроенергії є першорядним, отже з метою енергозбереження розробники інтелектуальних роботів змушені обмежувати обчислювальні потужності використовуваних апаратних платформ.

1.2 Математична модель проблеми SLAM

SLAM – це проблема оцінки. Ми хочемо оцінити як змінну X, яка включає траєкторію або позу робота, так і змінну M, що представляє положення орієнтирів у середовищі. Враховуючи набір вимірювань $\mathrm{Z} = \{z_1, \dots, z_m\}$ і модель вимірювання або спостереження $h(.)$, яка виражає z_k як функцію X і M [10]:

$$z_k = h(\mathrm{X_k}, \mathrm{M_k}) + \epsilon_k, \tag{1.1}$$

де $\mathrm{X_k}$, $\mathrm{M_k}$ – є підмножинами X та M, а ϵ_k – шум випадкового вимірювання.

SLAM має тенденцію розв’язувати проблему максимальної апостеріорної ймовірності (MAP) таким чином, що:

$$\{\mathrm{X}^*, \mathrm{M}^*\} = \underset{\{\mathrm{X,M}\}}{\operatorname{argmax}} p(\mathrm{X}, \mathrm{M}|\mathrm{Z}) = \underset{\{\mathrm{X,M}\}}{\operatorname{argmax}} p(\mathrm{Z}|\mathrm{X}, \mathrm{M}) p(\mathrm{X}, \mathrm{M}), \tag{1.2}$$

де $(\mathrm{Z}|\mathrm{X}, \mathrm{M})$ – це ймовірність вимірювання Z за X і M, а $p(\mathrm{X}, \mathrm{M})$ – це апріорне знання X та M.

Якщо припустити, що спостереження z_k є незалежними, проблема MAP виглядає так:

$$\{\mathrm{X}^*, \mathrm{M}^*\} = \underset{\{\mathrm{X},\mathrm{M}\}}{\operatorname{argmax}} \prod_{k=1}^{m} p(z_k|\mathrm{X}, \mathrm{M}) p(\mathrm{X}, \mathrm{M}) = \\ = \underset{\{\mathrm{X},\mathrm{M}\}}{\operatorname{argmax}} \prod_{k=1}^{m} p(z_k|\mathrm{X_k}, \mathrm{M_k}) p(\mathrm{X}, \mathrm{M}). \tag{1.3}$$

1.3 Імовірнісне рішення SLAM фреймворку

SLAM є рекурсивним процесом оцінювання. Такий процес часто розглядається в імовірнісному вигляді, коли потрібно виконувати класичний крок передбачення та оновлення.

Розглядаючи робота, який рухається в невідомому середовищі [10], визначаємо:

x_k – вектор стану, що описує робота в момент часу k;

$x_{k|k-1}$ – оцінений вектор стану в момент часу k, беручи до уваги знання про попередній стан;

u_k – вектор керування, застосований на $k-1$ для переміщення транспортного засобу в стан x_k (якщо він був передбачений);

m_i – вектор, що описує i-й орієнтир;

$z_{k,i}$ – спостереження i-ї ознаки, зробленої в час k;

X – набір місць розташування транспортних засобів від часу 0 до k;

$U_{0:k}$ – набір керувальних входів від часу 0 до k;

$Z_{0:k}$ – набір спостережень від часу 0 до k;

M – набір орієнтирів або карт;

$M_{k|k-1}$ – оцінювання карти в момент часу k з огляду на попередню карту в момент часу $k-1$.

Оскільки ми розглядаємо ймовірнісну форму *SLAM*, у кожен момент часу k хочемо обчислити функцію розподілу ймовірностей:

$$P(x_k, M|Z_{0:k}, U_{0:k}). \tag{1.4}$$

Щоб продовжити, потрібно застосувати рекурсивний метод, який оперує апріорними даними $P\left(x_{k-1|k-1}, M_{k-1|k-1}\middle|Z_{0:k-1}, U_{0:k-1}\right)$, оновлюваними за допомогою u_k і z_k.

Для цього спочатку необхідно визначити модель руху, що передбачає стан за вхідним сигналом керування $P(x_k|x_{k-1}, u_k)$, таким чином:

$$\begin{gathered} P\left(x_{k|k-1}, M_{k|k-1}\middle|Z_{0:k-1}, U_{0:k}\right) = \\ = \int P\left(x_{k|k-1}|x_{k-1|k-1}, u_k\right) \times \\ \times P(x_{k-1|k-1}, M_{k-1|k-1}|Z_{0:k-1}, U_{0:k-1})\, dx_{k-1|k-1} \end{gathered} . \tag{1.5}$$

Аналогічно також потрібно визначити модель сприймання або спостереження $P(z_{i,k}|X_k, M)$, що пов'язує інформацію датчика щодо виявлення i в час k з оцінкою стану, таким чином:

$$P\left(x_{k|k}, M_{k|k}\middle|Z_{0:k}, U_{0:k}\right) = \frac{P\left(z_{i,k}\middle|x_{k|k-1}, M_{k|k-1}\right)}{P(x_{k|k-1}, M_{k|k-1}|Z_{0:k-1}, U_{0:k})}. \tag{1.6}$$

Початково проблему максимізації апостеріорної ймовірності в алгоритмі SLAM вирішують за допомогою розширеного фільтра Калмана (EKF). Він зменшує невизначеність та надає оцінку на кожному кроці алгоритму. Використовуючи ймовірнісну модель, EKF гарантує збіжність та послідовність карти. Однак він дуже чутливий до помилок асоціації даних, а постійне оновлення всіх орієнтирів та їх матриці коваріації потребує значного обчислювального зусилля. На рис. 1.1 зображено блок-схему процесу EKF-SLAM. Сучасні підходи до вирішення цієї проблеми MAP використовують оптимізаційні методи, такі як Bundle-Adjustment (BA) [11], або глибокі нейронні мережі [12].

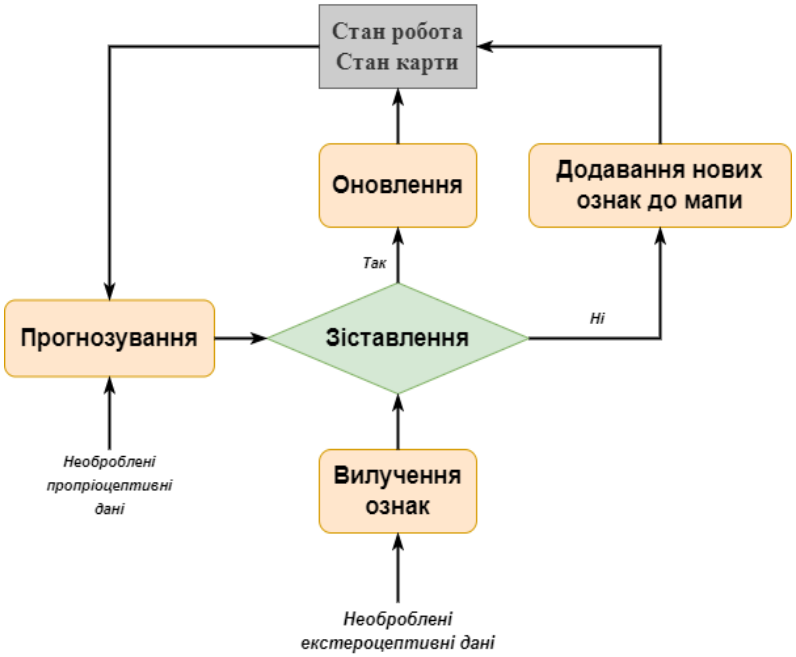


Рисунок 1.1 – Блок-діаграма процесу EKF-SLAM [3]

Коли екстероцептивні дані надходять у час t, стан робота на цей час передбачається за допомогою рівняння (1.5), а виявлені ознаки порівнюються з тими, що є на мапі. Збіги дозволяють оновлювати стан та мапу за допомогою рівняння (1.6). Якщо виявлення немає на мапі, воно ініціалізується, якщо це можливо, і додається до неї. Це виконується рекурсивно.

На рис. 1.2 показано процес SLAM і зображені всі змінні, що використовуються в цьому розділі.

Траєкторія із синіми трикутниками побудована на основі оцінок, а траєкторія із білими трикутниками є правдивою, побудованою на основі реальних фактів.

Синім кольором показана оцінена траєкторія та карта, білим – дані правдивої траєкторії.

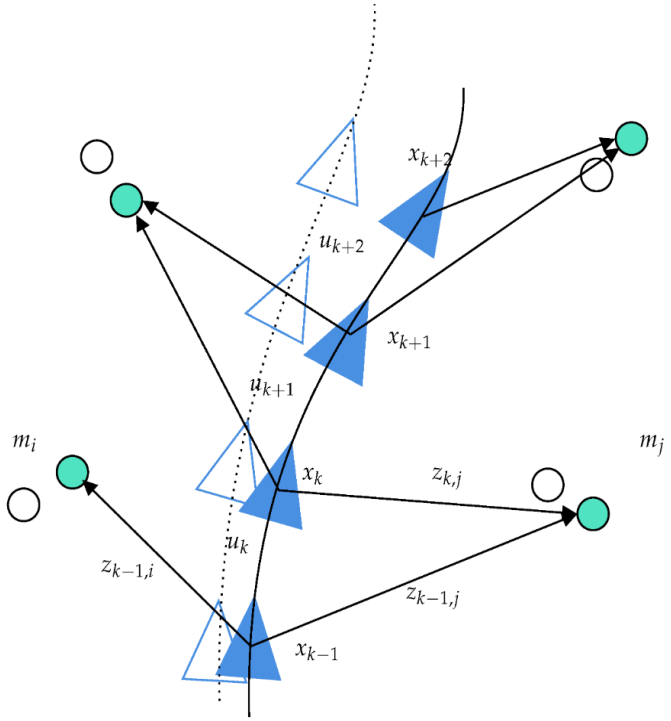


Рисунок 1.2 – Ілюстрація процесу SLAM та позначення [7]

На цьому кроці ми обчислили як крок передбачення, так і крок оновлення, і SLAM може бути представлений як ітеративна оцінка для наступного рівняння, яке є комбінацією (1.5) та (1.6):

$$\begin{gathered} P\left(x_{k|k}, M_{k|k} \middle| Z_{0:k}, U_{0:k}\right) \propto \cdots \\ P\left(z_{k,i} \middle| x_{k|k-1}, M_{k|k-1}\right) \times \ldots \\ \int P\left(x_{k|k-1} \middle| x_{k-1|k-1}, u_k\right) \times \ldots \\ P(x_{k-1|k-1}, M_{k-1|k-1} | Z_{0:k-1}, U_{0:k-1})\, dx_{k-1|k-1} \end{gathered} \quad . \qquad (1.7)$$

Рівняння (1.7) надає рекурсивний Байєсівський метод для реалізації SLAM. Розв’язання проблеми SLAM повинне обробляти відповідні обчислення для моделі руху та моделі сприйняття, щоб ефективно розрахувати рекурсивний метод. Сучасні підходи, як правило, використовують механізацію вимірювання інерційної вимірювальної установки як крок передбачення або припущення про рух транспортного засобу (сталий швидкісний рух, сталий прискорений рух тощо). Щодо моделі спостереження, у випадку візуального SLAM вона часто базується на

моделі зворотного глибини або класичній перспективній моделі огляду. Щодо LiDAR, RGB-D або RADAR підходів, модель спостереження значно спрощується, оскільки спостереження є прямим 2D/3D вимірюванням 3D світу.

$$P2D = \Pi(P3D, K, T). \quad (1.8)$$

де $T = [R, t]$ жорстке перетворення, яке забезпечує 6D позу датчика;

K – внутрішні параметри датчика;

$\Pi(.)$ – функція перспективної проекції.

Як видно, цю функцію потрібно перевернути, щоб відповідати класичній моделі спостереження ($P3D = \mathrm{g}(P2D, K, T)$). Однак така інверсія не є простою, тому крок оцінювання часто відкладається до додаткового спостереження $P2D$. Потім виконується тріангуляція $P3D_{map}$.

1.4 Метод лазерного сканування

LiDAR може виявляти відстань до перешкод, і це є найкращим датчиком для побудови сіткової карти, яка відображає структуру та перешкоди на площині руху робота. У ранніх дослідженнях SLAM часто використовували LiDAR як основний датчик.

Зменшення похибок локалізації є предметом багатьох досліджень протягом багатьох років. Похибки можна поділити за їхньою природою на систематичні та несистематичні. Систематичні похибки легше зменшити, оскільки цей вид помилок передбачуваний за часом і значенням. Несистематичні походять в основному від особливостей робочого простору і представлені ковзанням, занесенням або іншими непередбачуваними результатами контакту колеса з поверхнею. Одним із найважливіших внесків у цю область є фільтр Гауса від Р.Е. Калмана [13]. Техніка фільтра Калмана широко використовується для калібрування одометрії, але ефективне використання даних одометрії вимагає візуальних спостережень, що призводить до великого навантаження на обчислювальні блоки. Крім того, оскільки

фільтр Калмана використовує лінійний перехід стану та вимірювання, його не можна безпосередньо використовувати для розв’язання дискретних задач. Тому було розроблено серію розширень, які лініаризують нелінійні дискретні системи. Одним з них є Розширений фільтр Калмана (EKF) [14], який наразі є одним із найбільш використовуваних у мобільній робототехніці, особливо в SLAM. Крім того, рішення фільтрації SLAM, яке базується на застосуванні Розширеного фільтра Калмана (EKF), є першим успішно впровадженим і найчастіше використовуваним алгоритмом Online SLAM.

Розширений фільтр Калмана (EKF) використовувався для оцінки положення робота, але продуктивність не є ідеальною. Для деяких сильно нелінійних систем цей метод приведе до більшої кількості помилок відрізання, що може призвести до неточного позиціонування та картографування.

Фільтри сімейства Калмана мають здатність використовувати дані з різних джерел (сенсорів) та ефективно об’єднувати їх у точні оцінки, і це є однією з їх найцінніших особливостей. Найбільшим недоліком фільтрів сімейства Калмана є залежність від попередніх станів, що може призвести до значного зменшення середньої похибки в разі несподіваного вимірювального катастрофи. Ще один підхід або парадигма SLAM базується на фільтрах частинок. Фільтри частинок, так само як і фільтри Калмана, базуються на правилі Байеса з Марковськими припущеннями, але, на відміну від них, вони природно дискретні та використовують набір M частинок, кожна i-та з вагою важливості $w^{(i)}$ для визначення правильної оцінки положення [4]. Фільтри частинок ефективні та швидкі в глобальній локалізації, де вони в основному працюють, але можуть використовувати дані лише з одного домену. Широке використання фільтрів частинок для SLAM прийшло з бліскучої ідеї введення методу Rao-Blackwellisation в звичайний алгоритм частинок. Підхід Rao-Blackwellized в фільтрах частинок представлений широким спектром різних реалізацій, включаючи оптимізований для сіткових карт Gmapping запропонований Grisetti у 2007році, який використовується як основа або посилання для наших поточних та попередніх досліджень. Він ґрунтується на покращеному Rao-Blackwellized фільтрі частинок

(RBPF), який покращує точність позиціонування та зменшує обчислювальну складність шляхом покращення запропонованого розподілу та адаптивної методіки перевибірки [4].

Як ефективну альтернативу ймовірнісним підходам, останніми роками популярні оптимізаційні методи. У 2010 році Kurt Konolige запропонував такий представницький метод, як Karto-SLAM, який використовує розріджене коригування положення, щоб вирішити проблему безпосереднього матричного розв'язання в нелінійній оптимізації. Hector SLAM [15], запропонований в 2011р. оціює використання методу Гаусса-Ньютона для вирішення проблеми сканування відповідності, цей метод не потребує інформації відометра, але вимагає високоточного LiDAR. У 2016 році Google запропонувала помітний метод під назвою Cartographer, застосувавши лазерний замкнення петлі до підмап та глобальної карти, накопичувана похибка зменшується [8].

1.5 Методи SLAM одометрії

Успішне застосування алгоритмів на основі SLAM стає можливою завдяки використанню точної інформації про рух, яка часто обчислюється за допомогою інкрементальної локалізації. Однією з них є найстаріший, найшвидший і найпростіший метод короткострокової локалізації – колісна одометрія. Існує багато різних методів для зменшення похибок у цьому підході до локалізації, таких як точне відлічування. Нажаль, колісна одометрія, навіть з використанням додаткової інформації від датчиків руху та різних вказаних методів оцінювання, сильно піддавалася несистематичним помилкам і в основному має слабку ефективність в ускладнених або неочікуваних умовах терену.

Іншою методикою, яку називають візуальною одометрією – візуальна одометрія [4] не залежить від контакту з поверхнею колеса, але, оскільки вона базується на камері, вимагає наявності зовнішнього джерела світла. Тому в разі ізольованого робочого середовища під землею цей метод не може бути використаний без додаткових систем освітлення. Для вирішення цієї проблеми

була розроблена інкрементальна методика локалізації, яка базується на лазерних дальномерах – лазерна одометрія (LO) [16]. Під час поточних досліджень щодо ускладненого та ізольованого терену були проаналізовані наступні ефективні методи Лазерної одометрії (LO): один із найнадійніших підходів до лазерної одометрії – ітеративний метод відповідних точок (ICP), який базується на точка-лінія (PL-ICP), швидший, але не такий точний, як метод PL-ICP – підхід Polar Scan Matching (PSM), комбінований алгоритм PL-ICP і PSM, оцінка 2D лазерної одометрії, яка застосовує статичне припущення про світ за допомогою фільтрації динамічних перешкод за допомогою каучійного оцінювача та розраховує рух робота з руху спостережуваних орієнтирів, наданих функцією швидкості датчиків – візуальна одометрія на основі потоку дальності (R2FO), і рішення оцінки положення 3D SLAM – лазерна одометрія та картографування (LOAM). Таким чином, було припущено, що візійна система сенсорів цільової робототехнічної платформи повинна базуватися на лазерних дальномерах, які дозволяють застосовувати інкрементальний підхід лазерної одометрії для локальної оцінки положення в умовах низького рівня освітлення та для виключення будь-яких джерел помилок щодо контакту колеса з поверхнею.

1.6 Об’єднання інформації з декількох сенсорів

Введення даних допоміжних датчиків може покращити стійкість системи SLAM. На даний момент для LiDAR-SLAM та Visual-SLAM найбільш часто використовуються допоміжні сенсори – енкодер та інерціальна вимірювальна одиниця (IMU), які можуть надавати додаткові дані про рух робота. Системи SLAM з такими допоміжними сенсорами зазвичай працюють краще.

Останнім часом, на основі робіт LiDAR-SLAM та Visual-SLAM, деякі дослідники почали проводити дослідження інтеграції цих двох основних сенсорів [17-18]. У [17] автори застосували візуальний одометр для надання початкових значень для двовимірного лазерного методу ітеративного замикання точок (ICP) на невеликому БПЛА та досягли хороших результатів в реальному часі та точності.

У [18] представлено графову структуру на основі SLAM з монокулярною камерою та лазером з припущенням, що стіна перпендикулярна до землі та вертикально плоска. У [19] інтегруються різні передові методи SLAM на основі зору, лазеру та інерціальних вимірювань за допомогою розширеного фільтра Калмана для БПЛА у приміщенні. У [20] презентовано метод локалізації, заснований на співпраці між повітряними та наземними роботами в закритому приміщенні, де RGB-D сенсор та 2D LiDAR прикріплені до БПЛА для створення 2.5D карти висот та його локації. У [21] надається метод оцінки масштабу та корекції дрейфу шляхом поєднання моно лазерного дальномера та камери для моно-SLAM. У [22] була запропонована візуальна SLAM система, яка поєднує зображення, отримані з камери, і розріджену глибинну інформацію, отриману з 3D LiDAR, за допомогою прямого методу. У [23] було виконано фузію EKF на позиції, обчислені модулем LiDAR та модулем зору, і була запропонована покращена стратегія відстеження, щоб впоратися з проблемою відстеження візуальної SLAM, коли відбувається втрата об’єкта. З камерою та LiDAR стають стандартними конфігураціями для роботів, а фузія лазерної та візуальної інформації стане гарячою темою досліджень для SLAM, оскільки вона може забезпечити більш стійкий результат для реальних застосувань [8].

1.7 Проблематика застосування

Загалом, методи LiDAR-SLAM будують карту зайнятості сітки, яка готова для планування маршруту та керування навігацією. Однак для будівництва більших карт потрібне виявлення та корекція замкненого циклу, що не є легким завданням для сіткової карти. Оскільки отримані скануванням дані є двовимірними хмарами точок, які не мають явних ознак і дуже схожі між собою, виявлення замкненого циклу на основі даних сканування безпосередньо часто є неефективним. І цей недолік також поширюється на швидку функцію переміщення, коли робот працює з заданою картою. У навігаційному пакеті, що надається операційною системою

роботів (ROS), робот потребує ручного введення початкової позиції перед автоматичною навігацією та рухом.

З іншого боку, більшість підходів Visual-SLAM створюють карти ознак, які добре підходять для локалізації, але не підходять для планування маршруту. RGB-D або 3D-LiDAR здатні побудувати повну 3D-сцену, але це обмежено використанням через високу вартість обчислення або знаходження. Для споживчих роботів вартість датчиків та оброблювального обладнання є чутливою. Низькоміцні датчики LiDAR стають популярними. Однак, створення надійної системи навігації низької вартості не є легкою роботою. Оскільки низькоміцні датчики LiDAR мають значно гірші характеристики щодо частоти, роздільної здатності та точності, ніж звичайні датчики. Багато попередніх робіт, наприклад [24], вже пропонували методи для покращення точності зведення сканування для таких низькоміцних датчиків LiDAR, проте це працює лише для сусідніх позицій.

Акумулювання похибок може швидко наростати та призводити до невдачі у будівництві більших карт. Знайти ефективне та надійне рішення SLAM та переміщення з низькою вартістю обчислення та знаходження все ще є викликом для комерційно використовуваних сервісних роботів [8].

1.8 Висновки по першому розділу

В результаті виконання першого розділу магістерської кваліфікаційної роботи проведено аналіз предметної області та останніх досліджень методів одночасної локалізації та картографування. Розглянута концепція SLAM. Побудована її математична модель та імовірнісне рішення проблеми. Описані загальні алгоритми лазерного сканування та SLAM одометрії. Визначена проблематика застосування та підхід до об’єднання інформації з декількох сенсорів.

2 СИНТЕЗ МЕТОДУ ГІБРИДНОГО ФІЛЬТРУВАННЯ AMCL-EKF

Оцінка положення на основі гібридної системи AMCL-EKF (Adaptive Monte Carlo Localization – Extended Kalman Filter) використовує обидва алгоритми для покращення точності та надійності локалізації рухомого робота.

AMCL є алгоритмом адаптивної локалізації методом Монте-Карло, який дозволяє роботу адаптуватися до змін у середовищі та уникати помилок, таких як ковзання та занесення. Цей алгоритм використовує фільтр частинок для оцінки положення робота в середовищі, використовуючи збережені зразки стану.

Розширений Фільтр Калмана (EKF) використовується для локальної оцінки положення робота в реальному часі, використовуючи дані від лазерного одометру, інерційної навігації, або інших датчиків. EKF дозволяє обробляти нелінійні залежності між даними і положенням робота.

2.1 Гібридна оцінка положення за допомогою AMCL-EKF

Підходи до оцінки положення на основі SLAM, вирішують завдання локалізації інтелектуального робота в обох локальних та глобальних координатних системах. Локальна координатна система, яка обчислюється за допомогою інкрементальної техніки локалізації в цій роботі, називається «odom», тоді як глобальна, яка посилається на карту робочого середовища, позначається як «map». Однією з основних припущень дослідження є застосування датчика зондування площі з плоского планувальника із можливістю розширення до 3D систем LiDAR, якщо це потрібно для завдань огляду. Завдяки цьому та кінематичним обмеженням цільової робототехнічної платформи, яка представлена колісним роботом, розроблена в даному дослідженні система відстеження положення на основі SLAM базується на методах $X^{(2)}$ [4].

Для вибору основних методів лазерної одометрії (LO) та перевірки їхньої ефективності в тестовому середовищі використовується більш точна 2D LO – техніка PL-ICP та RF2O, які були протестовані в середовищі Rviz і GAZEBO.

Згідно зі зібраними результатами, техніка RF2O показала значно кращу ефективність порівняно з PL-ICP (навіть з підключеним IMU) в тих самих середовищах.

Кожна з технік LO стикається з аналогічними проблемами – дрейфом та помилками орієнтації, помилками асоціації даних, спричиненими динамічними змінами кутів Ейлера φ та ψ на нерівному терені, де вимірювані перешкоди створюють дуже схожі орієнтири, і важко визначити, чи робот стоїть на місці, чи рухається. Для вирішення цих проблем може допомогти відомий та рекомендований інерціальний вимірювальний блок (IMU), який, в цілому, зменшує значення помилок. Оскільки нерівний терен в основному передбачає неструктуровані або пошкоджені робочі простори, там часто присутні непередбачувані динамічні явища, такі як зсув основи робота без втручання чи порушення руху через ковзання та занесення. Оцінка положення на основі AMCL успішно використовувалася для вирішення проблеми ковзання та занесення, де фільтр Калмана не справляється. Це можливо завдяки природі алгоритмів MCL, які вибирають оцінки положення з відомих перешкод, що надає можливість для ефективної глобальної локалізації, де їх зазвичай використовують. Однак алгоритми локалізації Монте-Карло як частина сімейства фільтрів частинок не можуть використовувати різні джерела вимірювань, тому їх застосування вимагає лише інформації про одометрію з одного джерела, тоді як фільтри Калмана можуть працювати з різноманітними вимірюваннями.

Таким чином, в поточних дослідженнях було вирішено використовувати обидва ці підходи для інкрементальної оцінки положення в межах системи «odom». Локальне відстеження положення представлено трьома різними джерелами положення – лазерною одометрією, IMU та AMCL, оціненою за допомогою Розширеного фільтра Калмана (рис. 2.1 – рис. 2.2), що складається з девяти ступенів свободи вимірювань (9DOF) та їхніх коваріацій. Використання фільтра частинок як

джерела інкрементальної локалізації, оціненої за допомогою гауссівського фільтра, замість його заміни або як алгоритму глобальної локалізації, є оригінальним рішенням, вирішення проблем одночасної локалізації та картографуванням.

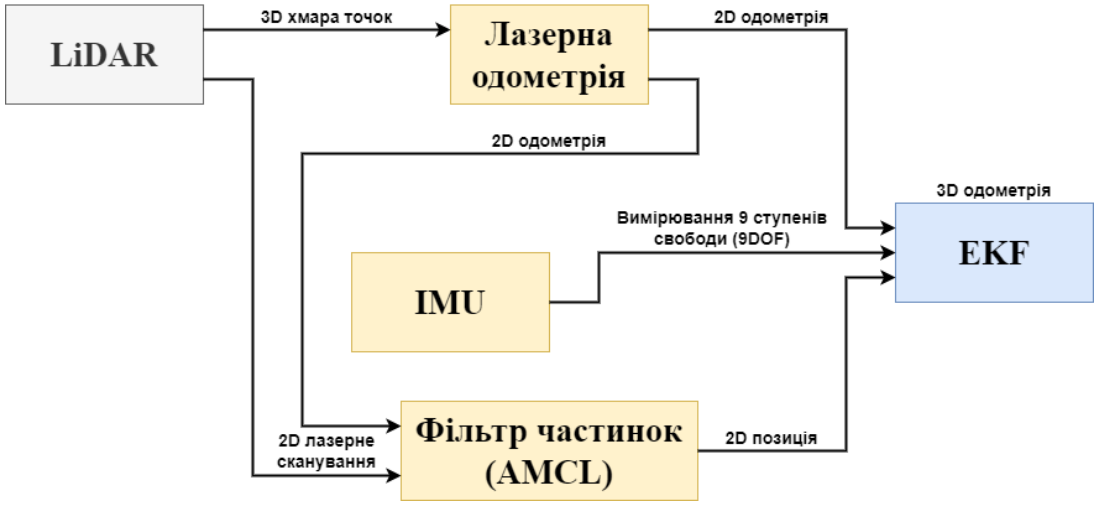


Рисунок 2.1 – Розроблене гібридне оцінювання положення за допомогою AMCL-EKF (комбінована техніка AMCL-EKF для локального положення з трьома ступенями свободи)

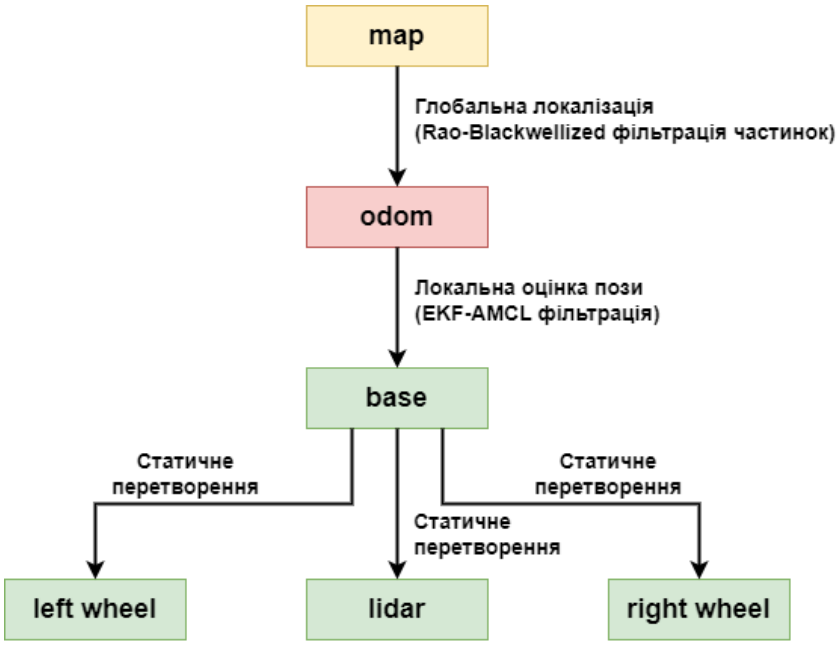


Рисунок 2.2 – Розроблене гібридне оцінювання положення за допомогою AMCL-EKF (зв’язки між координатними системами в підході глобальної локалізації)

Після аналізу різних реалізацій алгоритму EKF, загальний оцінювач, розроблений T. Муром та ін. [25], був обраний як основа для оцінювання положення. Об'єднання даних від кількох датчиків було вирішено налаштуванням вектора конфігурації, де змінні пов'язані з координатами, які обчислюються з кожного введення. Налаштування вектора конфігурації для запропонованого підходу показано на рис. 2.3. Дуже важливою річчю в цьому підході є синхронізація всіх вимірювань, які передаються EKF, інакше буде присутніми стрибки в оцінених значеннях. Це тому, що розрахунки AMCL виконуються після одометрії та після надання даних лазерного скану, тому для несинхронізованого випадку деякі оцінки будуть базуватися на поточних та минулих даних.

Координати відстежень	x	y	z	ϕ	θ	ψ	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{\phi}$	$\dot{\theta}$	$\dot{\psi}$	$\ddot{x}$	$\ddot{y}$	$\ddot{z}$
RF2O	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
IMU	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
AMCL	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Рисунок 2.3 – Локальна оцінка положення: налаштування вектора конфігурації для EKF [4]

Глобальна локалізація вирішується на основі використання фільтра частинок SLAM з реалізацією Rao-Blackwellized від G. Grisetti та ін. [26]. Це рішення обчислює останню локалізацію та 2D сіткову карту зайнятості і разом з вхідними даними від AMCL має хорошу ефективність в сценах, які були нарушені ковзанням та анесенням. Оскільки положення $X^{(3)}$ з 3D даних одометрії, яке оцінюється за допомогою EKF, може бути виражено як $\{X^{(2)}\ z, \varphi, \psi\}$, його використано як інформацію про рух для наступного алгоритму 2D SLAM. Таким чином, розроблене рішення відстеження положення робота базується на двох окремих техніках локалізації, які пов'язані з локальними координатними системами «odom» та глобальними «map». Зв'язки трансформації між всіма координатними системами на платформі рухомого робота показані на рис. 2.2.

2.2 Висновки по другому розділу

AMCL-EKF (Adaptive Monte Carlo Localization with Extended Kalman Filter) є методом гібридного фільтрування, який поєднує два різні підходи до оцінки положення робота, заснований на використанні інформації від SLAM.

Гібридне фільтрування AMCL-EKF об’єднує ці три елементи, використовуючи AMCL для оцінки положення за допомогою частинок та EKF для інтеграції інформації від SLAM. Це дозволяє краще управляти невизначеністю та поліпшує точність оцінки положення робота в реальному часі.

Такий гібридний підхід допомогає уникнути проблем з EKF, пов’язаних із лінійністю та гауссівськими припущеннями, та використовує AMCL для кращої обробки нелінійних та некерованих змін в середовищі.

3 ОПИС КОНЦЕПЦІЇ ТА ІНСТРУМЕНТІВ ФРЕЙМВОРКУ ROS 2

Для реалізації технічного завдання було прийняте рішення використовувати операційну систему для роботів ROS 2. ROS пропонує різноманітні рішення для вирішення проблем SLAM.

ROS – це система метаоперацій з відкритим вихідним кодом, призначена для роботів. Вона надає всі необхідні для операційної системи послуги, включаючи апаратні засоби візуалізації, низькорівневе управління пристроєм, функціональність для загального використання, передачу повідомлень між процесами та пакетами керування [3].

ROS заснований на архітектурі графів, де обробка даних відбувається у вузлах, які можуть отримувати та передавати повідомлення між собою. Бібліотека орієнтована на Unix-подібні системи [27, 28].

ROS має дві основні складові: операційна система ros, яка була описана вище, та набір пакетів ros-pkg, що підтримуються користувачами та організовані у набори, які називаються стеки. Ці пакети реалізують різні функції робототехніки, такі як SLAM, планування, сприйняття, моделювання та інші [27].

3.1 Обґрунтування вибору фреймворку

Існують дві основні версії ROS: ROS 1 та ROS 2, які мають різні функції, переваги та проблеми.

ROS 1 – оригінальна версія ROS, що найбільш широко використовується. Вона була запущена у 2007 році і отримала підтримку великої та активної спільноти розробників та користувачів. ROS 1 заснований на моделі зв’язку «публікація-підписка», в якій вузли (процеси, що виконують обчислення) обмінюються повідомленнями (структурами даних) через теми (іменовані канали). ROS 1 також пропонує послуги (взаємодія запит-відповідь), дії (завдання, що виконуються з зворотним зв’язком) і параметри (загальні значення конфігурації).

ROS 2 – це новіша і просунута версія ROS. Вона була випущена у 2016 році і з того часу швидко розвивається. ROS 2 розроблена для усунення деяких обмежень та проблем ROS 1, таких як масштабованість, продуктивність, безпека та кросплатформова сумісність. ROS 2 заснована на рівні абстракції проміжного програмного забезпечення, що дозволяє використовувати різні протоколи та технології зв'язку, такі як DDS (Служба розповсюдження даних). У ROS 2 також представлені такі поняття, як життєвий цикл вузлів, якість обслуговування та склад.

Однією з основних відмінностей між ROS 1 та ROS 2 є те, що вони не повністю сумісні один з одним. Це означає, що ви не можете запускати вузли або пакети ROS 1 на ROS 2 або навпаки. Однак є деякі інструменти та мости, які можуть допомогти вам перенести код або взаємодіяти між двома версіями.

Найбільш важливим фактором при виборі між ROS 1 та ROS 2 є умови використання та вимоги. Якщо розробляється простий автономний робот, якому не потрібна висока продуктивність, безпека чи надійність, можна вибрати ROS 1, який легше використовувати та розгортати. Однак якщо розробляється складна система, якій необхідно обробляти великі обсяги даних, кілька платформ або виконувати важливі завдання, вибирається ROS 2, який пропонує більшу гнучкість, функціональність і надійність.

У нашій роботі використовується складна робототехнічна система, яка обробляє велику кількість інформації при роботі, тому була обрана друга версія ROS.

3.2 Архітектура ROS

В архітектурі ROS можна виділити три концептуальні рівні:

– рівень файлової системи (Filesystem level);

– рівень обчислювального графа (Computation Graph level);

– рівень спільноти (Community level).

Перший рівень – це рівень файлової системи. На цьому рівні розташована внутрішня структура ROS – структура папок, файли, необхідні для роботи.

Другий рівень – це рівень обчислювального графа, на якому відбувається взаємодія між процесами та системами. На цьому рівні знаходяться концепції та модулі, які є в ROS для створення систем, обробки всіх процесів, комунікації з більш ніж одним комп'ютером тощо.

Третій рівень – це рівень спільноти. Цей рівень містить інструменти та концепції для обміну знаннями, алгоритми та код від будь-якого розробника.

3.3 Файлова система ROS

Першим рівнем в архітектурі ROS є файловий рівень системи (рис. 3.1).

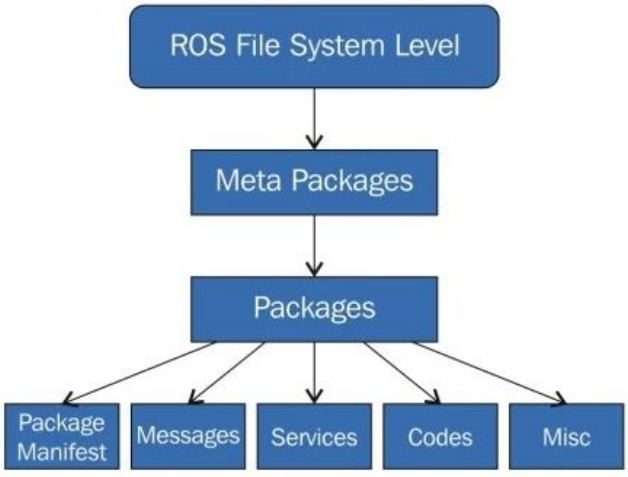


Рисунок 3.1 – Стек файлової системи ROS [30]

Пакети: пакети ROS – це найбільш базова одиниця програмного забезпечення ROS. Вони включають в себе процеси виконання ROS (вузли), бібліотеки, файли конфігурації та інше, організоване як єдиний блок. Пакети – це атомарний елемент збірки та випуску в програмному забезпеченні ROS.

Маніфест пакету: файл маніфесту пакету знаходиться всередині пакету і містить інформацію про пакет, автора, ліцензію, залежності, прапорці компіляції тощо. Файл package.xml всередині пакету ROS – це маніфест цього пакету.

Мета-пакети: термін «мета-пакет» використовується для групи пакетів для спеціального призначення. На старіших версіях ROS, таких як Electric і Fuerte, це називалося стеками, але пізніше це було вилучено на користь простоти та виникнення мета-пакетів. Один з прикладів мета-пакету – це стек навігації ROS.

Мета-пакети маніфесту: мета-пакет маніфесту схожий на маніфест пакету; відмінності в тому, що він може включати пакети всередині нього як залежності під час виконання і вказувати тег експорту.

Повідомлення (.msg): повідомлення ROS – це тип інформації, яка надсилається з одного процесу ROS до іншого. Ми можемо визначити власне повідомлення всередині папки msg в пакеті (my_package/msg/ MyMessageType.msg). Розширення файлу повідомлення – .msg.

Сервіси (.srv): сервіс ROS – це вид взаємодії «запит/відповідь» між процесами. Типи даних для відповіді та запиту можуть бути визначені всередині папки srv в пакеті (my_package/srv/MyServiceType.srv).

Репозиторії: більшість пакетів ROS управляються за допомогою системи контролю версій (VCS), такої як Git, subversion (svn), mercurial (hg) і т. д. Колекцію пакетів, які використовують спільну систему контролю версій, можна називати репозиторіями. Пакети в репозиторіях можуть бути зібрані за допомогою інструменту автоматизації випуску catkin або colcon, в залежності від версії ROS (перша чи друга).

Як і у випадку звичайної операційної системи, програми в ROS розділені на папки, які містять деякі файли, що описують її функціональність (рис. 3.2).

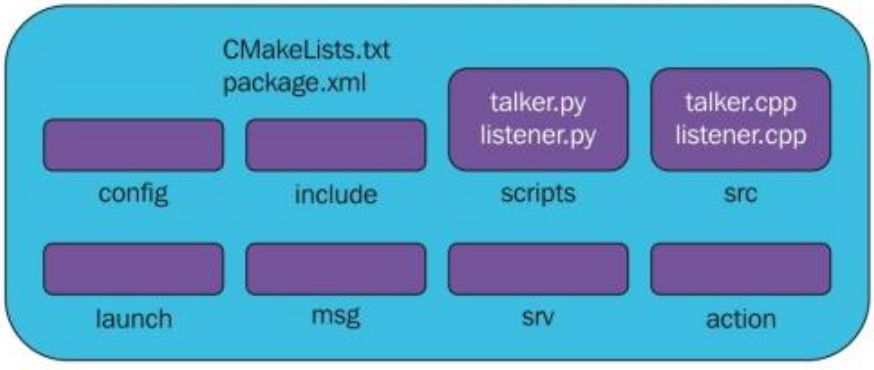


Рисунок 3.2 – Структура типового пакету ROS [30]

Конфіг: усі файли конфігурації, які використовуються у цьому пакеті ROS, розташовані в цій папці. Користувач створює цю папку, і зазвичай її називають config, щоб зберігати в ній файли конфігурації.

Include/package_name: ця папка містить заголовки та бібліотеки, які нам потрібно використовувати всередині пакету.

Scripts: у цій папці розташовані виконувані сценарії Python. На блок-схемі ми бачимо два приклади сценаріїв.

Src: ця папка зберігає вихідні коди. На блок-схемі ми можемо побачити два приклади вихідного коду.

Launch: у цій папці розташовані файли запуску, які використовуються для запуску одного чи декількох вузлів ROS.

Msg: ця папка містить визначення власних повідомлень.

Srv: ця папка містить визначення сервісів.

Action: ця папка містить визначення дії. Ми дізнаємося більше про actionlib у наступних розділах.

Package.xml: це файл маніфесту пакету цього пакету.

CmakeLists.txt: це файл збірки Cmake цього пакету.

3.4 Рівень обчислювального графа

Обчислення в ROS виконуються за допомогою мережі процесів, яку називають вузлами ROS. Ця мережа обчислень може називатися графом обчислень (рис. 3.3).

Основні концепції у графі обчислень включають в себе ROS-вузли, майстер, сервер параметрів, повідомлення, теми, сервіси та Bags (сумки). Кожна концепція в графі призначається для внесення в цей граф різними способами.

Цей стек також містить інструменти, такі як rostopic, rosparam, rosservice та rosnode, для інтроспекції попередніх концепцій.

Стек ros_comm містить пакети проміжного рівня комунікації ROS, і ці пакети загалом називаються ROS Graph layer (шар графа ROS).

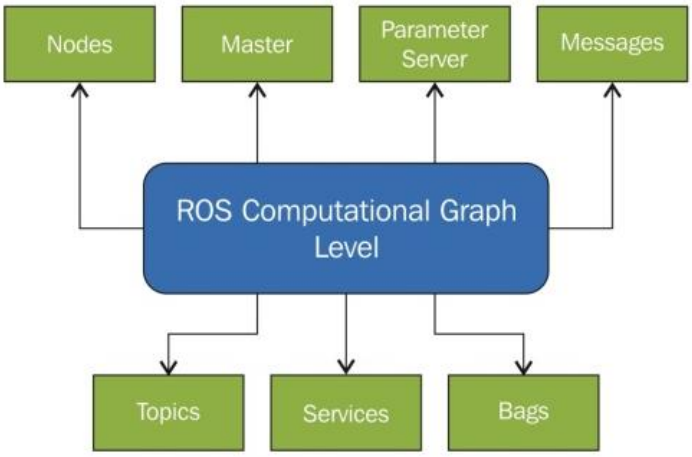


Рисунок 3.3 – Структура слоя ROS Graph [30]

Вузли: вузли – це процеси, які виконують обчислення. Кожен вузол ROS написаний з використанням бібліотек клієнта ROS, таких як roscpp і rospy. За допомогою API бібліотек клієнта ми можемо реалізувати різні типи методів комунікації в вузлах ROS. У роботі може бути багато вузлів для виконання різних видів завдань. За допомогою методів комунікації ROS вони можуть обмінюватися даними між собою. Однією з мет вузлів ROS є створення простих процесів замість великого процесу з усіма функціональностями. Будучи простою структурою, вузли ROS легко відлагоджуються.

Майстер: ROS Master забезпечує реєстрацію і пошук імен для решти вузлів. Без ROS Master вузли не зможуть знаходити один одного, обмінюватися повідомленнями чи викликати сервіси. У розподіленій системі майстер повинен працювати на одному комп'ютері, і інші віддалені вузли можуть знаходити один одного, спілкуючись з цим майстром.

Сервер параметрів: сервер параметрів дозволяє зберігати дані в центральному місці. Всі вузли можуть отримувати доступ до цих значень і модифікувати їх. Сервер параметрів є частиною ROS Master.

Повідомлення: вузли взаємодіють між собою, використовуючи повідомлення. Повідомлення – це просто структура даних, яка містить типоване поле, що може містити набір даних і яку можна відправити іншому вузлу. Є стандартні примітивні типи (цілі числа, десяткові числа, логічні значення і т. д.),

які підтримуються повідомленнями ROS. Ми також можемо будувати власні типи повідомлень, використовуючи ці стандартні типи.

Теми: кожне повідомлення в ROS транспортується за допомогою іменованих шин, які називаються темами. Коли вузол відправляє повідомлення через тему, ми можемо сказати, що вузол публікує тему. Коли вузол отримує повідомлення через тему, ми можемо сказати, що вузол підписується на тему. Публікуючий вузол і підписуючий вузол не відомі один одному. Ми можемо навіть підписатися на тему, яка може не мати жодного видавця. Коротко кажучи, виробництво і споживання інформації декупльовані. Кожна тема має унікальне ім'я, і будь-який вузол може отримати доступ до цієї теми і відправляти через неї дані, якщо у нього є вірний тип повідомлення.

Сервіси: у деяких застосунках роботів модель «публікація/підписка» може бути недостатньою, якщо потрібна взаємодія «запит/відповідь». Модель «публікація/підписка» – це вид односторонньої транспортної системи, і коли ми працюємо з розподіленою системою, нам може знадобитися взаємодія «запит/відповідь». У таких випадках використовуються сервіси ROS. Ми можемо визначити визначення сервісу, яке містить дві частини: одна – для запитів, а інша – для відповідей. За допомогою сервісів ROS ми можемо написати серверний вузол і клієнтський вузол. Серверний вузол надає сервіс під певною назвою, і коли клієнтський вузол відправляє повідомлення-запит до цього сервера, він відповість і відправить результат клієнту. Клієнту може знадобитися зачекати, доки сервер відповість. Взаємодія через сервіс ROS схожа на виклик віддаленої процедури.

Сумки: сумки – це формат для зберігання та відтворення даних повідомлень ROS. Сумки є важливим механізмом для зберігання даних, таких як датчикові дані, які можуть бути важко зібрати, але необхідні для розробки і тестування алгоритмів робота. Сумки є дуже корисними функціями при роботі з складними механізмами робота.

3.5 Висновки по третьому розділу

В результаті виконання третього розділу магістерської кваліфікаційної роботи було проведено аналіз концепції та інструментів фреймворку ROS 2. Було визначено різницю між поколіннями фреймворків ROS 1 та ROS 2 та обраний більш підходящій до розроблюваної роботи. Описали архітектуру та файлову систему ROS. Розібрали інструменти стеку обчислювального графу.

4 СТВОРЕННЯ ВІРТУАЛЬНОГО СЕРЕДОВИЩА З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ ROS 2

Для реалізації роботи було використано операційне середовище Ubuntu 22.04 та версія фреймворка для програмування роботів ROS 2 Humble.

Для візуалізації інтелектуального робота та навколишнього середовища були використані пакети Gazebo та Rviz, які є частиною пакету Robot Operating System. Gazebo дозволяє моделювати фізичні властивості інтелектуального робота, навколишнього середовища, показання різних датчиків тощо. Rviz дозволяє створювати віртуальну модель робота та візуалізувати дані датчиків, карти, маршрути.

Різницю між ними можна підсумувати в наступному уривку з Моргана Квіглі (одного з оригінальних розробників ROS) у його книзі програмування роботів з ROS: «Rviz показує вам, що відбувається, як думає робот, а Gazebo показує, що відбувається насправді» [3].

4.1 Створення моделі інтелектуального робота

URDF (Unified Robot Description Format) – це XML-файл, який описує зовнішній вигляд робота в реальному житті. Він містить повний фізичний опис робота. Створення тіла робота є першим кроком при роботі з Nav2.

URDF-файл містить інформацію про геометрію робота, структуру ланцюгів і ланок, а також може включати інші елементи, такі як сенсори та вузли зчеплення. Цей файл дозволяє системі ROS (Robot Operating System) отримати повний опис робота для використання в симуляціях, візуалізаціях та управлінні.

При роботі з Nav2 (Navigation2), який є пакетом для навігації в ROS 2, створення URDF-файлу є важливим етапом. URDF допомагає системі розуміти фізичні характеристики робота, що є ключовим для навігації та взаємодії з оточенням.

Тіло робота складається з двох компонентів:

– ланки (Links): Ланки є жорсткими частинами робота. Вони представляють кістки робота;

– з’єднання (Joints): Ланки з’єднуються між собою за допомогою з’єднань. З’єднання – це частини робота, які рухаються, забезпечуючи рух між з’єднаними ланками.

Всі роботи складаються з ланок та з’єднань. Ці компоненти дозволяють роботу виконувати рухи та змінювати своє положення в просторі. Розуміння структури ланок та з’єднань важливо для створення URDF-файлу, який описує фізичні властивості робота та використовується в ROS для симуляцій, візуалізацій та управління роботом.

На (рис. 4.1) зображено вигляд ланок та з’єднання симульованого інтелектуального робота з лідаром.

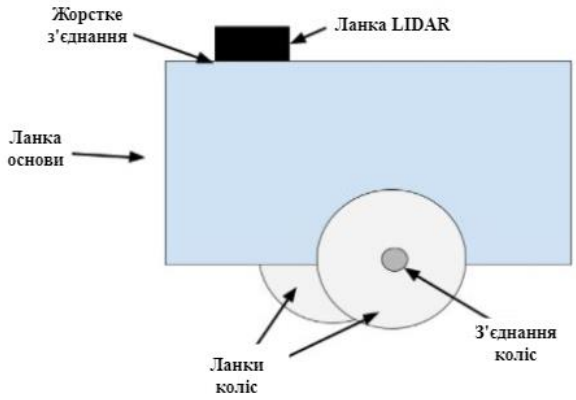


Рисунок 4.1 – Зображення структури ланок та з’єднання симульованого інтелектуального робота

Шарнірні з’єднання коліс – це обертові з’єднання. Обертові з’єднання викликають обертальний рух. Шарнірні з’єднання на рис. 4.1 з’єднують ланки коліс з ланкою основи робота.

Фіксовані з’єднання взагалі не мають руху. ЛІДАР з’єднаний з основою робота за допомогою фіксованого жорсткого з’єднання (тобто це може бути простий гвинт, який з’єднує ЛІДАР з основою робота).

Також бувають призматичні з’єднання. Призматичні з’єднання викликають лінійний рух між ланками (на відміну від обертального руху).

В URDF файлі було визначено кожну ланку та кожне з’єднання симульованого інтелектуального робота (рис. 4.2 – рис. 4.3).

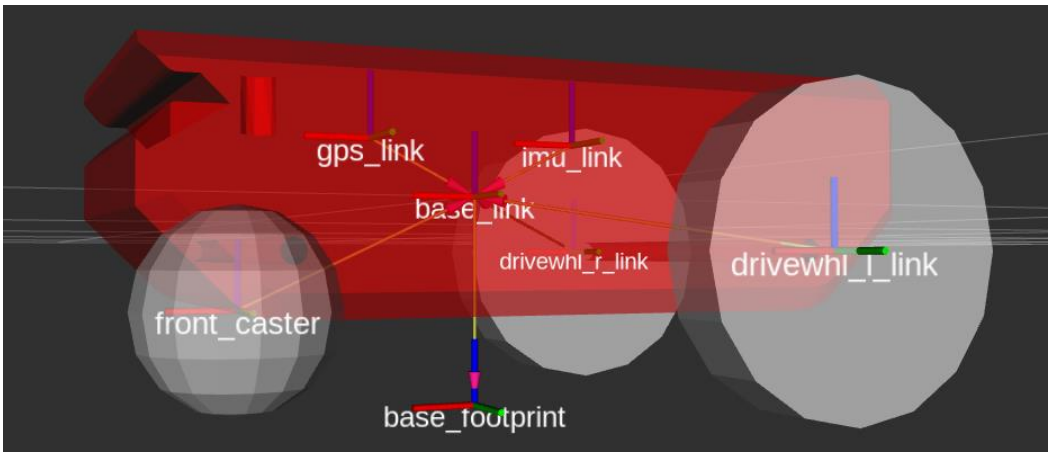


Рисунок 4.2 – Створена модель URDF у фреймворку ROS 2

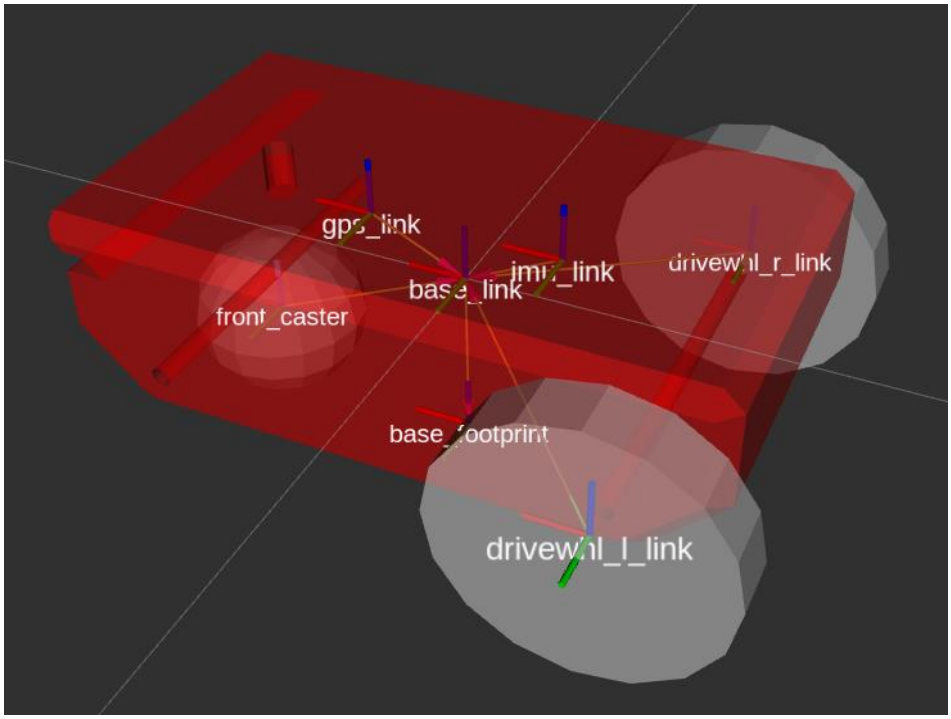


Рисунок 4.3 – Створена модель URDF (Rviz)

Ланки інтелектуального робота включають основу робота (часто називається шасі), два привідних колеса, переднє ведуче колесо, IMU та GPS. Ми також маємо віртуальну ланку, відому як базовий відбиток – ланка безпосередньо під центром робота.

Кожну ланку робота з’єднує шарнір. Є лише два шарніри, які рухаються – обертові шарніри для лівого та правого коліс, які обертаються, приводячи в рух ланки коліс.

У цьому файлі використовується xacro, що означає мову XML macro. Xacro допомагає уникнути наводження в URDF-файлі, дозволяючи визначати строкові константи, які ми можемо використовувати повторно в усьому файлі URDF (наприклад, wheel_radius (радіус колеса)). Усередині цього файлу ми визначаємо, як буде виглядати наш робот (тобто візуальні властивості), як робот буде вести себе при зіткненні з чимось (тобто властивості колізій) і його масу (тобто інерційні властивості).

На (рис. 4.4) можна побачити створене дерево трансформацій координат (tf).

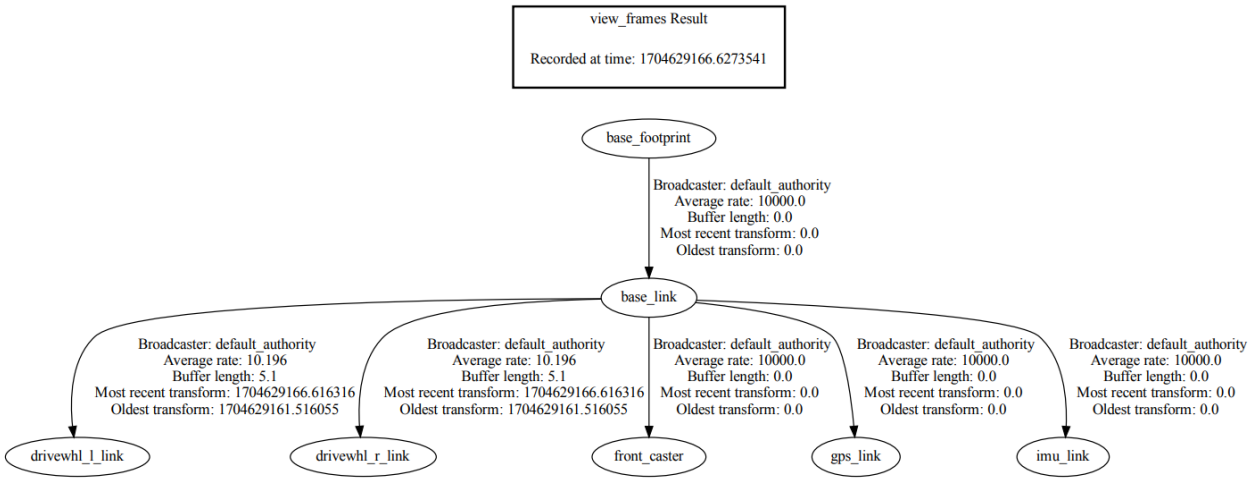


Рисунок 4.4 – Створене дерево трансформацій координат (tf)

На ньому можна побачити механізм для визначення, збереження та оновлення трансформацій між різними системами координат. Він дозволяє

розроблюваній системі обмінювати інформацією про своє положення в реальному часі, що є важливим для робототехнічних застосувань.

На (рис. 4.5) можна побачити створену архітектури нашої системи ROS.

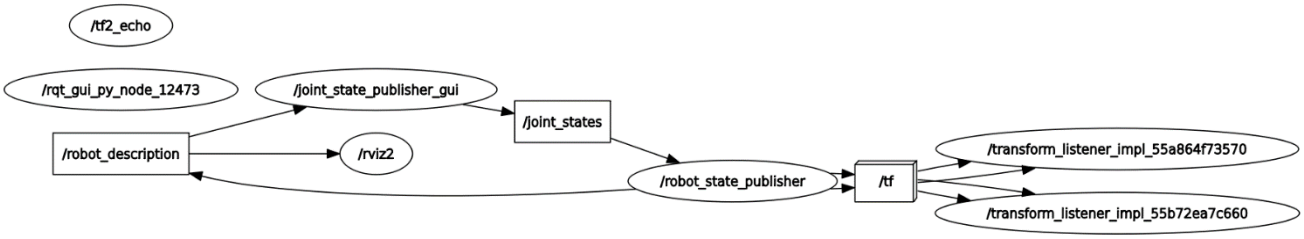


Рисунок 4.5 – Створена архітектура системи ROS (rqt_graph)

На ньому можна спостерігати, як інтерфейс видавця стану спільних об’єктів (Joint State Publisher GUI) маніпулює станами спільних об’єктів (тобто кутами кожного об’єднання колеса). Ці дані подаються до видавця стану робота (Robot State Publisher). Видавець стану робота публікує зв’язки координатних рамок в tf, і оновлена модель робота подається до Rviz – програми візуалізації моделі робота.

4.2 Налаштування одометрії для інтелектуального робота в ROS 2

У робототехніці одометрія – це використання даних від датчиків для оцінки зміни положення, орієнтації та швидкості робота з часом відносно певної точки (наприклад, x=0, y=0, z=0). Інформацію для одометрії зазвичай отримують від датчиків, таких як колісні енкодери, ІНС (інерціальний вимірювальний пристрій) та LiDAR.

У ROS координатна система, яку найчастіше використовують для одометрії, відома як «odom» (одометрична) рамка. Схоже до одометра у автомобілі, який вимірює обертання коліс, щоб визначити відстань, яку пройшов автомобіль від початкової точки, рамка «odom» – це точка в світі, де робот вперше починає рухатися.

Положення та орієнтація робота в межах рамки «odom» стає менш точною з часом і відстанню через те, що датчики, такі як ІНС (які вимірюють прискорення)

та колісні енкодери (які вимірюють кількість обертань кожного колеса), не є ідеальними.

Наприклад робот натрапляє на стіну. Його колеса можуть обертатися безперервно, але робот не буде нікуди рухатися (це називається це «ковзанням коліс»). Одометр коліс вказував би на більшу відстань, пройдену роботом, ніж в реальності. Нам потрібно враховувати ці неточності.

Для корекції неточностей датчиків у більшості застосувань ROS є додаткова координатна рамка, відома як «map», яка надає глобально точну інформацію та коригує дрейф, який виникає в межах рамки «odom».

Стек навігації ROS 2 потребує:

– публікації повідомлень nav_msgs/Odometry на тему ROS 2;

– публікації координатного перетворення від «odom» (батьківська рамка) → «base_link» (дочірня рамка).

Отримання обох цих частин даних опублікованими в системі ROS – кінцева мета в налаштуванні одометрії для робота.

Переформотуємо попередньо створену URDF модель інтелектуального робота в SDF модель.

URDF (Unified Robot Description Format) та SDF (Simulation Description Format) – це два різних формати опису роботів для використання в робототехніці та симуляції роботів. Ось основні відмінності між ними:

а) призначення:

1) URDF: В основному використовується для опису статичних моделей роботів та їхніх кінематичних ланцюгів. URDF зручно використовувати для візуалізації та роботи з реальними роботами;

2) SDF: Зазвичай використовується для симуляційного середовища, де важливо визначити динаміку, сенсори та інші параметри для симуляції реальної роботи.

б) функціональність:

1) URDF: Обмежений у можливостях описувати динаміку та інші деталі, які важливі для симуляції;

2) SDF: Дозволяє більш детально описувати фізичні властивості робота, такі як маса, інерція, сили і т. д.

в) сумісність:

1) URDF: Зазвичай використовується з ROS (Robot Operating System) для взаємодії з реальними роботами;

2) SDF: Використовується в робототехнічних симуляційних середовищах, таких як Gazebo, для віртуальної симуляції роботів.

г) розширюваність:

1) URDF: Менш гнучкий та менш розширюваний порівняно з SDF;

2) SDF: Дозволяє використовувати більш широкий спектр параметрів для реалістичної симуляції.

Таким чином, нам потрібно використовувати файл SDF для речей, пов’язаних з Gazebo, і файл URDF для речей, пов’язаних з ROS. Необхідно створити файл SDF, який якнайточніше відповідав би роботу, створеному за допомогою файлу URDF.

При форматуванні є деякі незначні відмінності між форматами файлів URDF та SDF.

В SDF-файлі видно, що ми використовуємо плагін сенсора IMU для моделювання даних IMU. Замість використання колісних енкодерів для розрахунку одометрії від руху коліс, ми використовуємо диференційний плагін приводу Gazebo та плагін видачі стану з’єднань.

Плагін диференційного приводу буде підписуватися на команди швидкості через тему /cmd_vel та буде публікувати інформацію про одометрію на тему /wheel/odometry.

Робот із диференційним приводом – це мобільний робот, рух якого базується на двох окремо приводжених колесах, що знаходяться по обидва боки тіла робота.

Плагін видачі стану з’єднань буде публікувати кути з’єднань коліс, які постійно змінюються, коли робот починає рухатися.

У реальному проекті з робототехніки для розрахунку системи одометрії ми б використовували сенсор IMU та колісні енкодери. У цьому проекті, ми використовуємо дані колісних енкодерів для створення повідомлення про одометрію [31].

Для більшої реалістичності моделі, добавимо сітку до основи робота. Добавимо файли у форматі STL до папки з мешами. Меш-файл дозволяє роботу виглядати більш реалістично (не просто використовуючи базові форми, такі як коробки чи сфери). Доопрацюємо модель, додавши до неї нову ланку LiDAR зі зв’язками та плагінами для неї, а також створимо інерційну секцію до переднього колеса (рис. 4.6).

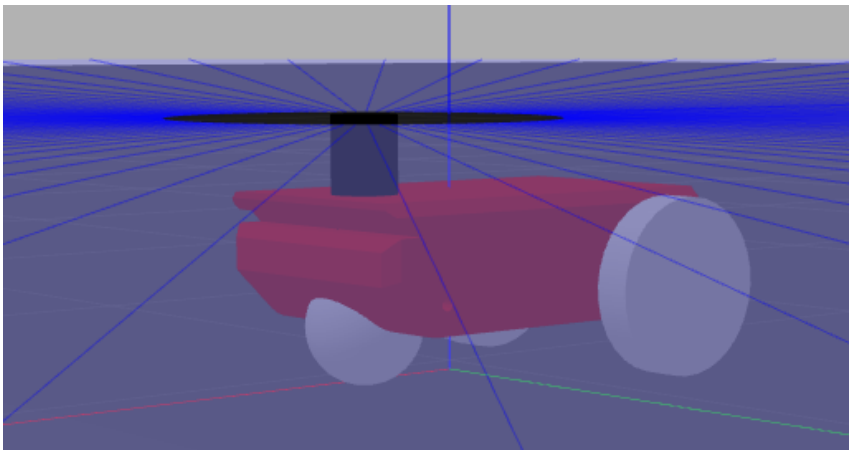


Рисунок 4.6 – Створена модель SDF у фреймворку ROS 2 (Gazebo)

Перевіримо публікацію даних з IMU нашої моделі (рис. 4.7).

```
konstigor@GE75-Raider-9SE:~$ ros2 topic list
/basic_mobile_bot_gps/vel
/clock
/cmd_vel
/gps/fix
/imu/data
/joint_states
/parameter_events
/rosout
/scan
/wheel/odometry
```

Рисунок 4.7 – Дані IMU створеної моделі

Усі дані нашої моделі були опубліковані, модель створена правильно, подивимося поточне положення та орієнтацію робота відносно його вихідної точки (рис. 4.8).

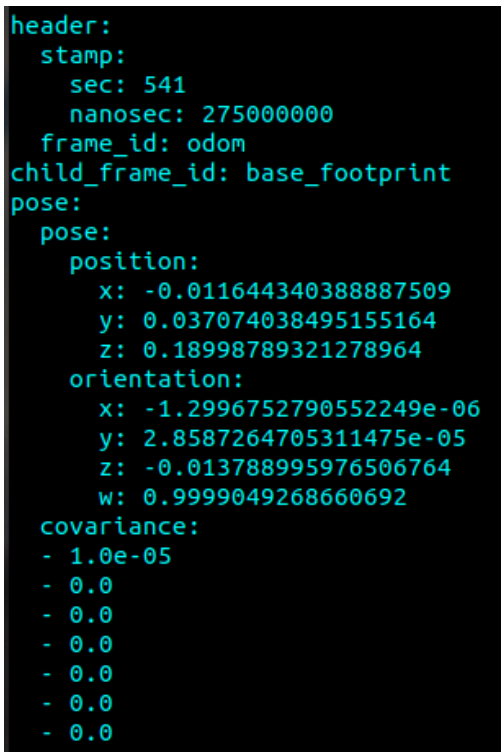


Рисунок 4.7 – Поточне положення робота згідно його вихідної точки

Так само, як файл SDF може бути використаний для визначення вигляду робота, ми можемо використовувати файл SDF для визначення того, як повинна виглядати оточуюча обстановка робота. Оточення нашого робота повинно бути як найреалістичніше.

Створення файлу SDF для всього віртуального середовища (рис. 4.8 – рис. 4.9).

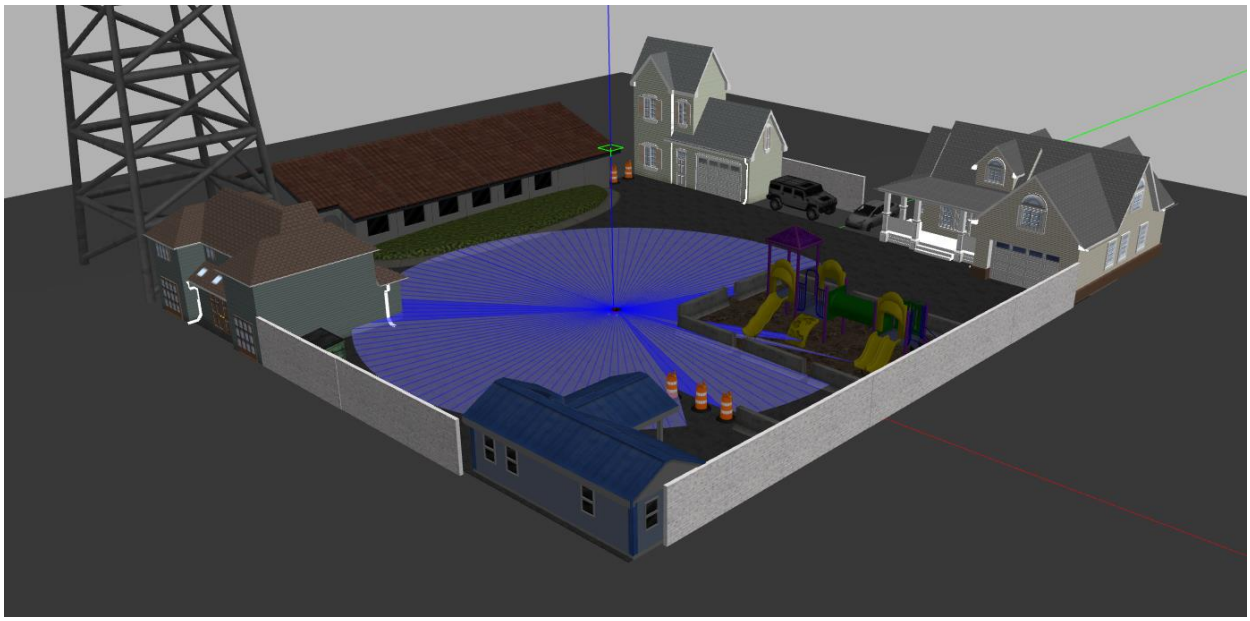


Рисунок 4.8 – Створена SDF модель віртуального середовища (Gazebo) у фреймворку ROS 2

Рисунок 4.8 – Фізичні параметри створеного віртуального середовища

Створимо та скомпілюємо пакет нашої розроблювальної системи (colcon build). «colcon build» – це команда, пов’язана з інструментом Colcon (Cmake/Catkin Command Line Tool), який використовується для збирання, тестування і управління пакетами для проектів, написаних на ROS.

При виклику «colcon build», він ініціює процес збирання розроблювального проекту. Ця команда буде визначати, компілювати і збирати програмне забезпечення, враховуючи всі залежності та конфігурації, які були визначені для проекту. На рис 4.9 представлено результат побудованого проекту.

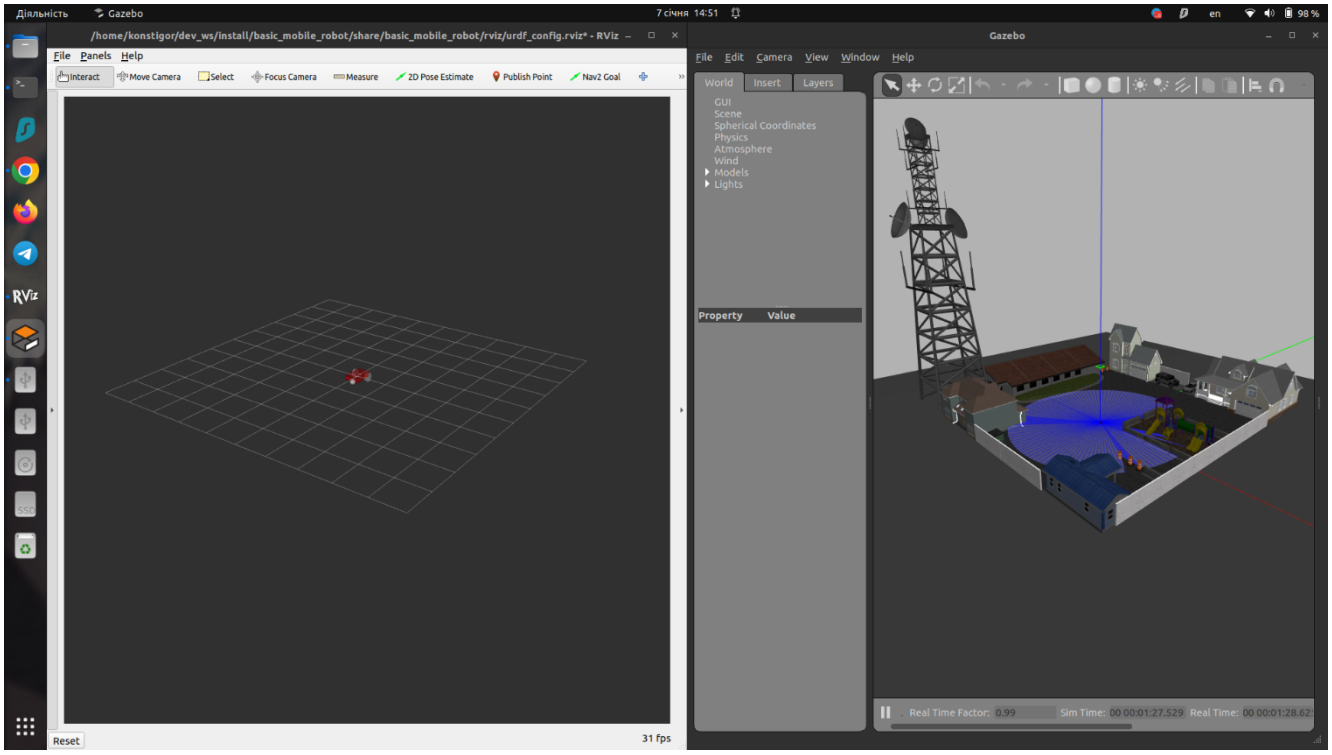


Рисунок 4.9 – Фізичні параметри створеного віртуального середовища

Перевіримо правильність налаштування пакеті «Odom» та «IMU». На рис. 4.10 можна побачити активні теми. Перейдемо по темам /imu/data та /wheel/odometry, що отримати більш детальну інформацію (рис. 4.11).

```
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 topic list
/basic_mobile_bot_gps/vel
/clicked_point
/clock
/cmd_vel
/gps/fix
/imu/data
/initialpose
/joint_states
/parameter_events
/performance_metrics
/robot_description
/rosout
/scan
/tf
/tf_static
/wheel/odometry
```

Рисунок 4.10 – Список активних тем нашої симуляції

```
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 topic info /imu/data
Type: sensor_msgs/msg/Imu
Publisher count: 1
Subscription count: 0
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 topic info /wheel/odometry
Type: nav_msgs/msg/Odometry
Publisher count: 1
Subscription count: 0
konstigor@GE75-Raider-9SE:~/dev_ws$ _
```

Рисунок 4.11 – Робота одометрії та інерційної навігації

Тема /imu/data публікує повідомлення типу sensor_msgs/Imu, а тема /wheel/odometry публікує повідомлення типу nav_msgs/Odometry. Інформація, яка публікується на цих темах, отримується від вбудованих модулів IMU та диференційного приводу Gazebo, які визначені у нашому файлі SDF для робота.

Можна побачити, що обидві теми поки ще не мають підписників. Створемо вузол robot_localization, який буде підписуватися на обидві ці теми, щоб забезпечити злиту, локально точну та плавну інформацію про одометрію для ROS 2 Navigation Stack.

4.3 Налаштування пакету robot_localization

Налаштування пакету robot_localization для ROS 2 на симульованому інтелектуальному роботі [32]. Ми використаємо пакет robot_localization для об’єднання даних одометрії з теми /wheel/odometry і даних IMU з теми /imu/data для надання локально точних та плавних оцінок одометрії. Колеса можуть прослизговувати, тому використання пакету robot_localization може допомогти у корекції цього.

Ми налаштуємо пакет robot_localization для використання фільтра розширеного Калмана (ekf_node) для об’єднання даних від введення датчиків. Ці введення датчиків надходять від плагінів IMU та диференційованого приводу Gazebo, які визначені у нашому файлі SDF.

У реальному робототехнічному проекті, замість симульованих даних ІМП та одометрії, ми використовували б дані від сенсора IMU, такого як BNO055, та від колісних енкодерів, відповідно.

Вузол ekf_node буде підписуватися на наступні теми (типи повідомлень ROS в дужках):

– /wheel/odometry: оцінка положення та швидкості на основі інформації від плагіна диференційованого приводу Gazebo (у реальному робототехнічному проекті це була б інформація, отримана з лічильника обертів колісних енкодерів). Орієнтація в форматі кватерніону. (nav_msgs/Odometry);

– /imu/data: дані від датчика інерціального вимірювання (IMU) плагіна Gazebo (sensor_msgs/Imu.msg).

Цей вузол буде публікувати дані на наступним темам:

– /odometry/filtered: Згладжена інформація про одометрію (nav_msgs/Odometry);

– /tf: Координатне перетворення від рамки odom (батько) до рамки base_footprint (дитина).

Перейдемо по темам /imu/data та /wheel/odometry, щоб переконатися у наявності підписників після налаштування пакету robot_localization (рис. 4.12).

```
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 topic info /imu/data
Type: sensor_msgs/msg/Imu
Publisher count: 1
Subscription count: 1
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 topic info /wheel/odometry
Type: nav_msgs/msg/Odometry
Publisher count: 1
Subscription count: 1
konstigor@GE75-Raider-9SE:~/dev_ws$ _
```

Рисунок 4.12 – Робота одометрії та інерційної навігації після налаштування пакету robot_localization

Переглянемо вивід трансформації від odom → base_footprint (рис. 4.13).

```
konstigor@GE75-Raider-9SE:~/dev_ws$ ros2 run tf2_ros tf2_echo odom base_footprint
[INFO] [1704649488.006736150] [tf2_echo]: Waiting for transform odom ->  base_footprint: Invalid fram
e ID "odom" passed to canTransform argument target_frame - frame does not exist
At time 224.830000000
- Translation: [0.045, 0.003, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.003, 1.000]
- Rotation: in RPY (radian) [0.000, -0.000, 0.006]
- Rotation: in RPY (degree) [0.000, -0.000, 0.369]
- Matrix:
  1.000 -0.006  0.000  0.045
  0.006  1.000  0.000  0.003
  0.000  0.000  1.000  0.000
  0.000  0.000  0.000  1.000
```

Рисунок 4.13 – Вузол трансформації від odom до base_footprint

Будуємо граф вузлів (рис. 4.14) та дерево перетворення координат (рис. 4.15), щоб побачити всі системи координат нашої розроблювальної системи.

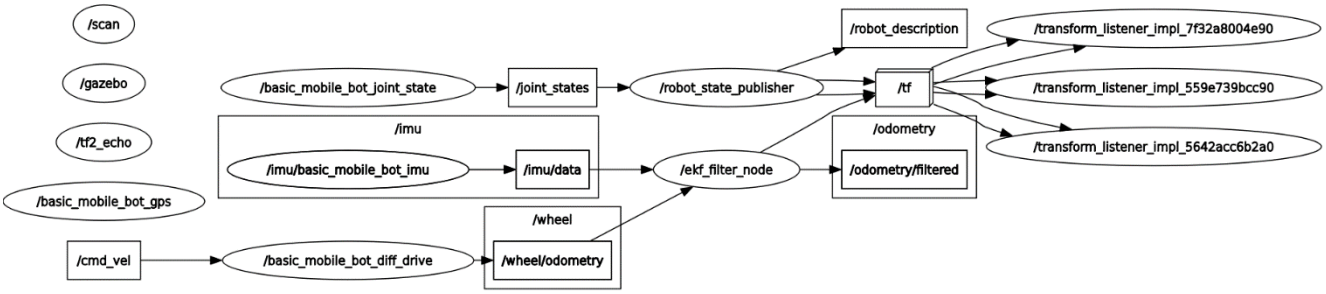


Рисунок 4.14 – Граф вузлів

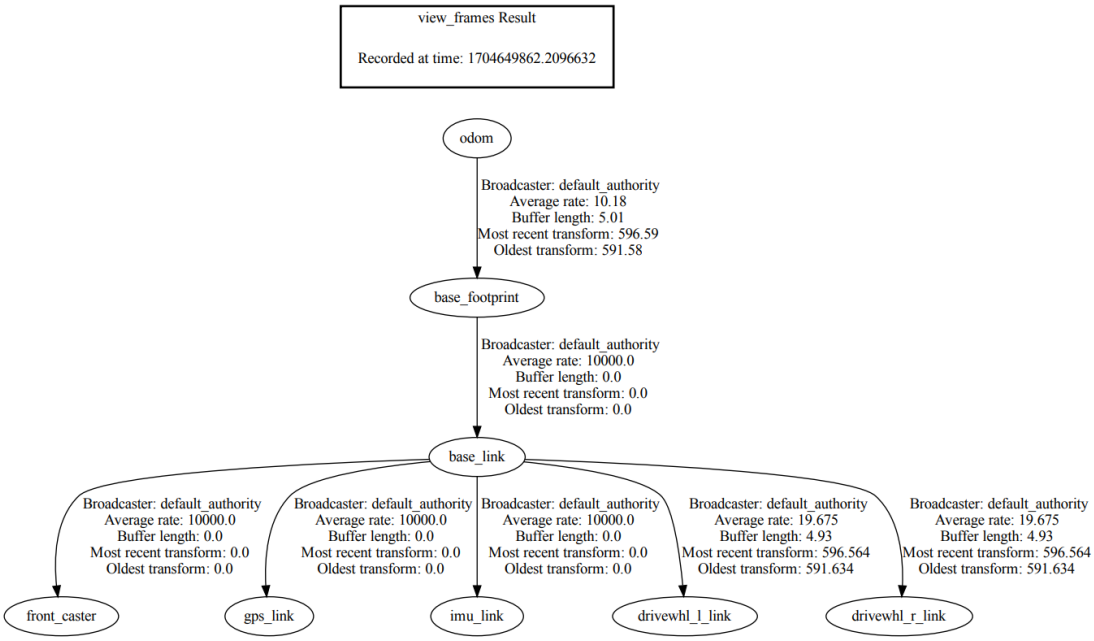


Рисунок 4.15 – Дерево перетворення координат

Зрештою ми отримали синхронізовані світи Rviz і Gazebo віртуального середовища. Модна побачити, що батьківською рамкою є рамка odom. Рамка odom – це початкове положення та орієнтація робота. Кожна інша рамка нижче в ієрархії є дитиною рамки odom.

4.4 Налаштування LiDAR для симуляції інтелектуального робота в ROS 2

Роздивимося вузол /scan (рис. 4.16 – рис. 4.17).

```
konstigor@GE75-Raider-9SE:~$ ros2 topic info /scan
Type: sensor_msgs/msg/LaserScan
Publisher count: 1
Subscription count: 0
```

Рисунок 4.16 – Вузол LiDAR

У цієї теми /scan є лише один видавець (тобто наш LIDAR).

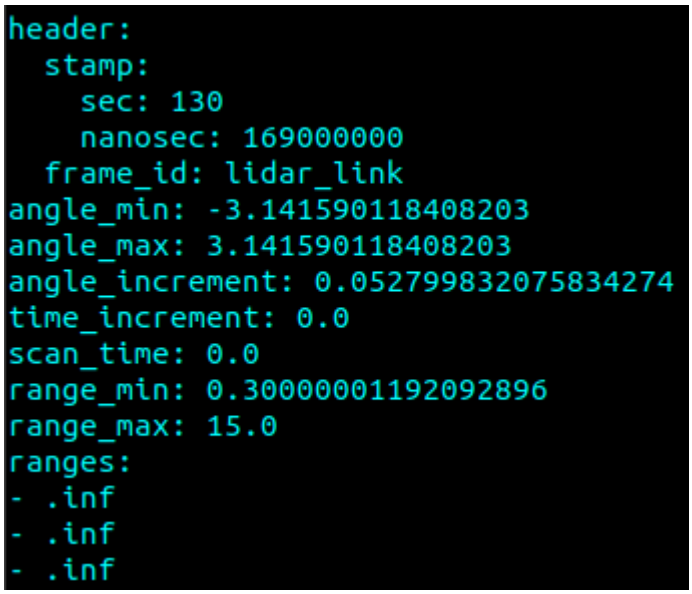


Рисунок 4.17 – Зчитування з вузла свідчення LiDAR

Відліки для всіх 360 променів LIDAR рівні «.inf'», що означає нескінченність. Ці вимірювання мають сенс, оскільки в довкіллі поки немає об'єктів для виявлення. Створимо та скомпілюємо пакет нашої розроблювальної системи.

У вікні Rviz, що відкрився, перевіряємо статуси tf «Transform OK». Візуалізуємо вимірювання сенсора з LiDAR. У Rviz встановлюємо опцію Fixed Frame у Global Options на odom.

Додаємо LaserScan під /scan на вкладці By topic. У випадаючому меню Topic під LaserScan в лівій панелі Rviz встановлюємо Reliability Policy на Best Effort, а розмір на 0,1 метра (рис. 4.18). На рис. 4.19 спостерігаємо результат роботу стеку LiDAR, отримані результати свідчать про правильне налаштування нашої симуляційної среди для проведення експериментів.

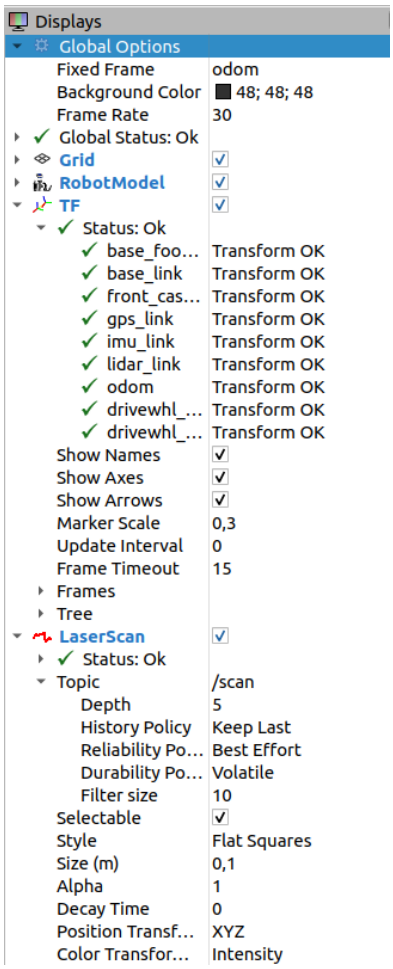


Рисунок 4.18 – Зчитування з вузла свідчення LiDAR

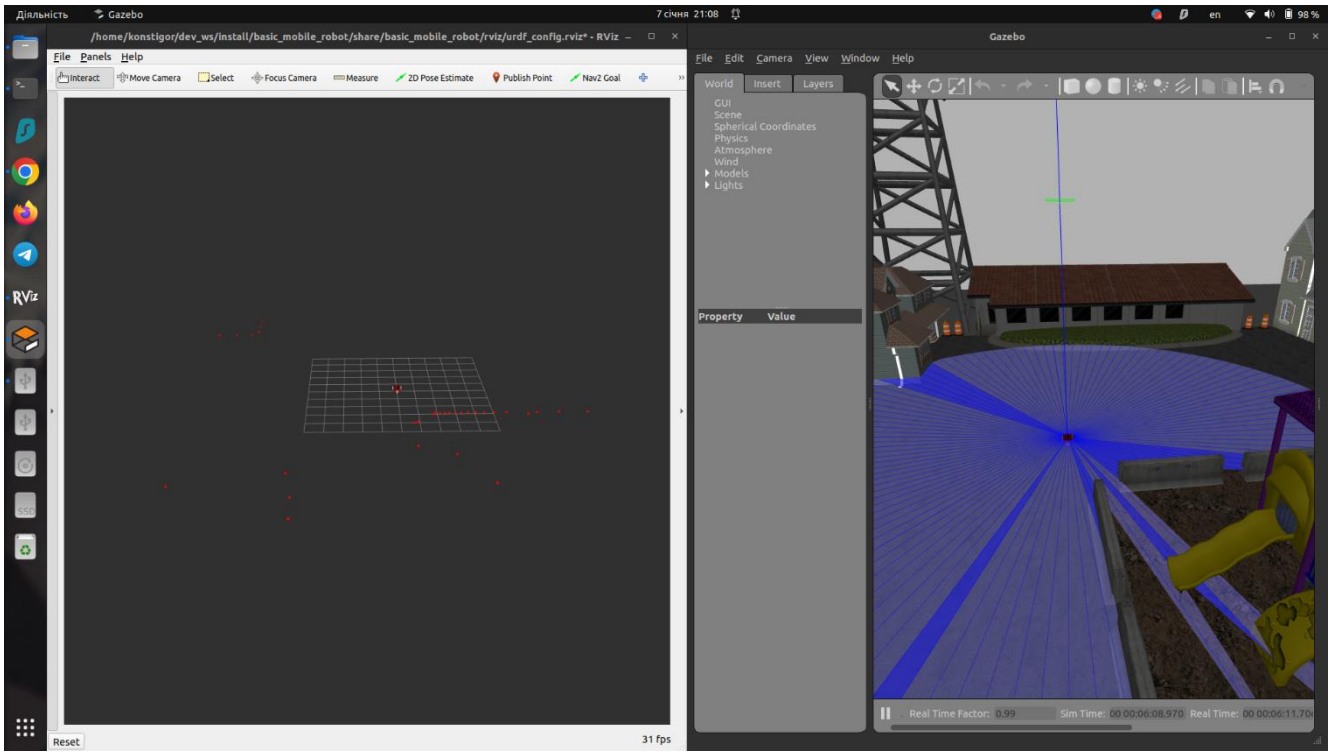


Рисунок 4.19 – Зчитування з вузла свідчення LiDAR

4.5 Висновки по четвертому розділу

В результаті виконання четвертого розділу магістерської кваліфікаційної роботи було створенно віртуальне середовище для проводження експериментів дослідження одночасної локалізації і картографування з використанням фреймворку ROS 2. Були створені URDF та SDF моделі інтелектуального робота, налаштовані пакети, вузли та теми графа.

5 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

Використовуючи інформацію, отриману зі сканів LiDAR, побудуємо карту навколишнього середовища та локалізацію на цій карті. Мета полягає в тому, щоб дозволити нашому роботу автономно навігувати як у відомих, так і у невідомих середовищах (тобто SLAM).

Як зазначено в офіційній документації, два найбільш поширені пакети для локалізації – це пакет nav2_amcl та slam_toolbox. Обидва ці пакети публікують трансформацію координат map → odom, що є необхідним для того, щоб робот локалізувався на карті.

5.1 Обґрунтування вибору Navigation Stack 2

Стек навігації – це набір пакетів, що гармонійно працюють разом у ROS 1 для навігації мобільних роботів. Він приймає інформацію з одометрії, потоків датчиків та команд пози цілі та виводить команди безпечної швидкості на базу нашого мобільного робота (рис. 5.1).

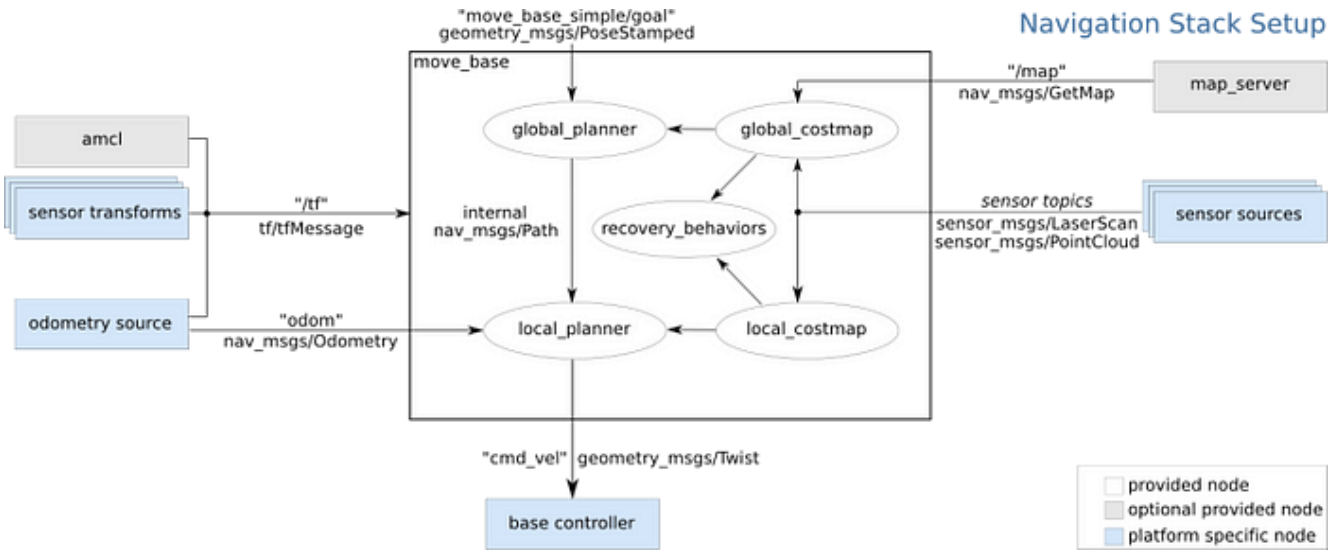


Рисунок 5.1 – Дизайн навігаційного стеку NAV 1 [28]

У стеку навігації є кілька пакетів, що працюють разом. І він переважно працює навколо базового пакету переміщення – центрального мозку нашого навігаційного завдання з усіма кінцевими автоматами, планувальниками, внутрішніми картами витрат та поведінкою відновлення.

У наведеній вище конструкції:

– білі вузли є обов'язковими компонентами, які вже реалізовані;

– сині вузли необхідні та вимагають налаштування для кожної платформи робота;

– сірі вузли вже реалізовані і не є обов'язковими.

Замість того, щоб прямо говорити про те, що знаходиться всередині мозку (рухаємо основу – білий вузол), ми спочатку розглядаємо його як чорну скриньку, а потім розбираємо на частини. Давайте спочатку поговоримо про те, що потрібно цій чорній скриньці.

Джерело одометрії – потрібна інформація про одометрію у вигляді nav_msgs/Odometry, щоб мати відомість про положення та рух робота.

Джерела датчиків – потрібна інформація лазерного скану або хмари точок про середовище для уникнення перешкод під час локального та глобального планування шляху (використовуючи карту вартостей).

Трансформації датчиків – потрібно, щоб робот публікував трансформації між координатними системами. Це включає трансформацію odom → base_link від одометрії (та map → odom, якщо використовується карта).

Контролер бази – потрібен контролер бази робота, який приймає вихідні дані від пакету move base у вигляді команди швидкості geometry_msgs/Twist для робота.

(Необов'язково) amcl – потрібен для трансформації map → odom при використанні карти.

(Необов'язково) сервер карт – потрібен для завантаження карти при використанні.

Отже, ось те, що потрібно нашому центральному мозку. О, також він потребує позначки мети (geometry_msgs/PoseStamped) як зовнішнього спускового пристрою. Як ще він міг би знати положення цілі робота?

Примітка – систему навігації можна ініціалізувати з або без статичної карти. Без неї робот розглядає лише зустрічені перешкоди та оптимістично планує глобальні маршрути для невидимих областей. Він перепланує маршрут пізніше, коли зустрічає більше перешкод.

Фактично move base отримує всю цю інформацію, планує глобальні та локальні маршрути з урахуванням відповідно глобальної та локальної карт вартостей і виводить команди швидкості, поки не досягне мети.

Давайте нарешті розглянемо компоненти всередині move base:

– глобальна карта вартостей – карта вартостей (2D-представлення сітки вокселів) всієї карти для глобального/повнодовжинного планування маршруту. Вона також доповнена інформацією про перешкоди в реальному часі з датчиків;

– локальна карта вартостей – карта вартостей локальної (видимої) області робота для локального/короткого планування маршруту. Ця карта вартостей створюється лише з інформацією про перешкоди в реальному часі з датчиків;

– глобальний планувальник – планувальник, який використовує алгоритм A^* для планування маршруту від поточного положення до кінцевого положення. Він використовується лише як високорівневий посібник для навігації в середовищі. Він отримує глобальну карту вартостей, інформацію про локалізацію робота та цільове положення;

– локальний планувальник – планувальник, який використовує метод Dynamic Window Approach (DWA) для короткострокового планування маршруту. Він використовується для надання фактичних команд швидкості роботу. Він отримує локальну карту вартостей та шлях від глобального планувальника як високорівневий посібник;

– відновлення – поведінка, яка використовується, коли робот застрягає (потенційний збій).

У підсумку, глобальний та локальний планувальники використовують дані від одометрії, інформації від сенсорів, карт вартостей та позначки мети, щоб генерувати команди швидкості для робота.

З усім цим навігаційний стек став сильним гравцем у сфері застосованої мобільної робототехніки.

Фактично, завдяки ефективній 3D-картографії на основі вокселів (моделювання невідомого 3D-простору), він чудово вирішив проблему створення, оновлення та отримання високоточних тривимірних даних сприйняття.

Проблеми з навігаційним стеком в ROS 1:

– пакет move base внутрішньо використовує монолітний і неналаштований становий автомат. Це не надає багато гнучкості розробникам для налаштування на замовлення;

– move base працює лише з роботами із диференційним приводом та голономними колесами;

– move base дозволяє використовувати лише один глобальний та локальний алгоритми планування одночасно. Ви не можете динамічно завантажувати різні плагіни для алгоритмів планування для своєї власної програми.

Введення в навігаційний фреймворк у ROS 2.

ROS 2 був створений для виведення ROS «з лабораторії». NAV 2 базується на ROS 2 та успішній спадщині навігаційного стеку.

Додавши до ROS 2 надійність, безпеку та швидкість, NAV 2 спрямована на вирішення зазначених проблем у навігаційному стеку.

Замість неконфігурованих монолітних станових машин використовується навігація на основі дерева поведінки.

Використовуються незалежні модульні сервери (планувальник, контролер, відновлення), які можуть бути видалені, замінені або розширені власними серверами.

Дозволяється використовувати кілька локальних планувальників траєкторії та шляхів для одного завдання навігації.

Дизайн навігаційного фреймворка в ROS 2. У Nav2 є два важливі патерни дизайну:

– BT навігатор – компонент найвищого рівня, який містить дерево поведінки для реалізації навігаційного поведінки;

– асинхронні сервери для конкретних завдань – кожен є вузлом ROS 2, який містить кілька плагінів алгоритмів для своєї задачі.

Додатково:

– ці вузли ROS 2 є управляєми (з життєвим циклом) вузлами для визначених поведінок кожного сервера;

– фреймворк NAV 2 тепер є цілісним і включає всі пакети, такі як AMCL та map server. Вони не розсипані.

А ось, як виглядає дизайн (рис. 5.2).

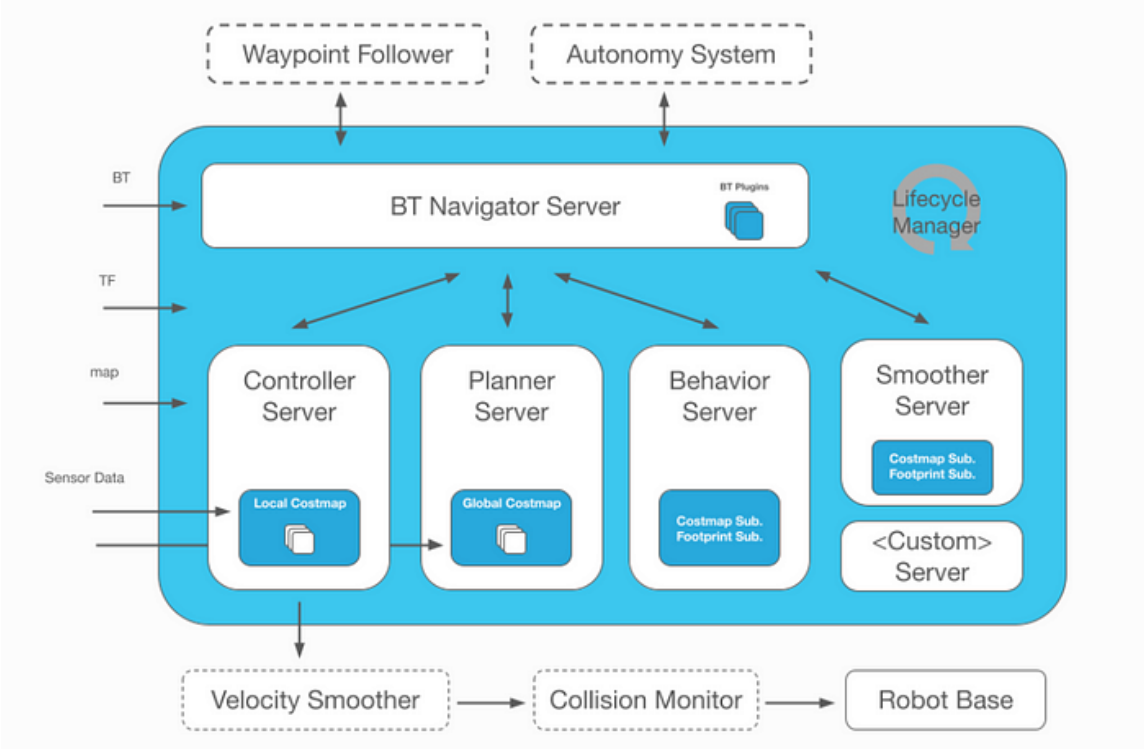


Рисунок 5.2 – Дизайн навігаційного стеку NAV 2 [30]

З самого початку новий мозок (синя коробка) потребує дизайну дерева поведінки та трансформацій/карти/даних з сенсорів, як і раніше.

Тепер давайте розглянемо основу цього дизайну.

Сервер навігатора BT (Behavior Tree) – це компонент найвищого рівня та точка входу, який містить дерево поведінки для реалізації навігаційних вчинків. Після отримання цілевого положення від користувача, він оркеструє завдання навігації з використанням свого дерева поведінки (рис. 5.3).

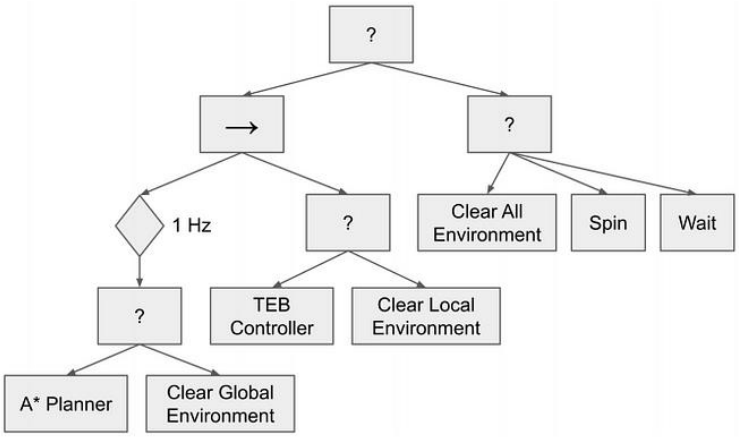


Рисунок 5.3 – Приклад дерева поведінки в BT Navigator [30]

ROS 2 action сервери використовуються для зв'язку з BT навігатором через повідомлення дії NavigateToPose для запиту навігації. З свого боку, Дерево Поведінки використовує подальші action сервери в контролері, планувальнику, сервері поведінок, згладжувачі для управління зусиллями, розрахунку планів, виконання відновлень і т.д.

Подальші сервери – вузли дерева поведінки BT навігатора взаємодіють з модульними серверами (контролер, планувальник, поведінка, згладжувач, власний) для організації та навігації робота. Кожен action сервер матиме свій унікальний тип .action для взаємодії (рис. 5.4).

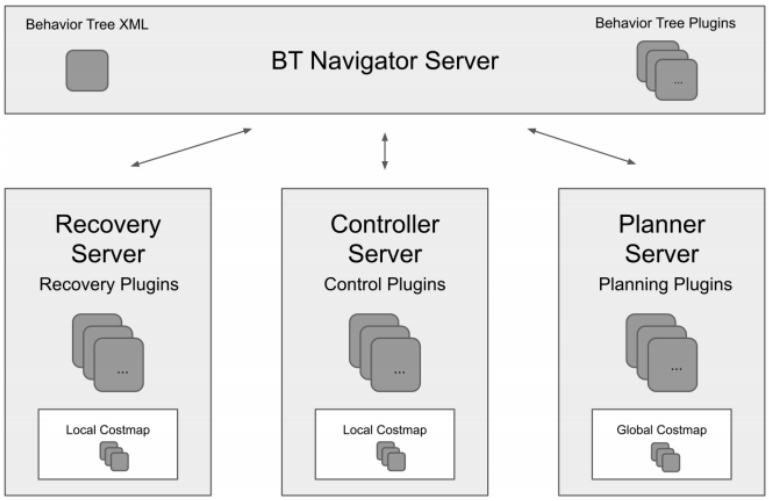


Рисунок 5.4 – Огляд Nav2 ядра [30]

На цих серверах розміщено кілька плагінів алгоритмів, що відповідають їхнім завданням, і один із них може бути обраний вузлом дерева поведінки під час виконання. Крім того, вони реалізують стандартний інтерфейс плагінів, що дозволяє створювати та вибирати нові алгоритми під час виконання.

Менеджер життєвого циклу – усі сервери, показані у проекті, реалізовані як керовані вузли (життєвий цикл) ROS 2.

Lifecycle Manager координує життєвий цикл програми BT Navigator та наступних серверних вузлів. Він проведе кожен сервер через життєвий цикл керованого вузла: неактивний, активний та завершений.

Додаткові переваги NAV 2:

– замість монолітних кінцевих автоматів – дерево поведінки, що налаштовується (з BehaviorTree.CPP);

– підтримка великої кількості типів роботів (диференціальних, голономних, акерманівських, на ногах);

– незалежні модульні сервери для основних функцій;

– виберіть плагін алгоритму для використання на серверах під час виконання;

– ROS 2 забезпечив багатоядерну обробку та продуктивність з низькою затримкою в реальному часі.

5.2 Навігація та SLAM за допомогою навігаційного стеку ROS 2

Додаємо статичну карту нашого світу, щоб наш робот міг планувати маршрут без перешкод між двома точками (рис. 5.5).

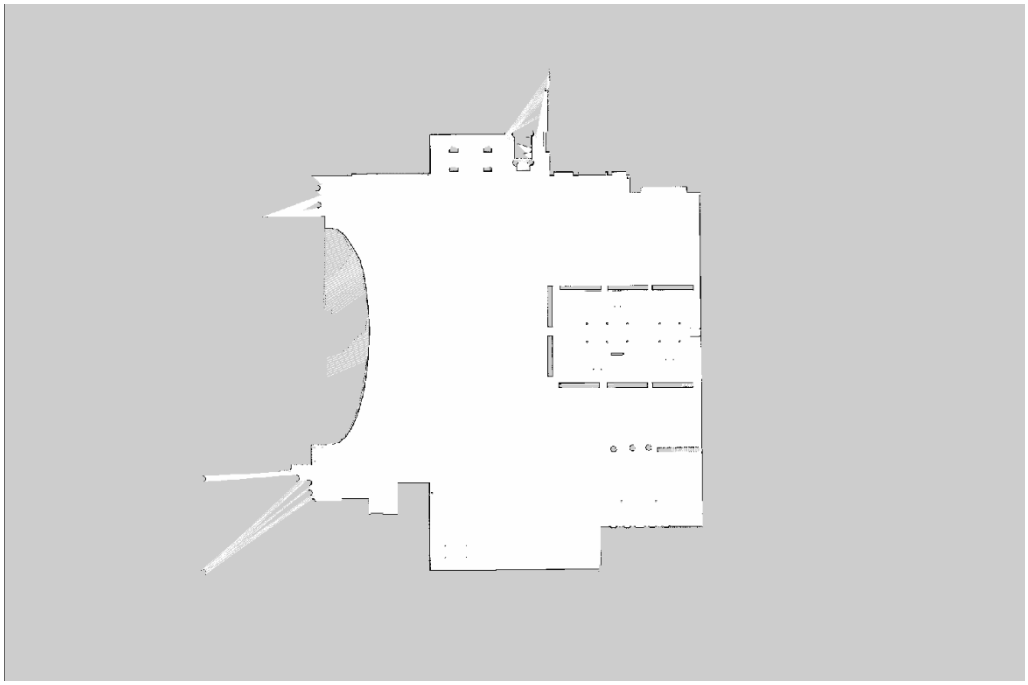


Рисунок 5.5 – Статична карта віртуального середовища

Під час симуляції буде використовуватися розроблений алгоритм гібридного фільтрування AMCL-EKF для локалізації робота в світі та для публікації координатного перетворення з кадру (map → odom).

При запуску симуляції робот починає будувати картку навколишнього середовища і одночасно здійснювати локалізацію. За допомогою пакету rqt_robot_steering, ми можемо змінювати лінійну швидкість робота, щоб запобігти явищу розриву карти при моделюванні. У процесі проведення експериментів було виявлено, що розроблюваний метод гібридного фільтрування куди швидше реагує на перешкоди, які з'являється перед ним, ніж будує карту навколишнього середовища. Це створює непотрібні зупинки та перепланування шляху, що в свою чергу може вносити зайвий шум у побудову карти навколишнього середовища.

Не зважаючи на виявлений недолік, у результаті симуляції та моделювання експерименту, було виявлено, що запропонований підхід має більш високу точність і стабільність позиціонування, розпізнавання та контролю у невідомій пересіченій місцевості порівняно з іншими методами 2D SLAM (рис. 5.6).

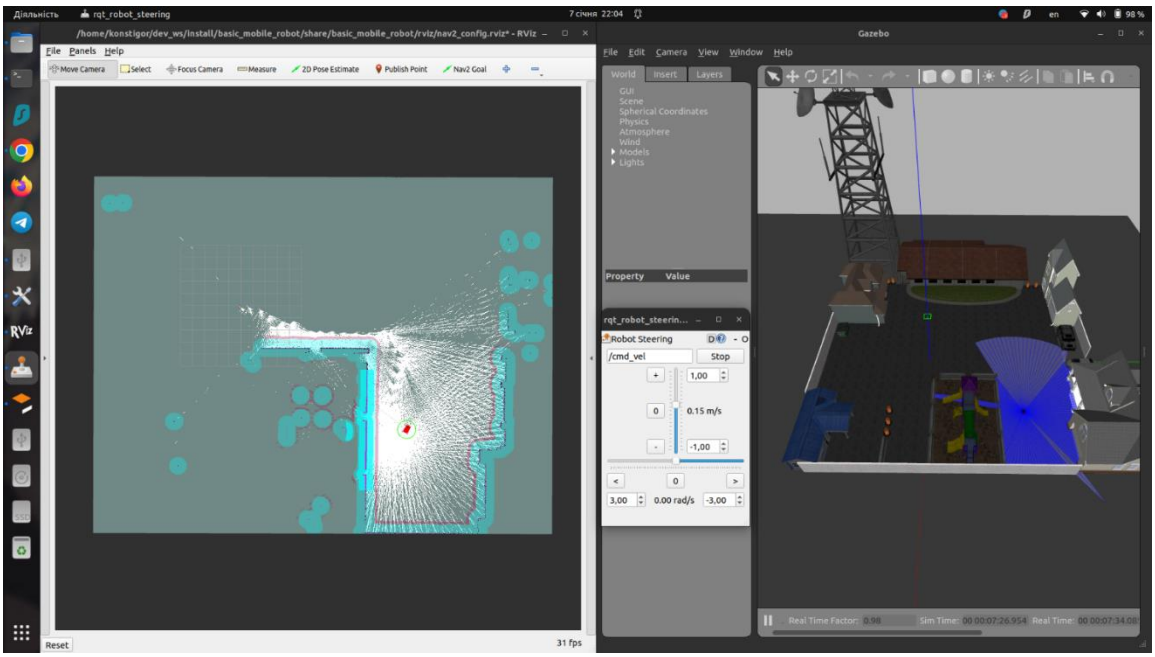


Рисунок 5.6 – Процес симуляції одночасної локалізації та картографування

Під час симуляції ми отримала карту навколишнього середовища (рис. 5.7).

Рисунок 5.7 – Отримана карта навколишнього середовища

Після отримання побудованої карти навколишнього середовища, ми отримаємо автономну карту вартості планування шляху, яку можна використовувати при автономній навігації через планування шляху (рис. 5.8). Пакет AMCL використовує фільтр частинок, щоб відстежувати позицію робота на відомій карті. Навігаційний стек зберігає інформацію про перешкоди у світі в двох картах витрат. Глобальна карта витрат використовується для довгострокового планування всього середовища, тоді як локальна карта витрат використовується для короткострокового планування та уникнення перешкод.

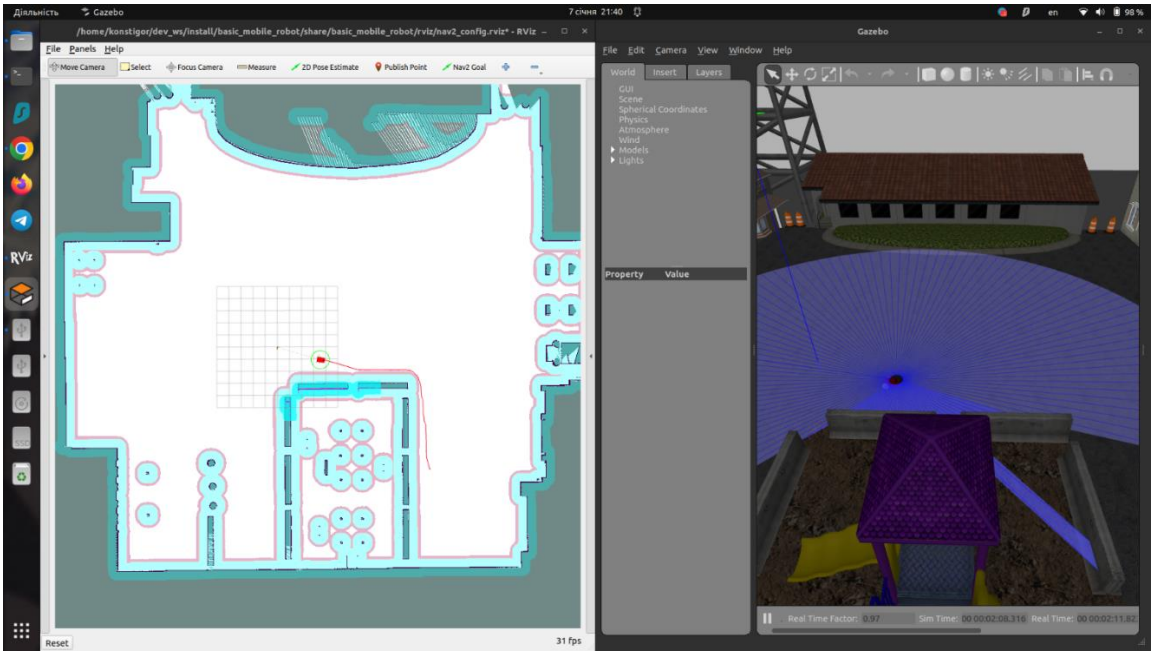


Рисунок 5.8 – Автономна навігація по карті вартості планування шляху

Скористаємося інструментами ROS, (rqt_graph) для перегляду залежностей роботи пакетів (рис. 5.9) та tf2_tools view_frames для перегляду дерева перетворень (рис. 5.10).

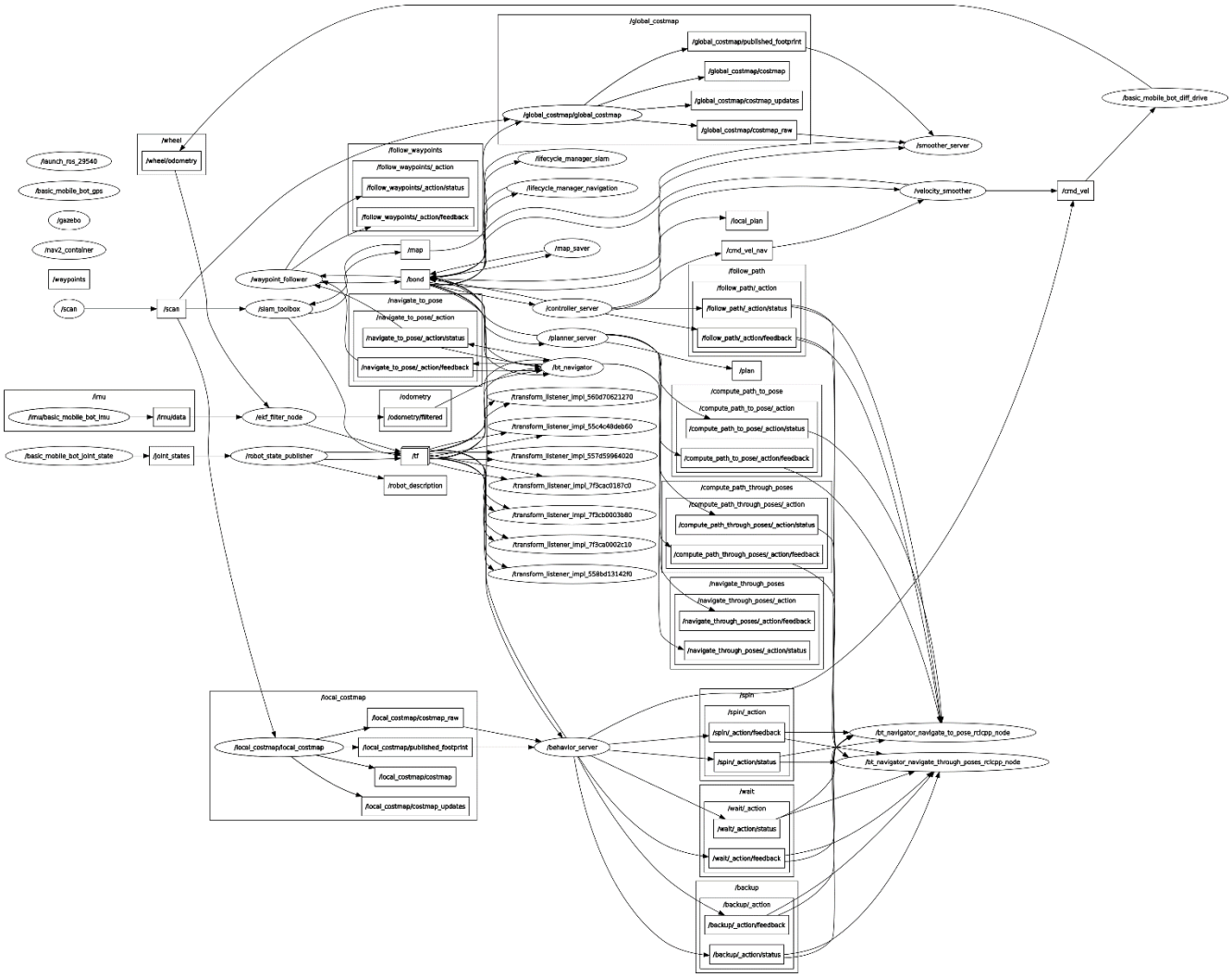


Рисунок 5.9 – Граф залежностей пакетів та їх вузлів при автономній локалізації та картографування

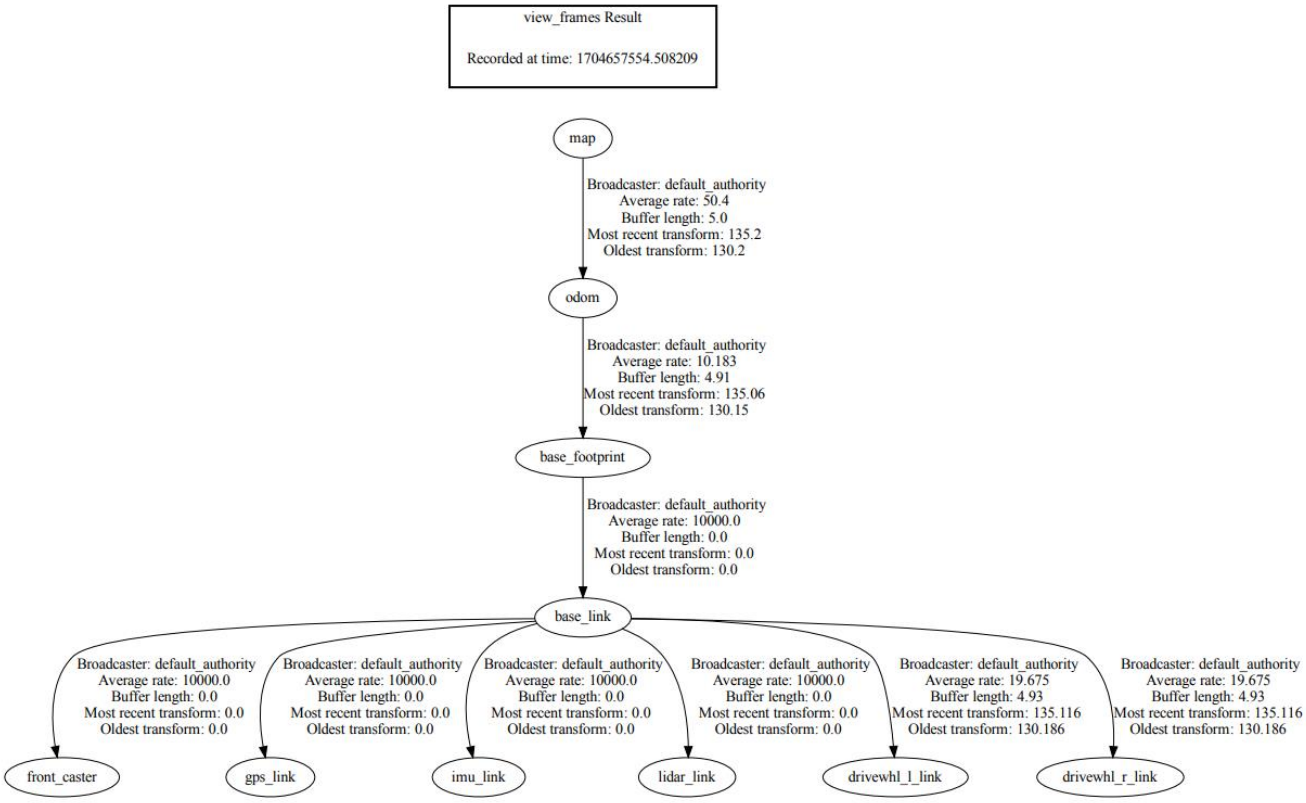


Рисунок 5.10 – Дерево перетворень автономній локалізації та картографування гібридного фільтрування AMCL-EKF

5.3 Висновки по п’ятому розділу

В результаті виконання п’ятого розділу магістерської кваліфікаційної роботи було визначено різницю між поколіннями навігації NAV 1 та NAV 2 та обраний більш підходящій до розроблюваної роботи. Була проведена симуляція навігації та SLAM за допомогою навігаційного стеку ROS 2. Був перевірений розроблюваний метод гібридного фільтрування AMCL-EKF.

6 ОХОРОНА ПРАЦІ

Охорона праці для програміста є важливою складовою забезпечення безпечних та комфортних умов праці. Нижче подано кілька рекомендацій щодо охорони праці для програміста:

а) Ергономіка робочого місця;

1) забезпечте правильне розташування монітора так, щоб програміст міг працювати, не напружуючи очі та шию;

2) використовуйте ергономічні стільці та столи, щоб уникнути проблем із спиною та шийною частинами тіла.

б) Перерви та вправи;

1) заохочуйте регулярні короткі перерви для відпочинку очей та руху;

2) пропонуйте працівникам виконувати комплекс вправ для розтягування, щоб покращити кровообіг та запобігти м'язовим напругам.

в) Організація робочого часу;

1) забезпечте можливість гнучкого графіку роботи для уникнення перевтоми та стресу;

2) обмежуйте тривалість безперервної роботи за комп'ютером та стежте за часом перерв.

г) Безпека інформації;

1) забезпечте безпечний доступ до інформації та данних, використовуючи ефективні засоби аутентифікації та шифрування;

2) проводьте навчання працівників з питань кібербезпеки та застосування найкращих практик у сфері безпеки програмного забезпечення.

д) Захист від впливу шкідливих факторів;

1) забезпечте відповідний захист від шкідливих впливів, таких як випромінювання, переохолодження чи перегрівання робочого приміщення;

2) встановлюйте спеціальні екрани або фільтри для захисту від випромінювання моніторів.

е) Безпека програмного забезпечення;

1) забезпечте регулярне оновлення програмного забезпечення для запобігання вразливостям;

2) введіть ефективні політики паролів та захисту даних для запобігання несанкціонованому доступу.

ж) Психологічне здоров'я.

1) створіть сприятливу атмосферу в колективі та підтримуйте відкритий комунікаційний процес;

2) забезпечте доступ до психологічної підтримки та навчання стратегіям копінгу зі стресом.

Загальний підхід до охорони праці для програміста повинен бути комплексним та спрямованим на забезпечення фізичного та психологічного благополуччя працівників.

6.1 Розрахунок освітленості і рівня шуму

Нормованим параметром природного освітлення являється коефіцієнт природного освітлення (КПО). КПО встановлюється в залежності від розряду виконуваних зорових робіт.

Робота оператора ПК відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5 – 1,0 мм), для яких при використанні бокового освітлення КПО=1,5%.

Для штучного освітлення нормованим параметром виступає $E_{мін}$ – мінімальний рівень освітленості, та $К_п$ – коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%.

Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300...500 лк.

Розрахунок штучного освітлення було проведемо для кімнати площею 20 $м^2$, ширина якої складає 5 м, довжина – 4 м, висота – 3 м.

Скористаємося методом використання світлового потоку. Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = \frac{E \cdot K \cdot S \cdot Z}{\eta}, \quad (6.1)$$

де F – світловий потік, що розраховується, лм;

E – нормована мінімальна освітленість, лк; $E = 300$ лк;

S – площа освітлюваного приміщення (у нашому випадку $S = 20$ м^2);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1,1...1,2, в нашому випадку $Z = 1{,}1$);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку $K = 1{,}5$);

η – коефіцієнт використання світлового потоку, (виражається відношенням світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп, і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{\text{ст.}}$) і стелі ($\rho_{\text{стелі}}$)), значення коефіцієнтів дорівнюють $\rho_{\text{ст.}}$ = 40% і $\rho_{\text{стелі}}$= 60%. Обчислимо індекс приміщення за формулою:

$$I = \frac{S}{h(A+B)}, \quad (6.2)$$

де S – площа приміщення, S = 20 м^2;

h – розрахункова висота підвісу, h = 2,9 м;

A – ширина приміщення, A = 4 м;

B – довжина приміщення, B = 5 м.

Підставивши значення отримаємо:

$$I = \frac{20}{2{,}9(4+5)} = 0{,}77.$$

Знаючи індекс приміщення I, знаходимо η = 0,22.

Підставимо всі значення у формулу для визначення світлового потоку F :

$$F = \frac{300 \cdot 1{,}5 \cdot 20 \cdot 1{,}1}{0{,}22} = 45000 \text{ лм}.$$

Для освітлення використані люмінесцентні лампи типу ЛБ 40-1, світловий потік яких F = 4320 лм. Розрахуємо необхідну кількість ламп у світильниках за формулою:

$$N = \frac{F}{F_л}, \qquad (6.3)$$

де N – кількість ламп, що визначається;

F – світловий потік, F = 45000 лм;

$F_л$ – світловий потік лампи, $F_л$ = 4320 лм.

$$N = \frac{45000}{4320} = 11.$$

В приміщенні використовуються світильники типу ОД. Кожен світильник комплектується двома лампами. Тобто необхідно використовувати 6 світильників із 12 працюючими лампами в них.

Розрахунок рівня шуму. Рівень шуму вимірюється в децибелах (дБ) і визначається кількістю звуку в середовищі. Одним з несприятливих факторів виробничого середовища є високий рівень шуму, створюваний друкованими

пристроями, обладнанням для кондиціонування повітря, вентиляторами систем охолодження в самих ЕОМ(електронно-обчислювальна машина). Для вирішення питань про необхідність і доцільність зниження шуму необхідно знати рівні шуму на робочому місці оператора. Рівень шуму, що виникає від декількох некогерентних джерел, що працюють одночасно, підраховується на підставі принципу енергетичного підсумовування випромінювань окремих джерел:

$$L_{\text{сер}} = 10 \lg \sum_{i=1}^{n} 10^{0,1 * L_i}, \tag{6.4}$$

де L_i – рівень звукового тиску і-го джерела шуму;

n – кількість джерел шуму.

Отримані результати розрахунку порівнюється з допустимим значенням рівня шуму для даного робочого місця. Якщо результати розрахунку вище допустимого значення рівня шуму, то необхідні спеціальні заходи щодо зниження шуму. До них відносяться: облицювання стін і стелі залу звукопоглинальними матеріалами, зниження шуму в джерелі, правильне планування обладнання і раціональна організація робочого місця оператора. Рівні звукового тиску джерел шуму, що діють на оператора на його робочому місці представлені в табл. 6.1.

Таблиця 6.1 – Рівні звукового тиску різних джерел

Джерело шуму	Рівень шуму, дБ
Клавіатура	10
Монітор	17
Вентилятор	45
Жорсткий диск	40
Сканер	42
Принтер	45

Зазвичай робоче місце оператора оснащене наступним обладнанням: вінчестер в системному блоці, вентилятори системи охолодження ПК, монітор, клавіатура, принтер і сканер.

Підставивши значення рівня звукового тиску для кожного виду устаткування у формулу, отримаємо:

$$L_{\Sigma} = 10\lg\left(10^{4} + 10^{4.5} + 10^{1.7} + 10^{1} + 10^{4.5} + 10^{4.2}\right) = \ 49{,}5 \text{ дБ}.$$

Отримане значення не перевищує допустимий рівень шуму для робочого місця оператора, рівний 65 дБ. І якщо врахувати, що навряд чи такі периферійні пристрої як сканер і принтер будуть використовуватися одночасно, то ця цифра може бути ще нижчою. Крім того при роботі принтера безпосередню присутність оператора необов'язково, тому що принтер обладнаний механізмом автоподачі аркушів.

6.2 Висновки по шостому розділу

В результаті виконання шостого розділу магістерської кваліфікаційної роботи проведено аналіз робочого місця інженера-програміста. Створені умови повинні забезпечувати комфортну роботу. Були зазначені оптимальні розміри робочого столу і крісла, робочої поверхні, а також проведено вибір системи і розрахунок оптимального освітлення виробничого приміщення, а також розрахунок рівня шуму на робочому місці. Дотримання умов, що визначають оптимальну організацію робочого місця інженера-програміста, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить як в кількісному, так і в якісному відносинах продуктивність праці програміста, що в свою чергу сприятиме якнайшвидшій розробці і налагодженню програмного продукту.

ВИСНОВКИ

Гібридна оцінка позиції, яка описана у роботі, об'єднує дві відокремлені парадигми – Калманівське та частинкове фільтрування – у новий локальний метод оцінки одометрії. Підхід ґрунтується на використанні даних від LiDAR та IMU-сенсорів, а також на локальному локалізаційному методі Монте-Карло, який використовується разом із технікою лазерної одометрії для локальної оцінки положення за допомогою розширеного фільтру Калмана. Ця дія дозволила об'єднати стійкість до ковзання та занесення, яку забезпечує AMCL, точне відстеження динамічних змін орієнтації у трьох вимірах, що є характерним для IMU, та незалежність від типу приводу методу лазерної відомості, завдяки багатомірній оцінці вимірювань у EKF для ефективного відстеження позиції. Глобальна локалізація вирішується в представленій системі алгоритмом 2D SLAM із розподіленою системою Рао-Блеквелла на основі даних про рух, які надходять від оцінок EKF. Ефективність, точність та надійність розробленого інтегрованого підходу на основі SLAM для керування мобільними роботами в ускладненому терені були підтверджені численними симуляціями. Симуляції виконувалися в середовищі Rviz та Gazebo, алгоритми керування роботом та реалізації використовувались в рамках ROS 2. Представлені результати показують, що розроблений метод інкрементальної локалізації AMCL-EKF дає кращу продуктивність в ускладненому, неструктурованому терені, тоді як в структурованому терені його помилки оцінки є порівнянні з методом RF2O. Розроблена техніка базується на нестандартному використанні частинкового фільтру MCL як джерела оцінки локалізації EKF в локальній системі координат замість його звичайної картографічної глобальної локалізації. На практиці це позитивно впливає на подальшу локалізацію на основі SLAM та побудову мапи. Усі досягнуті результати підтверджують обґрунтованість використання розробленого підходу локалізації для різних завдань, включаючи інспекцію в ускладнених та ізольованих теренах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. I. Nevliudov, S. Novoselov, O. Sychova and D. Mospan, "Multithreaded Software Control of Industrial Manipulator Movement," 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), Kremenchuk, Ukraine, 2022, pp. 1-6, doi: 10.1109/MEES58014.2022.10005675.

2. Сухачов К. Сучасні методи одночасної локалізації і картографування в режимі реального часу. Автоматизація та комп'ютерноінтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку : матеріали Всеукр. науково-практ. Internet-конф., м. Черкаси, 2023. С. 77–79.

3. Nevlyudov I., Novoselov S., Sukhachov K. Method of simultaneous localization and mapping for construction of 2.5d maps of the environment using ros. Innovative technologies and scientific solutions for industries. 2023. No. 2 (24). P. 145–160. URL: https://doi.org/10.30837/itssi.2023.24.145.

4. Kudriashov A., Buratowski T., Giergiel M. Hybrid AMCL-EKF filtering for SLAM-based pose estimation in rough terrain. Advances in mechanism and machine science. Cham, 2019. P. 2819–2828. URL: https://doi.org/10.1007/978-3-030-20131-9_279.

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

6. I. Nevliudov, S. Novoselov, O. Sychova and S. Tesliuk, "Development of the Architecture of the Base Platform Agricultural Robot for Determining the Trajectory Using the Method of Visual Odometry," 2021 IEEE XVIIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Polyana (Zakarpattya), Ukraine, 2021, pp. 64-68, doi: 10.1109/MEMSTECH53091.2021.9468008.

7. Investigation of widely used SLAM sensors using analytical hierarchy process / M. S. A. Khan et al. Journal of sensors. 2022. Vol. 2022. P. 1–15. URL: https://doi.org/10.1155/2022/5428097.

8. A simultaneous localization and mapping (SLAM) framework for 2.5D map building based on low-cost lidar and vision fusion / G. Jiang et al. Applied sciences. 2019. Vol. 9, no. 10. P. 2105. URL: https://doi.org/10.3390/app9102105.

9. Multi-Robot 2.5D localization and mapping using a monte carlo algorithm on a multi-level surface / V. A. Rosas-Cervantes et al. Sensors. 2021. Vol. 21, no. 13. P. 4588. URL: https://doi.org/10.3390/s21134588.

10. Debeunne C., Vivet D. A review of visual-lidar fusion based simultaneous localization and mapping. Sensors. 2020. Vol. 20, no. 7. P. 2068. URL: https://doi.org/10.3390/s20072068.

11. Visual SLAM: why bundle adjust? / A. P. Bustos et al. 2019 international conference on robotics and automation (ICRA), Montreal, QC, Canada, 20–24 May 2019. URL: https://doi.org/10.1109/icra.2019.8793749.

12. Deep learning for visual navigation of unmanned ground vehicles: a review / N. O'Mahony et al. 2018 29th irish signals and systems conference (ISSC), Belfast, 21–22 June 2018. URL: https://doi.org/10.1109/issc.2018.8585381.

13. Real-Time 3D mapping in isolated industrial terrain with use of mobile robotic vehicle / T. Buratowski et al. Electronics. 2022. Vol. 11, no. 13. P. 2086. URL: https://doi.org/10.3390/electronics11132086.

14. Improved lidar localization method for mobile robots based on multi-sensing / Y. Liu et al. Remote sensing. 2022. Vol. 14, no. 23. P. 6133. URL: https://doi.org/10.3390/rs14236133.

15. Freitas C. M. d. S. d. Autonomous navigation with simultaneous localization and mapping in/outdoor: master's thesis. 2020. URL: https://hdl.handle.net/10216/128968.

16. Colmap-slam: a framework for visual odometry / L. Morelli et al. The international archives of the photogrammetry, remote sensing and spatial information sciences. 2023. XLVIII-1/W1-2023. P. 317–324. URL: https://doi.org/10.5194/isprs-archives-xlviii-1-w1-2023-317-2023.

17. A robust and modular multi-sensor fusion approach applied to MAV navigation / S. Lynen et al. 2013 IEEE/RSJ international conference on intelligent robots

and systems (IROS 2013), Tokyo, 3–7 November 2013. URL: https://doi.org/10.1109/iros.2013.6696917.

18. Graph structure-based simultaneous localization and mapping using a hybrid method of 2D laser scan and monocular camera image in environments with laser scan ambiguity / T. Oh et al. Sensors. 2015. Vol. 15, no. 7. P. 15830–15852. URL: https://doi.org/10.3390/s150715830.

19. A multi-sensorial simultaneous localization and mapping (SLAM) system for low-cost micro aerial vehicles in gps-denied environments / E. López et al. Sensors. 2017. Vol. 17, no. 4. P. 802. URL: https://doi.org/10.3390/s17040802.

20. Nam T., Shim J., Cho Y. A 2.5D map-based mobile robot localization via cooperation of aerial and ground robots. Sensors. 2017. Vol. 17, no. 12. P. 2730. URL: https://doi.org/10.3390/s17122730.

21. Scale estimation and correction of the monocular simultaneous localization and mapping (SLAM) based on fusion of 1D laser range finder and vision data / Z. Zhang et al. Sensors. 2018. Vol. 18, no. 6. P. 1948. URL: https://doi.org/10.3390/s18061948.

22. Shin Y.-S., Park Y. S., Kim A. Direct visual SLAM using sparse depth for camera-lidar system. 2018 IEEE international conference on robotics and automation (ICRA), Brisbane, QLD, 21–25 May 2018. URL: https://doi.org/10.1109/icra.2018.8461102.

23. Xu Y., Ou Y., Xu T. SLAM of robot based on the fusion of vision and LIDAR. 2018 IEEE international conference on cyborg and bionic systems (CBS), Shenzhen, 25–27 October 2018. URL: https://doi.org/10.1109/cbs.2018.8612212.

24. FFT-Based Scan-Matching for SLAM Applications with Low-Cost Laser Range Finders / G. Jiang et al. Applied sciences. 2018. Vol. 9, no. 1. P. 41. URL: https://doi.org/10.3390/app9010041.

25. Moore T., Stouch D. A generalized extended kalman filter implementation for the robot operating system. Intelligent autonomous systems 13. Cham, 2015. P. 335–348. URL: https://doi.org/10.1007/978-3-319-08338-4_25.

26. Grisetti G., Stachniss C., Burgard W. Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE transactions on robotics. 2007. Vol. 23, no. 1. P. 34–46. URL: https://doi.org/10.1109/tro.2006.889486.

27. Joseph L. Learning Robotics using Python: Design, simulate, program, and prototype an autonomous mobile robot using ROS, OpenCV, PCL, and Python, 2nd Edition. Packt Publishing, 2018. 280 p.

28. Joseph L., Cacace J. Mastering ROS for robotics programming - second edition: design, build, and simulate complex robots using the robot operating system. Packt Publishing, 2018. 580 p.

29. I. Nevludov, O. Sychova, O. Reznichenko, S. Novoselov, D. Mospan and V. Mospan, "Control System for Agricultural Robot Based on ROS," 2021 IEEE International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk, Ukraine, 2021, pp. 1-6, doi: 10.1109/MEES52427.2021.9598560.

30. Joseph L., Cacace J. Mastering ROS for robotics programming - third edition: best practices and troubleshooting solutions when working with ROS. Packt Publishing, Limited, 2021.

31. Jaimez M., Monroy J. G., Gonzalez-Jimenez J. Planar odometry from a radial laser scanner. A range flow-based approach. 2016 IEEE international conference on robotics and automation (ICRA), Stockholm, 16–21 May 2016. 2016. URL: https://doi.org/10.1109/icra.2016.7487647.

32. Ojeda Cabrera J. HERNÁNDEZ, maximiliano; DEL ESTAL, héctor (eds.): "conceptos en disputa, disputas sobre conceptos", dykinson, madrid, 2022, 242p. https://hdl.handle.net/10016/36007. Agora. papeles de filosofía. 2023. Vol. 42, no. 2. URL: https://doi.org/10.15304/ag.42.2.8996.