Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

(тема)
Розроблення підсистеми прийняття рішень для орієнтації робота у динамічному просторі

Виконав: студент 2 курсу, гр. КІТПВм-22-1
Канаєв Владислав Дмитрович
(прізвище, ініціали)

Спеціальність
151 Автоматики і комп'ютеризованих технологій
освітньої програми Комп'ютерно-інтегровані технологічні процеси і виробництва
(код і повна назва напряму)

Тип програми освітньо-професійна
(повна назва освітньої програми)

Керівник доц. Максимова С. С.
(посада, прізвище, ініціали)

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

2024 р.

Я Канаєв Владислав Дмитрович, 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.

« 18 » Січня 2024р. _____ Канаєв В. Д.

Харківський національний університет радіоелектроніки

| | |
|---|---|
| Факультет | Автоматики і комп'ютеризованих технологій |
| Кафедра | Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки |
| Рівень вищої освіти | другий (магістерський) |
| Спеціальність | Автоматизація та компю'терно-інтегровані технології |
| Тип програми | освітньо-професійна |
| Освітня програма | 151 Комп'ютерно-інтегровані технологічні процеси і виробництва |

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_03_» листопада 2024 р.

# ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Канаєву Владиславу Дмистровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення підсистеми прийняття рішень для орієнтації робота у динамічному просторі
затверджена наказом по університету від 03.11. 2023 р. № 1287 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 16. 01. 2024 р.
3. Вихідні дані до роботи
3.1 Мова програмування для реалізації програми – Python
3.2 Середа розробки – Visual Studio Code

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз технічного завдання
4.2 Вступ
4.3 Поняття мобільного роботу
4.4 Програмна реалізація орієнтації робота у динамічному просторі
4.5 Розробка програмного коду
4.6 Перевірка роботи програми

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Демонстраційний матеріал представлений у форматі презентації PowerPoint.

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

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

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз літератури за темою роботи | 10.10.23 – 17.10.23 | Виконав |
| 2 | Поняття мобільного роботу | 18.10.23 – 30.10.23 | Виконав |
| 3 | Програмна реалізація орієнтації робота | 01.11.23 – 16.11.23 | Виконав |
| 4 | Розробка коду програми | 17.11.23 – 19.12.23 | Виконав |
| 5 | Інструкція з інсталяції програми | 20.12.23 – 29.12.23 | Виконав |
| 6 | Перевірка праці симуляції програми | 04.01.24 – 11.01.24 | Виконав |
| 7 | Оформлення пояснювальної записки | | |
| 8 | Подання роботи на рецензію | | |
| 9 | Подання роботи на підпис зав. кафедри | | |

Дата видачі завдання 03.11.2023 р.

Студент _____ Канаєв В. Д.
(підпис) ( прізвище, ініціали)

Керівник роботи _____ доц. Максимова С. С.
(підпис) (посада, прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 58 с., 21 рис., 3 дод., 12 джерел.

АВТОМАТИЗАЦІЯ, ПРОГРАМНА СИМУЛЯЦІЯ, МОДЕЛЮВАННЯ, МОБІЛЬНИЙ РОБОТ, ДИНАМІЧНИЦ ПРОСТІР, СИСТЕМА НАВІГАЦІЇ.

Мета роботи – підвищення ефективності виробництва за рахунок розробки підсистеми прийняття рішень для орієнтації мобільного робота.

Об’єкт розробки – процес орієнтації мобільного робота.

Предмет розробки – підсистема прийняття рішень для орієнтації мобільного робота

Для досягнення мети було розглянуто та проаналізувані сучасні мобільні роботи, які здатні орієнтуватися у динамічному просторі. Більш детально розглянули метод SLAM, що використовується в мобільних автономних засобах для побудови карти в невідомому просторі або для оновлення карти в заздалегідь відомому просторі з одночасним контролем поточного розташування та пройденого шляху.

Під час написання кваліфікаційної роботи було промодельовано рух мобільного робота у динамічному просторі. Розроблено код для симуляції прийняття рішень для орієнтації мобільного робота, який під час свого руху здатен генерувати карту, на фоні оточуючих його перешкод. Описано етапи установки програми та проведено її тестування.

# ABSTRACT

Explanatory note: 58 p., 21 pic., 3 applications, 12 sources.

AUTOMATION, SOFTWARE SIMULATION, MODELING, MOBILE ROBOT, SPACE DYNAMIC, NAVIGATION SYSTEM.

The purpose of the work is to increase production efficiency through the development of a decision-making subsystem for the orientation of a mobile robot.

The object of development is the orientation process of a mobile robot.

The subject of development is a decision-making subsystem for the orientation of a mobile robot

To achieve the goals, we considered and analyzed modern mobile robots that are able to navigate in a dynamic space. The SLAM method, which is used in mobile autonomous vehicles to build a map in an unknown space or to update a map in a previously known space with simultaneous control of the current location and the traveled path, was elaborated in more detail.

During the writing of the qualification work, the movement of a mobile robot in a dynamic space was modeled. A code has been developed for simulating decision-making for the orientation of a mobile robot, which is capable of generating a map against the background of surrounding obstacles during its movement. The stages of installing the program are described and its testing is carried out.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

НДР – науково-дослідна робота;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ТЗ – технічне завдання;

SLAM – Simultaneous Localization and Mapping.

# ВСТУП

В даний час мобільні роботи стають невід'ємною частиною автоматизованого виробництва. Вони надають унікальні можливості для підвищення продуктивності, зниження витрат і покращення безпеки на робочому місці. Однак, щоб роботи успішно функціонували в промислових умовах, їм потрібна ефективна система орієнтації, яка дозволяє їм безпечно й точно пересуватися, виконувати завдання та взаємодіяти з навколишнім середовищем, що дозволяє їм легко доставляти, завантажувати та розвантажувати матеріали практично без втручання людини.

Виробничі маніпулятори, роботи-безпілотники, інтерактивні іграшки, роботи-хірурги, побутові андроїди, роботи-собаки, дрони, роботи-укладачі цегли та інше. Мобільні роботи стали надійними помічниками в багатьох сферах. Вони здатні виконувати монотонні та небезпечні завдання, звільняючи людину від рутини та ризику. Однак, щоб мобільні роботи працювали в різноманітних і динамічних виробничих середовищах, їм потрібна надійна система орієнтації, яка забезпечує високу точність і надійність руху.

Мета роботи – підвищення ефективності виробництва за рахунок розробки підсистеми прийняття рішень для орієнтації мобільного робота.

Об’єкт розробки – процес орієнтації мобільного робота.

Предмет розробки – підсистема прийняття рішень для орієнтації мобільного робота.

Робота виконана згідно [1, 2].

# 1 АНАЛІЗ ЛІТЕРАТУРИ

## 1.1 Аналіз літератури за темою роботи

### 1.1.1 Поняття мобільного роботу

Мобільний робот – це робот, здатний пересуватись. Мобільна робототехніка зазвичай вважається підполем робототехніки та інформатики.

Мобільні роботи можуть переміщатися у своєму середовищі та не прив'язані до одного фізичного розташування. Мобільні роботи можуть бути "автономними", що означає, що вони здатні здійснювати навігацію по неконтрольованій середі без необхідності використання фізичних або електромеханічних пристроїв керування. В якості альтернативи мобільні роботи можуть покладатися на пристрої керування, які дозволяють їм переміщатися за заздалегідь визначеним маршрутом навігації щодо контрольованого простору. Навпаки, промислові роботи зазвичай більш менш нерухомі, що складаються зі зчленованого важеля і захоплюючого вузла, прикріпленого до нерухомої поверхні.

Мобільні роботи стали найпоширенішими у комерційних та промислових умовах. Лікарні використовують автономні мобільні роботи для переміщення матеріалів протягом багатьох років. У складах встановлені мобільні роботизовані системи для ефективного переміщення матеріалів із полиці для зберігання до зон виконання замовлень. Мобільні роботи також є основним напрямом поточних досліджень, і майже кожен великий університет має одну або кілька лабораторій, орієнтованих на дослідження мобільних роботів. Мобільні роботи також знаходяться у промислових, військових та охоронних системах. Внутрішні роботи – це споживчі товари, у тому числі розважальні роботи та ті, що виконують певні домашні завдання, такі як пилосос чи садівництво.

Компоненти мобільного робота - це контролер, програмне забезпечення для керування, датчики та виконавчі механізми. Контролер зазвичай є

мікропроцесор, вбудований мікроконтролер або персональний комп'ютер. Програмне забезпечення для мобільного управління може бути мовою рівня складання, або мовами високого рівня, такими як C, C++, Pascal, Fortran або спеціальним програмним забезпеченням реального часу. Датчики, що використовуються, залежать від вимог робота. Вимогами можуть бути нечіткі розрахунки, тактильне та безконтактне зондування, ранжування тріангуляції, запобігання зіткненням, місцезнаходження та інші конкретні програми [3].

### 1.1.2 Аналіз мобільних роботів, здатних орієнтуватися у динамічному просторі.

На підприємствах для автоматизації процесів та підвищення продуктивності використовуються різні види мобільних роботів, які здатні орієнтуватися в просторі. Деякі з них використовуються у виробництві, складах, логістиці та інших галузях. Ось кілька прикладів таких роботів:

AGV - це рухомі платформи, які використовуються для автоматичного переміщення матеріалів або товарів у виробничих об'єктах або складах. Вони можуть використовувати різні системи орієнтації, включаючи магнітні смуги, лазери або візуальні сенсори, щоб навігувати.

AMR - це роботи, які використовують різноманітні сенсори та системи штучного інтелекту для навігації у просторі. Вони можуть використовувати візуальне сприймання, лазерні датчики, радіолокацію та інші технології для визначення свого положення та управління маршрутом.

Дрони часто використовуються на підприємствах для віддаленого моніторингу, інвентаризації або вантажних перевезень. Вони можуть бути обладнані камерами, GPS та іншими сенсорами для орієнтації в просторі.

Роботи з автоматичним переміщенням робочого обладнання: На виробництві можуть використовуватися роботи, які переміщають робоче обладнання, таке як робочі столи або інструменти, до різних робочих місць. Ці роботи зазвичай обладнані системами лазерної навігації та обробки даних для визначення місця розташування.

Мобільні роботи для обслуговування і очищення: Мобільні роботи використовуються для очищення підлоги, прибирання місць для відпочинку або обслуговування інших об'єктів на підприємствах.

Ці мобільні роботи використовують різні технології для навігації, включаючи лазери, візуальні сенсори, GPS і інші методи. Вони можуть бути програмовані для виконання різних завдань, що спрощує та автоматизує багато процесів на підприємствах.

### 1.1.3 Сучасні мобільні роботи

Сучасний МР, який здатний виконувати перелічені функції – це складний комплекс, де поєднані новітні досягнення багатьох галузей науки і техніки. Проектування та виробництво мобільних роботів пов’язане із застосуванням високих і критичних технологій таких науково-технічних напрямків, як компактні та енергоємні джерела й перетворювачі енергії; високонадійні транспортні засоби високої прохідності; високоточні супутникові та інерційні навігаційні системи [4].

Порівняльний аналіз структурних схем існуючих МР різного призначення показує, що всі вони є складними комплексами, які укладаються з трьох основних частин: рухомий виконавчий агрегат (власне робот), пульт дистанційного керування роботом і агрегат живлення комплексу. Базова модель може бути оснащена будь-якими із наведених нижче спеціалізованих функціональних систем або пристроїв:

– супутникова навігаційна система;

– інерціальна навігаційна система;

– радіаційно-дозиметрична система;

– система гідродинамічного руйнування;

– система рентгеноскопічного контролю;

– система аналізу газових середовищ;

– система кріогенного охолодження;

– система виявлення мін та інші [5].

## 1.2 Метод орієнтиру SLAM для мобільних роботів

SLAM від англ. розшифровується як Одночасна Локалізація та Картографування (Simultaneous Localization and Mapping). Даний метод навігації використовується для визначення розташування та орієнтації автономних роботів на заздалегідь невідомій їм місцевості, а також для оновлення або доповнення вже відомих карт навколишнього простору.

У цілому нині, принцип роботи SLAM відбувається так. Роботу необхідно в кожний момент часу знати своє місцезнаходження, а також поступово сканувати навколишній простір за допомогою сенсорів, складаючи, таким чином, карту місцевості. Карта будується поступово, принаймні дослідження роботом нових областей. Основним джерелом інформації про місцезнаходження робота є одометрія, отримана тим чи іншим чином (колеса, комп'ютерний зір, IMU або їх комбінація). Однак, у міру вибудовування карти, робот починає звірятися з карткою. Наприклад, якщо робот проїжджає по тій області приміщення, яку він вже відсканував, відбувається звіряння за певними патернами. В результаті, якщо апарат розуміє, що поточні показання одометрії не відповідають показанням карти, відбувається коригування одометрії.

З математичної точки зору, SLAM намагається оцінити карту та весь шлях, пройдений роботом. Таким чином, поза робота розраховується тільки в кінці траєкторії, виконаної роботом [6].

При використанні SLAM відображення картки може бути як у 2D, так і в 3D. Відображення геометрії місцевості в 3D є складнішим і, зазвичай, вимагає більше пам'яті. Однак, це не завжди дозволяє забезпечити якісне розуміння навколишнього середовища, тому двовимірного відображення достатньо більшості завдань. Крім цього, методи SLAM можна розділити за підходами SLAM, що використовуються, для представлення карт. Наприклад, SLAM, засновані на сітці точок (grid-based), на основі розпізнавання унікальних об'єктів

(feature-based) та на основі графіків (graph-based) для більш розріджених зображень, топологічні (topological) для розпізнавання типів та семантичні (semantic) ) для вищого рівня розуміння довкілля [6].

### 1.2.1 Feature-based SLAM

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

### 1.2.2 Graph-based SLAM

SLAM на основі графів або мереж також намагається створити карту за допомогою графа, вузли якого відповідають позиціям робота в різні моменти часу, а ребра є просторовими обмеженнями, що зв'язують пози робота разом. Обмеження полягають у розподілі ймовірностей щодо перетворення між позами. Використовуючи граф, побудований з урахуванням вимірювань датчиків, система визначає найімовірнішу конфігурацію поз з урахуванням ребер графа. Одним із найбільш популярних підходів до SLAM на основі графів є метод відображення в реальному часі (VSLAM Rtabmap), що ґрунтується на інкрементному детекторі замикання циклу на основі візуальних образів [6].

### 1.2.3 Grid-based SLAM

Теоретично найпростішим методом є підхід на основі сітки. За такого підходу середовище розбивається на сітку точок певного розміру. Кожна точка може бути перешкодою, не зайнята або не досліджена. Наприклад, осередок зі значенням 1 буде вважатися зайнятим, а інший зі значенням 0 буде повністю вільним [6]. Також значення точки може змінюватись від 0 до 1 за допомогою проміжних значень.

### 1.2.4 Топологічний SLAM

Топологічне представлення проблеми SLAM спрямоване створення графо-подобного описи довкілля, а чи не точної метричної карти. У топологічному описі вузли можуть відповідати значним місцям, які легко розрізнити. Основною ідеєю цього підходу було те, що люди та тварини не створюють точних карт навколишнього середовища, в якому вони знаходяться. Як правило, ці методи підходять для навігації в простих середовищах і застосовувати їх у складніших і більших середовищах важко [6].

### 1.2.5 SemanticSLAM

Подання карток також може бути виконане у вигляді моделей семантичних карток. На відміну від топологічних підходів, де відображення фільтрує метричну інформацію та використовує лише розпізнавання місць для розрізнення розташування, семантичне зіставлення пов'язує семантичні концепції з об'єктами навколишнього середовища. Семантичне відображення все ще знаходиться на ранніх стадіях розробки і, залежно від рівня необхідних семантичних можливостей, може бути дуже складним для реалізації. Незважаючи на це, згідно з Carlos Miguel [6] було проведено безліч досліджень з семантичного відображення та семантичного SLAM [7, 8].

### 1.2.6 Laser SLAM

Метод побудови карти з лідара дозволяє, як правило, побудувати наочнішу схему приміщення, на якій будуть видно всі стіни, об'ємні меблі, перегородки та інші об'єкти. На такій карті простіше орієнтуватися, якщо, наприклад, потрібно побудувати маршрут для робота, або зрозуміти, де він знаходиться. Однак, у карт, побудованих подібним чином, є суттєвий недолік - лідар сканує тільки одну 2D площину, розташовану на якому-небудь зафіксованому рівні (мається на увазі стандартний 2D лідар, тому що існують лідари, що сканують 3D простір, проте їх вартість в рази більше, ніж вартість не тільки 2D лідарів, але також RGBD та стерео-камер).

Під час процесу картографування потенційні перешкоди знаходяться на різних рівнях вертикальної осі. У випадку, коли мобільний робот має низький профіль (робот пилосос), це не так сильно впливає на ефективність картографування, оскільки більшість перешкод, що знаходяться в межах висоти робота все ж таки будуть розпізнані лідером. Більше того, в деяких випадках це буде перевагою – наприклад, робот низького профілю може проїхати під столом, стільцем та іншими предметами меблів, у яких обсяг займаного простору змінюється залежно від висоти. Через те, що лідер сканує безпосередньо певну площину простору, він зможе зафіксувати лише ті частини перешкод, що знаходяться на одній з ним висоті. В результаті, низькопрофільний робот звертатиме увагу лише на ті об'єкти (частини об'єктів) на його шляху, з якими потенційно може зіткнутися.

Якщо ж робот має великі габарити, особливо у висоту, даний спосіб картографування не зможе визначити всі об'єкти, що становлять небезпеку. Цю проблему можна вирішити кількома способами. По-перше, можна використати декілька лідерів на різних рівнях висоти. Однак, неможливо заздалегідь припустити, на яких саме висотах можуть бути перешкоди. По-друге, можна будувати карту за допомогою одного лідара, а на додаток використовувати камеру глибини, яка фіксуватиме перешкоди, але не наносити їх на карту (Rtabmap не дозволяє будувати карту з використанням даних і з лідара і з камер глибини одночасно, якщо тільки не конвертувати все у формат даних лідара). Дані про перешкоди будуть використовуватися лише для локальної тимчасової картки в стеку навігації ROS. Таким чином, робот буде створювати карту, що легко читається і, в той же час, буде здатний визначити всі перешкоди, які йому необхідно уникати в процесі пересування.

Незважаючи на те, що на карті, побудованій за лідером, знаходяться не всі перешкоди, створення такої карти має і позитивні сторони для навігації. Rtabmap може використовувати лідер не тільки для визначення перешкод, але також зіставляти карту та дані лідера в даний момент за допомогою функції ICP (Iterative Closest Point). Якщо положення робота за даними лідара не відповідає

карті, алгоритм коригує поточні координати робота. Через те, що картка за даними лідара має набагато менше шумів, її використання разом із функцією ICP дасть якісніший результат, ніж використання ICP на карті, побудованої камерою глибини.

Даний функціонал корисний у випадку, якщо одометрія накопичила помилку, або коли робот рухається по певній ділянці в протилежному напрямку, в якому ця ділянка була спочатку досліджена. Оскільки глобальна навігація Rtabmap відбувається за візуальними образами, то під час дослідження робот бачить зображення як правило перед собою. Тоді як під час руху у протилежному напрямку, він зможе знаходити візуальні образи, відповідні тим, що він зберіг. Звичайно, це можна нівелювати шляхом дослідження тих самих ділянок приміщення двічі – рухаючись по них у протилежних напрямках, проте це збільшує час, що витрачається на картографування [9].

## 1.3 Аналіз програми CoppeliaSim

CoppeliaSim розроблено на основі універсальної архітектури. Там не є основною чи центральною функціональністю у CoppeliaSim. Швидше CoppeliaSim має різні відносно незалежні функції, тобто можна ввімкнути або вимкнути за потреби, також на основі моделі. Уявіть собі сценарій моделювання, де промисловий робот має забрати ящики та перенести їх в інше місце; CoppeliaSim обчислює динаміку захоплення й утримання коробок і виконує кінематичне моделювання для інших частин циклу, коли динамічні ефекти незначні. Це підхід дає можливість розрахувати промислового робота рух швидко і точно, чого б не було якби це було повністю змодельовано з використанням складної динаміки бібліотеки. Цей тип гібридного моделювання є виправданим у цьому ситуації, якщо робот жорсткий і фіксований, а не інакше під впливом середовища. На додаток до адаптивного включення різних своїх функціональні можливості вибірково, CoppeliaSim також може використовувати їх у симбіотичній манері, співпрацюючи з одним інший. У випадку гуманоїдного

робота, наприклад, CoppeliaSim може обробляти рухи ніг, (a) спочатку обчислюючи зворотна кінематика для кожної ноги (тобто від бажаної стопи положення та орієнтація, розраховуються всі положення суглобів ніг); а потім (b) призначення обчислених позицій з'єднання використовуються як цільові позиції суглобів модулем динаміки. Це дозволяє дуже багатогранно визначити рух гуманоїда таким чином, оскільки кожну ногу потрібно було б просто призначити дотримуйтеся 6-вимірного шляху: решта обчислень є автоматично піклується. Функціональність пов'язана з конкретними об'єктами сцени або конкретні розрахункові модулі, обидва описані далі.

### 1.3.1 Об'єкти сцени

Сцена моделювання CoppeliaSim або модель моделювання містить кілька об'єктів сцени або елементарних об'єктів, які збираються у деревоподібній ієрархії. Наступні об'єкти сцени підтримується у CoppeliaSim:

– з'єднання: з'єднання – це елементи, які з'єднують два або більше об'єкти сцени разом з одним-трьома ступенями свободи (призматичні, революційні, гвинтоподібні або сферичний). Вони можуть працювати в різних режимах (напр. режим сили/крутного моменту, режим зворотної кінематики тощо);

– фігури: фігури є трикутними сітками, які використовуються для жорстких моделювання та візуалізація тіла. Вони можуть бути оптимізовано для швидкої динамічної реакції на зіткнення обчислення, як угруповання первісних або опукл форми. Інші об'єкти сцени або модулі обчислень сильно покладаються на форми для своїх розрахунків (датчики наближення, модуль динаміки або модуль розрахунку відстані між сітками для приклад);

– датчики наближення: вони виконують точну обчислення мінімальної відстані до частини фігури міститься в конфігурованому обсязі виявлення, як на відміну від простого виявлення на основі промені. Це призводить до більш безперервної роботи та таким чином забезпечує більш реалістичне моделювання;

– датчики зору: датчики зору дозволяють витягувати інформація про складне зображення зі сцени моделювання (кольори, розміри об'єктів, карти

глибини тощо). Вбудований функція фільтрації та обробки зображень дозволяє склад блоків фільтруючих елементів. Бачення датчики використовують апаратне прискорення для отримання необроблених зображень (OpenGL);

– датчики сили: вони представляють жорсткі зв'язки між форми, які можуть реєструвати прикладені сили та крутні моменти, і що може умовно розпатися, коли дано поріг перевищений;

– графіки: графіки можуть записувати велику різноманітність попередньо визначені або спеціальні потоки даних. Потоки даних можуть потім буде відображено безпосередньо (часовий графік заданих даних тип), або комбіновані один з одним для відображення X/Y графіки або тривимірні криві;

– камери: вони дозволяють візуалізувати сцену, коли пов’язані з вікном перегляду;

– світло: світло освітлює сцену або окрему сцену об’єктів і безпосередньо впливати на камери або зір датчики;

– шляхи: вони дозволяють складні визначення руху пробіл (послідовність вільно комбінованих перекладів, обертання та/або паузи), і використовуються для керування а пальник зварювального робота по заданій траєкторії, або для забезпечення руху конвеєрної стрічки приклад;

– манекени: манекен – це система відліку, яка може бути використовується для різних завдань, і в основному використовується в у поєднанні з іншими об’єктами сцени, і як такі, можна розглядати як «помічника»;

– фрези: вони представляють настроювані опуклі об’єми які можна використовувати для імітації різання поверхні операції з формами (наприклад, фрезерування, лазерне різання, тощо).

### 1.3.2. Модулі розрахунку

Об’єкти сцени рідко використовуються самі по собі, швидше вони працювати на інших об’єктах сцени (або в поєднанні з ними) (наприклад, датчик наближення розпізнає форми). Крім того, CoppeliaSim пропонує кілька

розрахункових модулів, які можуть працювати безпосередньо на одному або кількох об’єктах сцени. Нижче наведені основні розрахункові модулі:

– модуль Kinematics: дозволяє виконувати кінематичні розрахунки (прямий/зворотний) для будь-якого типу механізму (розгалужені, закриті, надлишкові, що містять вкладені петлі тощо). Модуль базується на розрахунку демпфований псевдоінвер за методом найменших квадратів. Це підтримує умовні, демпфовані/недемпфовані та зважена роздільна здатність;

– модуль динаміки: дозволяє працювати з твердим тілом розрахунок динаміки та взаємодії (зіткнення відповідь, захоплення тощо) через Bullet Physics Бібліотека та Open Dynamics Engine. Симуляції на основі динаміки все ще знаходяться в зародку взуття й часто базується на наближеннях важливо не покладатися лише на одну фізику двигун, щоб перевірити результати. На момент написання, третє, підтримка високоякісної фізики через Готується Vortex Dynamics;

– модуль виявлення зіткнень: забезпечує швидке втручання перевірка між будь-якою фігурою або колекцією фігур. Цей модуль повністю незалежний від колізії алгоритм розрахунку відгуку динаміки модуль. Він використовує структури даних на основі бінарного дерева орієнтованих обмежувальних прямокутників для прискорень. Додаткова оптимізація досягається за допомогою temporal техніка когерентного кешування;

– модуль розрахунку відстані між сітками: дозволяє швидке обчислення мінімальної відстані між будь-якими форми (опукла, увігнута, відкрита, закрита тощо) або колекція форм. Модуль використовує ті самі дані структур як модуль виявлення зіткнень. Додаткова оптимізація також досягається за допомогою а техніка кешування тимчасової когерентності;

– модуль планування шляху/руху: обробляє голономічні задачі планування шляху та неголономного шляху завдання планування (для транспортних засобів, схожих на автомобілі) через підхід походить від випадкового дерева швидкого дослідження (RRT) [10]. Завдання на планування шляху о також підтримуються кінематичні ланцюги.

Для універсальності наведені вище модулі реалізовано в а загальним способом, не роблячи жодних припущень щодо базові сцени моделювання або моделі. Мета інтегрувати їх у CoppeliaSim замість того, щоб покладатися на них зовнішні бібліотеки дещо схожі на призначення з вбудованими сценаріями, як описано в розділі II, B: широкий більшість симуляцій або імітаційних моделей не вимагають будь-який спеціальний або висококласний інструмент. Натомість вони потребують хорошого набору основних інструментів. Якщо вони інтегровані в симулятор, і їхні визначення завдань, безпосередньо приєднані до імітаційних моделей, тоді моделі стають надзвичайно портативними: розповсюдження а створено імітаційну модель для іншої машини чи платформи через один файл моделі; немає необхідності розповсюджувати, перекомпілювати, інсталювати або перезавантажувати плагін. Подібним чином це також робить моделі дуже масштабованими: дубльовані моделі автоматично працює без необхідності будь-яких змін вихідний код. Процес дублювання може відбуватися навіть під час моделювання.

Традиційний підхід розширення функціональності за допомогою а плагіна, щоб підтримувати конкретну імітаційну модель курс також підтримується у CoppeliaSim [11].

## 1.4 Висновки до розділу

У цьому розділі було розглянуто таке поняття, як мобільний робот та розібрали для чого і де вони використовуються.

Під час аналізу можна зробити висновок, що розробка підсистеми управління для мобільних роботів важлива для забезпечення їх здатності орієнтуватися і ефективно виконувати завдання в приміщеннях виробничих просторів. Це включає в себе розробку алгоритмів навігації, систем датчиків, методів взаємодії з оточуючим середовищем та технічних засобів управління. Мобільні роботи можуть бути використані для автоматизації логістики та

складського управління, що сприяє оптимізації процесів зберігання та переміщення товарів.

Також було розглянуто метод SLAM, що використовується в мобільних автономних засобах для побудови картки у невідомому просторі, який буде використаний у симуляції орієнтировки мобільного робота.

# 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ОРІЄНТАЦІЇ РОБОТА У ДИНАМІЧНОМУ ПРОСТОРІ

## 2.1. Перетворення полярних координат на декартові:

Для визначення позиції мобільного робота нам потрібно перетворити полярні координати на декартові. Полярні координати представлені відстанню та кутом. Результуючі декартові координати (x, y) обчислюються таким чином:

$$x = r * \cos(\theta) + x_r \tag{2.1}$$

$$y = -r * \cos(\theta) + y_r \tag{2.2}$$

де r – відстань від робота,

θ – кут у полярних координатах,

$(x_r, y_r)$ – поточна позиція робота в декартових координатах.

## 2.2. Додавання невизначеності з використанням багатовимірного нормального розподілу

Для додавання випадкової невизначеності до вимірювань відстані та кута необхідно створити багатовимірне нормальне розподілення з середнього значення і матриці коваріації:

$$f(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k * |\Sigma|}} * e^{-\frac{1}{2}(x-\mu)^T * \Sigma^{-1} * (x-\mu)} \tag{2.3}$$

де x – вектор спостережень,

k – розмірність простору,

Σ – матриця коваріації,

μ – середнє значення у математичній статистиці

2.3. Обчислення відстані між двома точками в декартових координатах:

Для розрахунку відстані між двома точками $(x_1, y_2)$ і $(x_1, y_2)$ використовується евклідова відстань:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.4)$$

Це рівняння описує довжину прямої лінії між двома точками декартової площині, використовуючи теорему Піфагора.

2.4. Моделювання лазерного сенсора

Першим кроком є обчислення координат кінцевої точки променя декартових координатах. Це відбувається з використанням кута та дальності:

$$x_2 = x_1 + d * \cos\beta \quad (2.5)$$

$$y_2 = y_1 - d * \sin\beta \quad (2.6)$$

де $(x_1, y_1)$ – поточна координата по осі;

d – відстань або довжина (наприклад, вимірювана лазерним детектором);

$\beta$ – кут, який визначає напрямок вектора зміщення;

$(x_2, y_2)$ – нова координата по осі x після зміщення на відстань під кутом $\beta$ відносно позиції $(x_1, y_1)$.

Далі промінь дискретизується, щоб перевірити, чи є перешкода вздовж його шляху. Це робиться за допомогою параметра u, який змінюється від 0 до 1:

$$x = x_2 * u + x_1 * (1 - u) \quad (2.7)$$

$$y = y_2 * u + y_1 * (1 - u) \quad (2.8)$$

де (x,y) – нова координата по осі x після усунення;

$(x_1, y_1)$ – початкова координата по осі;

$(x_2, y_2)$ – кінцева координата по осі.

## 2.4 Перевірка кольору для визначення перешкод

Потім перевіряється колір пікселя на карті в цій точці. Якщо колір відповідає чорному (перешкоді), вимірюється відстань від сенсора до цієї точки з урахуванням невизначеності після чого данні зберігаються для подальшої орієнтації.

## 2.5 Висновки до розділу

Математична модель включає перетворення координат, додавання невизначеності з використанням багатовимірного нормального розподілу і моделювання роботи лазерного сенсора.

Всі ці кроки об'єднуються для створення даних про відстані та кути, які сенсор міг би виміряти в реальному світі, враховуючи випадкові фактори та перешкоди на карті.

# 3 ПРОЦЕС СТВОРЕННЯ ПРОГРАМИ-СИМУЛЯЦІЇ

## 3.1 Опис та модулювання мобільного робота

Для проведення симуляції моделі мобільного робота було використано програму під назвою «CoppeliaSim», що раніше була відома як «V-REP». Це потужне інструментарій надає можливості створення різноманітних сценаріїв та віртуальних середовищ, де можна тестувати та розробляти робототехнічні застосунки.

CoppeliaSim пропонує широкий вибір готових 3D-моделей різних типів роботів, а також інструменти для створення власних моделей. Це дозволяє користувачам моделювати та тестувати поведінку роботів в різних умовах, роблячи програму ідеальним інструментом для робототехнічного моделювання.

Для цілей демонстрації було обрано одну з доступних моделей робота. Щоб взаємодіяти з нею, необхідно розмістити обраного робота в середу моделювання CoppeliaSim (рисунок 3.1). Такий підхід дозволяє нам наблизитися до функціональності обраної моделі та вивчити її характеристики в зручному для нас віртуальному середовищі.

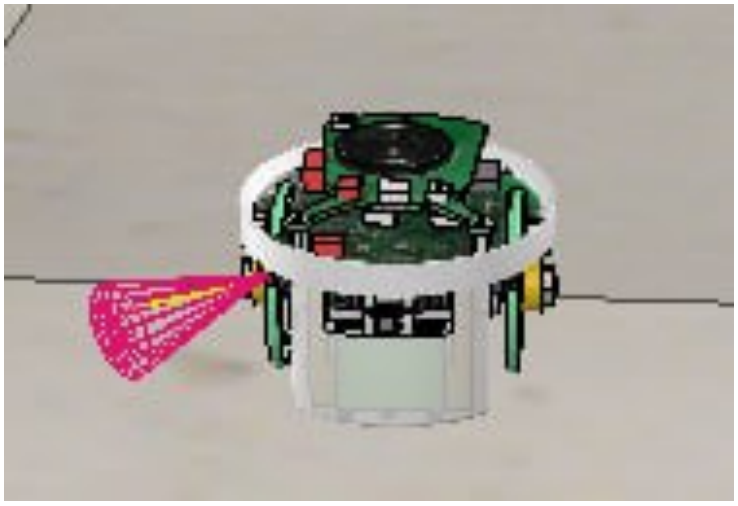

Рисунок 3.1 – Обрана модель робота

Щоб в CoppeliaSim вказати траєкторію для руху робота, було використовуємо  шлях (Path). Нам потрібно також перемістити його на середу моделювання і додайте точки шляху вказуючи їх координати. Після чого нам потрібно обрати об'єкт шляху і налаштувати його параметри. У властивостях об'єкта можна змінювати різні параметри, такі як швидкість, розташування, напрямок для більш гнучкого керування рухом робота. Побудований шлях можна побачити на рисунку 3.2.

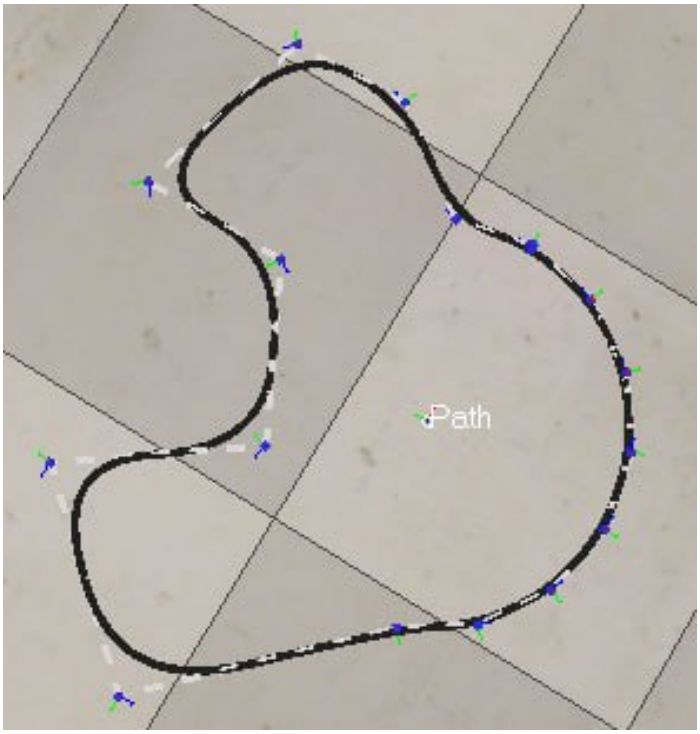


Рисунок 3.2 – Шлях яким рухатиметься мобільний робот

Наступним кроком небхідно додати перешкоди на шляху у нашого робота, щоб зробити сценарій більш складним та реалістичним. Додавання перешкод може включати в себе встановлення об'єктів, які перебувають на шляху робота, та визначення впливу цих перешкод на рух. Кінцевий варіант розташування об’єктів можна побачити на рисунку 3.3.

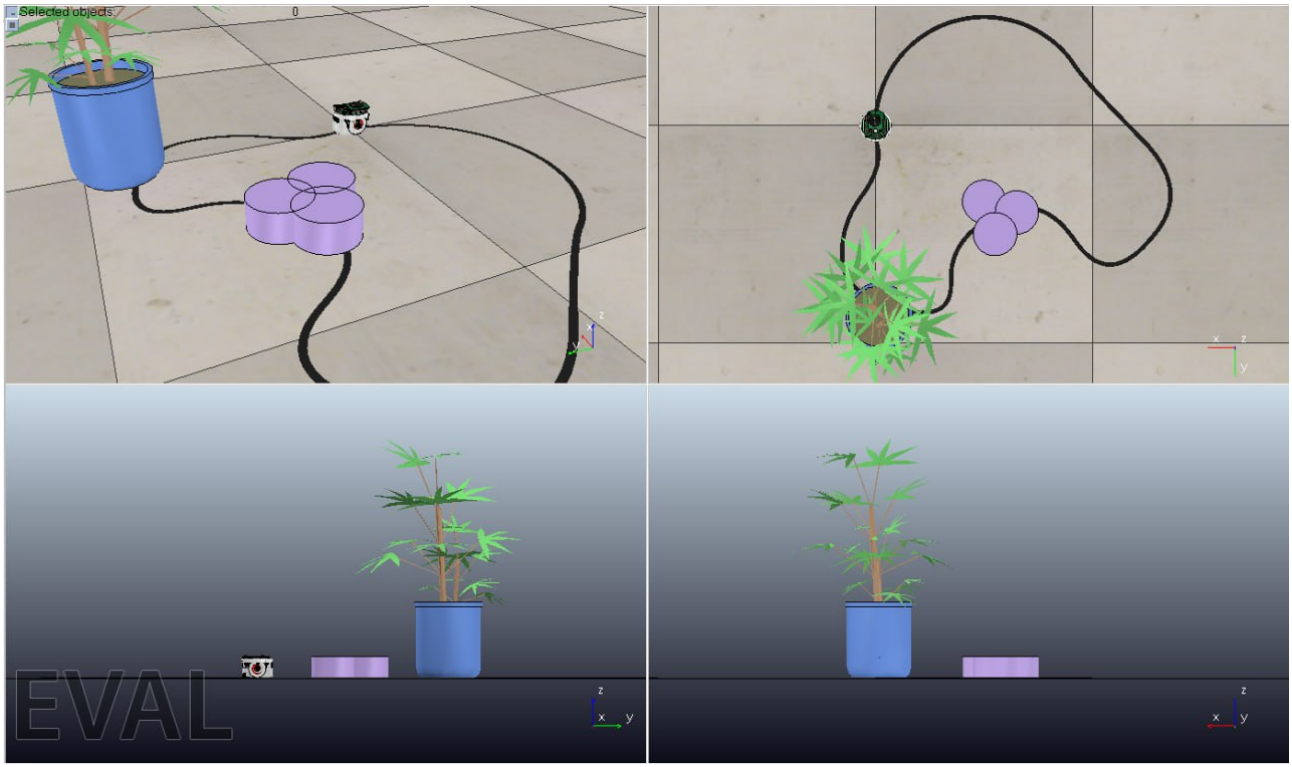


Рисунок 3.3 – Розташування всіх об’єктів у віртуальному середовищі

CoppeliaSim володіє широким функціоналом, що включає в себе можливість моделювання різноманітних сенсорів, таких як камери, датчики відстані, гіроскопи та інші, а також актуаторів, таких як двигуни та сервоприводи. Це важливий аспект, оскільки він надає користувачам унікальну можливість проводити більш деталізовані та реалістичні симуляції робототехнічних систем.

Можливість моделювати сенсори дозволяє віртуальним роботам отримувати дані з навколишнього середовища, що є ключовим для розробки та тестування алгоритмів сприйняття та взаємодії з ним. Також, наявність можливості моделювати актуатори робить симуляції більш динамічними, оскільки роботи можуть взаємодіяти з оточенням, реагуючи на отримані сенсорні дані та приймаючи відповідні рішення. Далі було додано до мобільного робота сенсор, щоб під час свого руху він мав можливість реагувати на перешкоди, які зустрічаються на його шляху (рисунок 3.4).

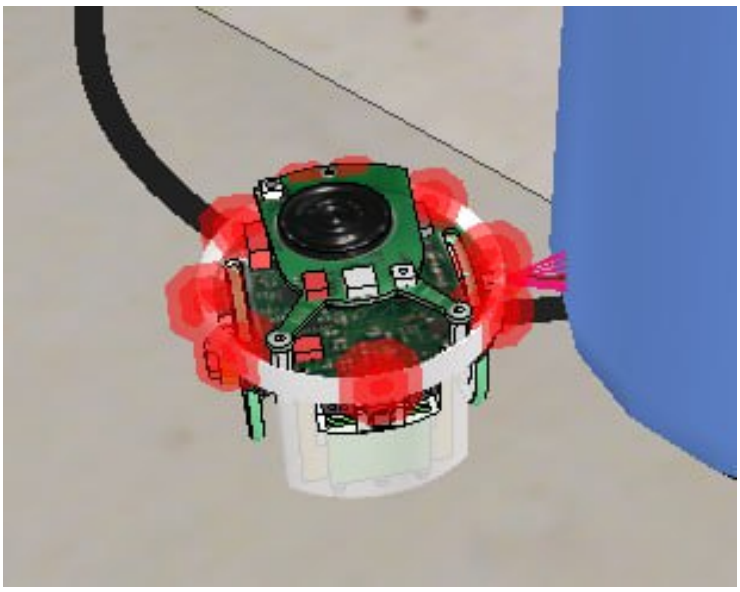

Рисунок 3.4 – Розташування сенсорів у мобільного робота

Також програма включає в себе вбудовану скриптову мову програмування, яка базується на Lua, щоб дозволити користувачам писати сценарії та керувати роботами під час симуляції. Це важлива функціональність, яка відкриває широкі можливості для розробників та дослідників.

Інтеграція скриптової мови Lua дозволяє створювати не лише прості, але й дуже складні та гнучкі програми для управління роботами в середовищі симуляції. Це означає, що користувачі можуть розробляти різноманітні сценарії, включаючи автономне керування роботами, взаємодію з оточуючим середовищем, вирішення завдань та багато іншого.

Використання скриптової мови Lua робить CoppeliaSim потужним інструментом для досліджень у галузі робототехніки, дозволяючи вдосконалювати та тестувати різні алгоритми та стратегії у віртуальному середовищі перед їхнім реальним застосуванням.

Останнім кроком нам потрібно написати скріпт, щоб керувати роботом під час симуляції. Перш за все потрібно створити основний цикл програми, буде управляти маршрутом мобільного робота використовуючи алгоритм Брезенхейма.

Після цього необхідно обробити дані від сенсорів робота та використовуємо їх в основному циклі для прийняття рішень. Якщо на шляху

робота зустрічається перешкода, то він буде намагатися оминути її, після чого знову повернеться на попередній шлях

## 3.2 Опис основних елементів програми

У якості мови програмування був обраний Python з використанням трьох бібліотек, які знадобляться для створення програми.

Бібліотеки NumPy і Math потрібні для математичних обчислень завдяки якою буде відтворюватись поведінка лідара і проводити обчислення для формування карти, яку буде створювати мобільний робот для орієнтування у просторі.

Бібліотека Pygame у більшості випадків використовується для створення ігор, але в нашому випадку вона буде симулювати рух мобільного робота.

Програма поділена на три основних файла. У першому env.py необхідно створити візуалізацію середовища для планування шляху руху робота. Після імпорту бібліотек було створено класс buildEnviroment, який буде створювати і відображати середовище (рисунок 3.5).

```
class buildEnviroment:
  def __init__(self,MapDimension):
    pygame.init()
    self.pointCloud=[]
    self.externalMap=pygame.image.load('SLAM/map.png')
    self.maph, self.mapw = MapDimension
    self.MapWindowName = 'RRT path planning'
    pygame.display.set_caption(self.MapWindowName)
    self.map = pygame.display.set_mode((self.mapw, self.maph))
    self.map.blit(self.externalMap,(0,0))
```

Рисунок 3.5 – Ініціалізація об’єкта класу buildEnviroment

У конструкторі класу ініціалізуються різні змінні та властивості об'єкта, такі як розміри вікна (maph, mapw), назва вікна (MapWindowName), створення

графічного, завантаження зовнішньої мапи, для симуляції перешкод. Для карти можна завантажити будь-яке зображення. В якості прикладу було обране зображення планування квартири (рисунок 3.6).

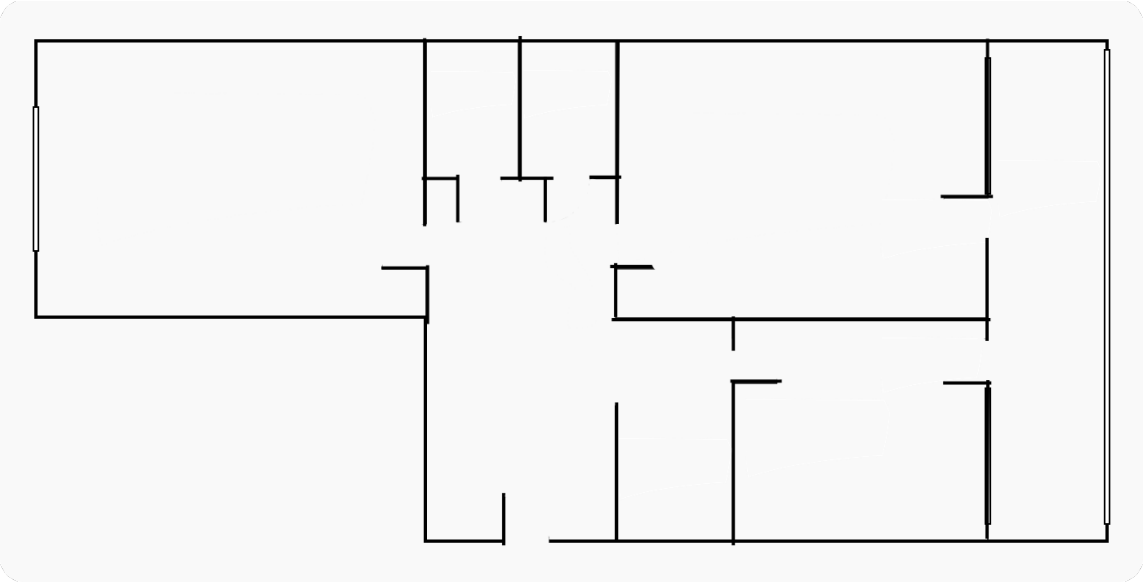

Рисунок 3.6 – Карта, яка використовується для демонстрації

Далі створимо метод (рисунок 3.7), який буде приймати відстань, кут, та координати, задля подальшого перетворювання відстані та кута (у полярних координатах) в координати в декартовій системі координат, враховуючи позицію робота за (2.1) та (2.2).

```
def AD2pos(self,distance,angle,robotPosition):
    x = distance * math.cos(angle)+robotPosition[0]
    y = -distance * math.sin(angle)+robotPosition[1]
    return (int(x),int(y))
```

Рисунок 3.7 – Метод перетворення полярних координатах в координати в декартовій системі

Щоб зберігати дані з сенсорів робота було створено метод dataStorage (рисунок 3.8), який викликає метод AD2pos, для перетворення даних та збереження їх у списку pointCloud, який ініціалізується на початку програми.

```
def dataStorage(self,data):
  print(len(self.pointCloud))
  for element in data:
    point=self.AD2pos(element[0],element[1],element[2])
    if point not in self.pointCloud:
      self.pointCloud.append(point)
```

Рисунок 3.8 – Метод збереження даних

Далі потрібно показати створену карту, для цього створемо метод show_sensorData (рисунок 3.9), який у якості зелених крапок відображає зібрані сенсорами дані і відтворимо їх.

```
def show_sensorData(self):
  self.infomap=self.map.copy()
  for point in self.pointCloud:
    self.infomap.set_at((int(point[0]),int(point[1])),(0,255,0))
```

Рисунок 3.9 – Метод збереження даних

Після створення та візуалізації середовища наступним кроком створимо симуляцію лазерного сенсора на основі даних про оточення. Назвемо другий файл sensor.py та після імпорту необхідних бібліотек заздалегідь створимо функцію uncertainty_add (рисунок 3.10), яка буде додавати невизначеність (шум) до відстані та кута, використовуючи багатовимірний нормальний розподіл з вказаною коваріацією (2.3).

```
def uncertainty_add( distance, angle,sigma):
  mean = np.array([distance, angle])
  covariance = np.diag(sigma ** 2)
  distance, angle = np.random.multivariate_normal(mean, covariance)
  distance=max(distance, 0)
  angle=max(angle,0)
  return [distance, angle]
```

Рисунок 3.10 – Функція додавання невизначеність до відстані та кута

Наступним кроком додамо клас LaserSensor (рисунок 3.11), який ініціалізує параметри лазерного сенсора, такі як дальність, швидкість, ступінь невизначеності, поточна позиція сенсора та розміри екрану.

```
class LaserSensor:
  def __init__(self,Range,map,uncertainty):
    self.Range=Range
    self.speed = 4
    self.sigma = np.array([uncertainty[0], uncertainty[1]])
    self.position=(0,0)
    self.map=map
    self.W,self.H = pygame.display.get_surface().get_size()
    self.sensedObstacles=[]
```

Рисунок 3.11 – Ініціалізація параметрів лазерного сенсору

Створимо метод distance (рисунок 3.12) та за (2.4) обчислимо евклідову відстань між позицією сенсора та перешкодою.

```
  def distance(self,obstaclesPosition):
    px=(obstaclesPosition[0]-self.position[0])**2
    py=(obstaclesPosition[1]-self.position[1])**2
    return math.sqrt(px+py)
```

Рисунок 3.12 – Обчислення евклідової відстані

Далі створимо найважливіший метод sense_obstacles (рисунок 3.13), який саме симулює роботу лазерного сенсора, який в свою чергу обстежує оточення. Він визначає лазерні промені у різних напрямках та перевіряє, чи існують перешкоди на шляху, після чого зберігає дані про виявлені перешкоди, додаючи шум, за допомогою функції uncertainty_add.

```
def sense_obstacles(self):
  data=[]
  x1, y1 = self.position[0], self.position[1]
  for angle in np.linspace(0,2*math.pi,60,False):
    x2,y2 = (x1 + self.Range * math.cos(angle), y1 - self.Range * math.sin(angle))
    for i in range(0, 100):
      u = i / 100
      x = int(x2 * u + x1 * (1 - u))
      y = int(y2 * u + y1 * (1 - u))
      if 0<x<self.W and 0<y<self.H:
        color=self.map.get_at((x,y))
        if (color[0],color[1],color[2]) == (0,0,0):
          distance=self.distance((x,y))
          output=uncertainty_add(distance, angle, self.sigma)
          output.append(self.position)
          data.append(output)
          break
  if len(data)>0:
    return data
  else:
    return False
```

Рисунок 3.13 – Метод симуляції роботи лазерного сенсору

Після написання основної логіки нашої програми було створено файл main.py, який буде імпортувати усе, що знаходиться у файлах env.py та sensors.py, далі задаємо початкову інформацію для нашої програми (рисунок 3.14), тут створється об’єкт environment, класу buildEnviroment з розмірами вікна програми (712, 1400) пікселів. Копіюємо оригінальну карту та створимо об’єкт laser, класу LaserSensor, за параметрами дальності 200, початковою картою, та заданою невизначеністю. Після чого карта заповнюється чорним кольором та копіюється для створення підготовчої мапи infomap.

```
enviroment = buildEnviroment((712,1400))
enviroment.originalMap = enviroment.map.copy()
laser = LaserSensor(200,enviroment.originalMap,uncertainty=(0.5,0.01))
enviroment.map.fill((0,0,0))
enviroment.infomap = enviroment.map.copy()
```

Рисунок 3.14 – Задаємо початкову інформацію для нашої програми

Для запуску програми наступним кроком ініціалізовано змінну running, яка вказує на те, що програма виконується. Та починається головний цикл програми. В цьому циклі проводиться перевірка події та стан додатку (рисунок 3.15).

```
running = True

while running:
    sensorON=False
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if pygame.mouse.get_focused():
            sensorON=True
        elif not pygame.mouse.get_focused:
            sensorON=False
```

Рисунок 3.15 – Задаємо початкову інформацію для нашої програми

Під час роботи циклу також постійно перевіряється чи увімкнен сенсор. Після чого програма отримує позицію миші (яка з нашого боку відтворюється як положення мобільного роботу), оновлює позиції лазера та сбирає дані сенсора, які потім відображається (рисунок 3.16).

```
if sensorON:
    position=pygame.mouse.get_pos()
    laser.position=position
    sensor_data = laser.sense_obstacles()
    enviroment.dataStorage(sensor_data)
    enviroment.show_sensorData()
enviroment.map.blit(enviroment.infomap,(0,0))
pygame.display.update()
```

Рисунок 3.16 – Збір даних сенсора

## 3.3 Встановлення та запуск програми

Для використання програми нам знадобиться встановлений Python3 та будь-який редактор коду (у цій роботі було використовано Visual Studio Code).

Для роботи програми необхідно встановити бібліотеки, які в неї використовуються за допомогою команди, яка вказана на рисунку 3.17.

```
○ →  py python3 -m pip install py numpy
```

Рисунок 3.17 – Встановлення необхідних бібліотек для роботи програми

Після завантаження усіх файлів програми (main.py, env.py, sensors.py) в одну теку або робочу директорію. Необхідно запустити головний файл програми (main.py), написавши команду у термінали, як показано на рисунку 3.18.

```
py python3 main.py
```

Рисунок 3.18 – Запуск програми

У разі правильного виконання попередніх пунктів відкриється вікно програми. Вам буде відображено карту і лазерний сенсор, а також буде можливість взаємодії з мишею, щоб побачити дію лідара. Якщо користувач натискає на кнопку закриття вікна, програма завершиться.

## 3.4 Висновки до розділу

Математична модель включає перетворення координат, додавання невизначеності з використанням багатовимірного нормального розподілу і моделювання роботи лазерного сенсора.

Всі ці кроки об'єднуються для створення даних про відстані та кути, які сенсор міг би виміряти в реальному світі, враховуючи випадкові фактори та перешкоди на карті.

# 4 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ ІЗ ПРОГРАМОЮ

## 4.1 Проведення симуляцій

Після запуску програми можна побачити початковий стан карти (рис. 4.1), де наш мобільний робот, ще не рухався і не зібрав інформацію з сенсорів.

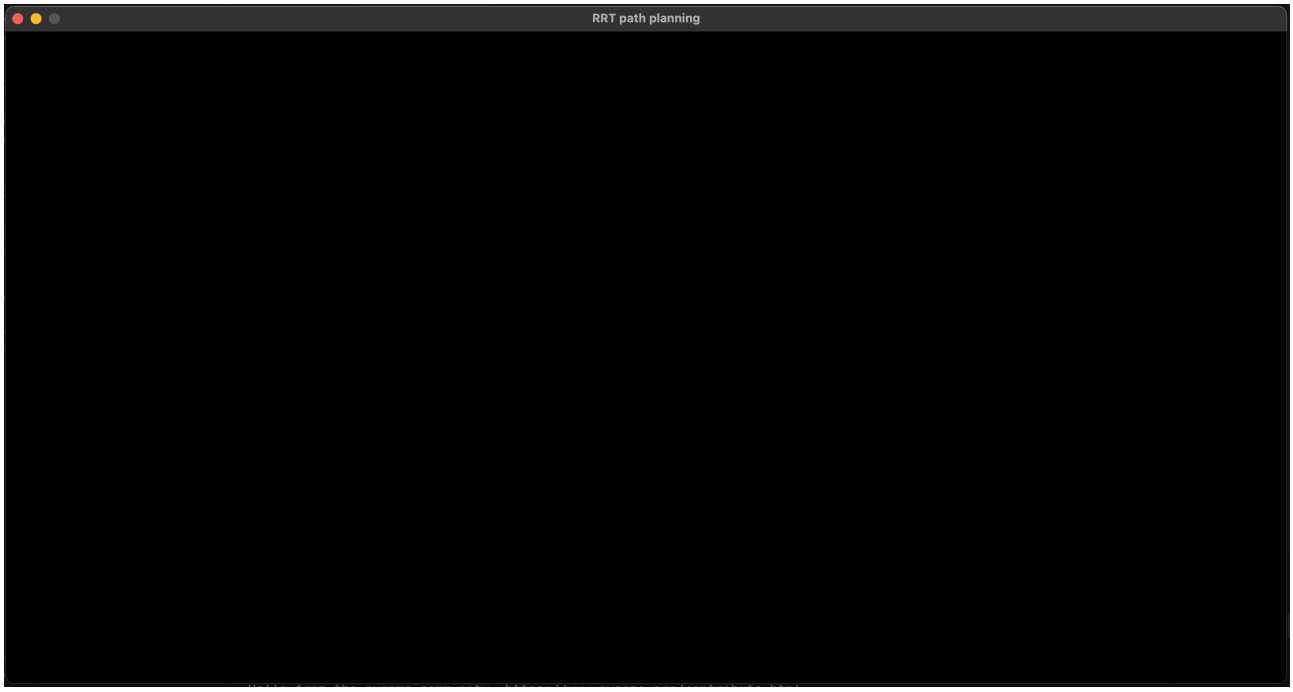


Рисунок 4.1 – Початковий інтерфейс програми.

Якщо почати рухати курсором (який презентуються як мобільний робот), то побачимо, як почне вимальовуватися карта (рисунок 4.2), протягом руху.

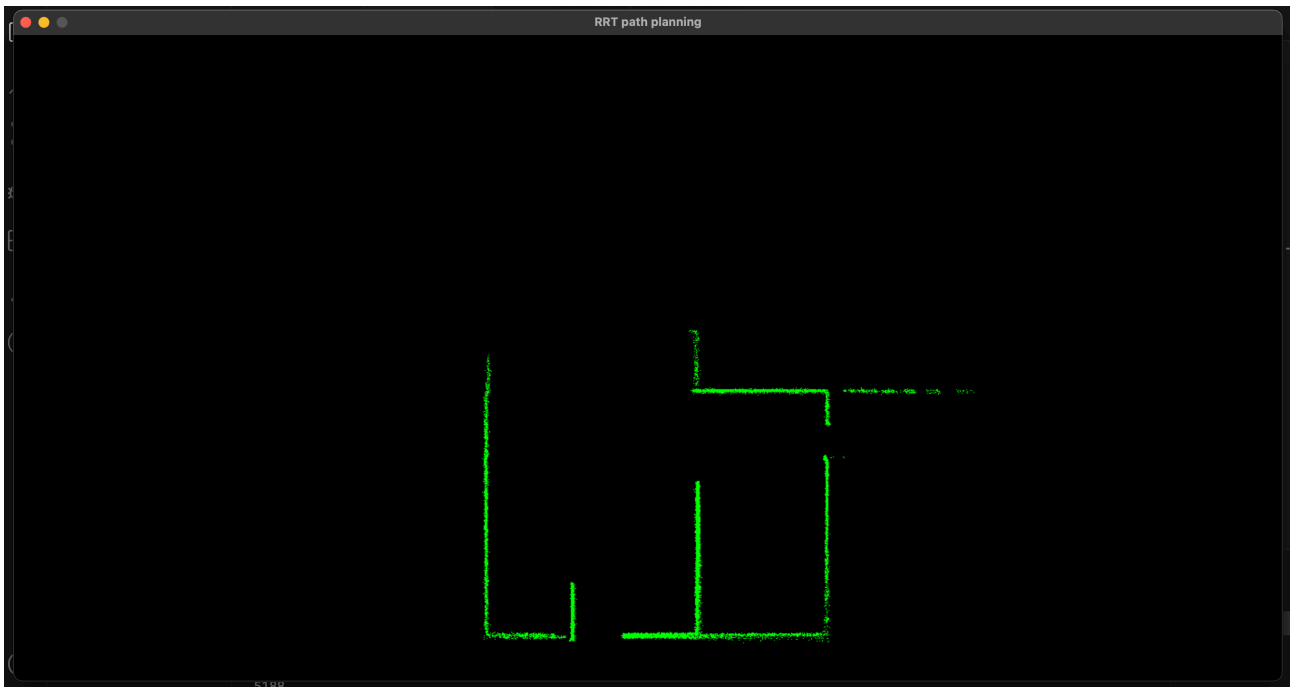


Рисунок 4.2 – Інтерфейс програми після пересування робота.

Вся зібрана сенсорами інформація зберігається. Після того, як наш мобільний робот обійде всю карту, у нього буде збережено повний план будівлі в якої він знаходиться, на основі пройденого ним маршруту (рисунок 4.3).

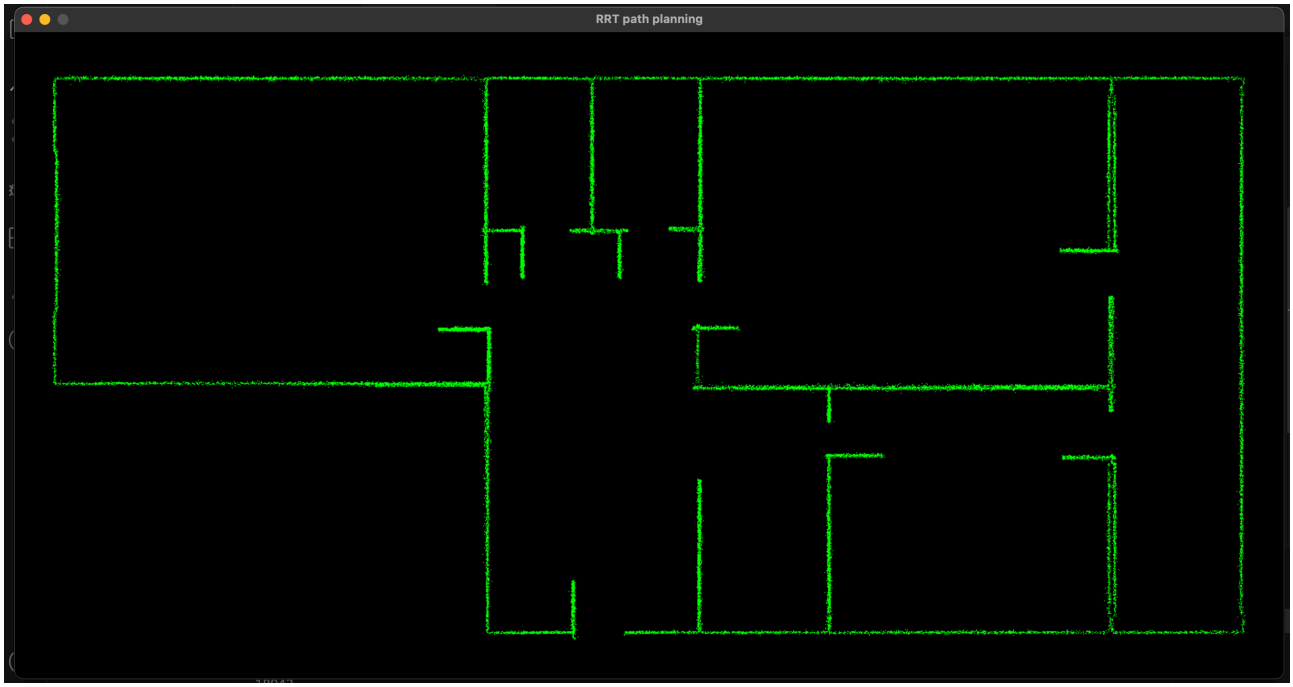


Рисунок 4.3 – Відкрита карта, після того, як робот обійшов всю будівлю.

## 4.2 Охорона праці

Організація охорони праці на підприємстві покладається на роботодавця. Завдання роботодавця також полягає у забезпеченні дотримання прав робітників, передбачених у нормативних та регуляторних актах з охорони праці.

Для створення безпечних і здорових умов праці роботодавець виконує, зокрема, такі функції:

– формує відповідні відділи і призначає уповноважених осіб для нагляду за дотриманням вимог охорони праці, затверджує внутрішні правила, технологічні карти та стандарти;

– затверджує колективний договір та вживає комплексні заходи для підтримання і підвищення рівня охорони праці;

– розробляє програму оптимізації виробництва, впроваджує новітні технології та наукові досягнення;

– відповідає за належний стан промислових будівель, приміщень, виробничого обладнання та машин;

– вживає невідкладних заходів для допомоги постраждалим, організовує виплату компенсації таким особам;

– ніціює проведення неупередженого та об’єктивного розслідування нещасних випадків, вивчає причини, що призвели до аварії та затверджує перелік профілактичних заходів, спрямованих на усунення ризиків виникнення аналогічних причин в подальшому;

– несе персональну відповідальність за рівень охорони праці і порушення іншими особами її вимог;

– здійснює нагляд за додержанням робітниками технологічних процесів, установлених правил поведінки та режиму роботи.

Крім того, роботодавець зобов’язаний за свої кошти забезпечити фінансування та організувати проведення попереднього (під час прийняття на роботу) і періодичних (протягом трудової діяльності) медичних оглядів працівників, зайнятих на важких роботах, роботах із шкідливими чи

небезпечними умовами праці або таких, де є потреба у професійному доборі, щорічного обов'язкового медичного огляду осіб віком до 21 року.

Створення окремої служби охорони праці в компанії є обов’язковим, якщо кількість штату становить 50 і більше осіб. Для менших підприємств дозволяється передати функції служби охорони праці в порядку сумісництва особам, які мають відповідну підготовку, або залучити сторонніх спеціалістів на договірних засадах.

Служба охорони праці підпорядковується безпосередньо роботодавцю.

Фахівці СОП мають право:

– видавати накази щодо усунення допущених недоліків, отримувати від відповідальних осіб відомості, документацію і пояснення з питань охорони праці;

– вимагати відсторонення від роботи працівників, які не пройшли обов’язкового медичного обстеження, навчання, інструктажу, атестації знань і не мають допуску до відповідних робіт;

– за наявності загрози для життя і безпеки робітників — призупиняти виробничий процес;

– ініціювати питання притягнення винних осіб до відповідальності.

Припис спеціаліста з охорони праці може скасувати лише роботодавець.

Додатково, з метою забезпечення пропорційної участі працівників у створенні комфортних та безпечних умов праці за рішенням колективу може створюватися Комісія з питань охорони праці. Висновки такої комісії мають рекомендаційний характер.

Стан охорони праці на підприємстві може відповідати встановленим нормативним вимогам лише за наявності належного рівня фінансування. Відповідальною особою за формування матеріального і грошового забезпечення охорони праці є роботодавець.

Для підприємств, незалежно від форм власності, або фізичних осіб, які відповідно до законодавства використовують найману працю, витрати на

охорону праці становлять не менше 0,5 відсотка від фонду оплати праці за попередній рік.

У бюджетних установах розмір фінансування на охорону праці визначається у колективному договорі з урахуванням фінансових можливостей підприємства.

У колективному договорі сторони передбачають, зокрема, комплексні заходи щодо досягнення встановлених нормативів безпеки, гігієни праці та виробничого середовища, підвищення рівня охорони праці, запобігання виробничому травматизму, професійному захворюванню, аваріям і пожежам; визначають обсяги та джерела фінансування цих заходів.

Які витрати належать до працеохоронних, визначає Перелік заходів та засобів з охорони праці, затверджений постановою КМУ від 27.06.2003 № 994. Цей Перелік, зокрема, передбачає витрати на придбання необхідної літератури; проведення навчання і перевірки знань із питань охорони праці посадових осіб й інших працівників протягом трудової діяльності та організацію лекцій, семінарів і консультацій із зазначених питань.

За недотримання норм витрат на охорону праці роботодавця можуть притягти до відповідальності [12].

## 4.3 Висновки до розділу

У цьому розділі була описана симуляція. В якої було продемонстровано створення карти за допомогою використання сенсорів робота.

Також були описані правила поведінки під час користування персональним комп'ютером на робочому місці.

# ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено аналіз літератури та визначили, що розробка підсистеми управління для мобільних роботів важлива для забезпечення їх здатності орієнтуватися і ефективно виконувати завдання в приміщеннях виробничих просторів. Це включає в себе розробку алгоритмів навігації, систем датчиків, методів взаємодії з оточуючим середовищем та технічних засобів управління. Мобільні роботи можуть бути використані для автоматизації логістики та складського управління, що сприяє оптимізації процесів зберігання та переміщення товарів.

Мобільні роботи мають потенціал підвищити ефективність виробництва, знизити операційні витрати та покращити безпеку на робочих місцях, що робить їх важливими інструментами в сучасній промисловості.

Під час написання кваліфікаційної роботи було промодельовано рух мобільного робота у динамічному просторі. Розроблено код для симуляції прийняття рішень для орієнтації мобільного робота, який під час свого руху здатен генерувати карту, на фоні оточуючих його перешкод. Описано етапи установки програми та проведено її тестування.

Також були описані математичні моделі за допомогою яких ми змогли перетворити данні від сенсорів та обчислити координати точок у просторі.

Реалізували сенсор виявлення перешкод з врахуванням невизначеності та розсіювання даних

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 55 с.

3. Максимова С. С., Канаєв В. Д. «Розробка підсистеми керування мобільного роботу для орієнтації в виробничому просторі»», матеріали VII-ої Міжнародної конференції «Виробництво & Мехатронні Системи 2023», Харків, 2023 р. – с.: 53 – 56.

4. Савенко Ю.Н. Мобильные роботы: экспортный потенциал и перспективы [Текст] / Ю.Н. Савенко, С. А. Сарапулов // Інформаційний вісник «Інновації та промисловість». Київ. №2, 2007.

5. Мобільні Роботи: Можливості, Перспективи, Проблеми. Шевченко О.О., Юсупов В.Т., Юрченко О.Д. Науковий керівник – к.т.н., доц. Грицюк В.Ю. Харківський національний університет радіоелектроніки

6. Pedrosa, E., L. Reis, C. M. D. Silva and H. S. Ferreira. Autonomous Navigation with Simultaneous Localization and Mapping in/outdoor. 2020.

7. Labbé, M, Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. J Field Robotics. 2019; 35: 416– 446.

8. Mathieu Labbé and François Michaud. Online Global Loop Closure Detection for LargeScale Multi-Session Graph-Based SLAM. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2661–2666, 2014.

9. Дослідження методів SLAM для навігації мобільного робота всередині приміщень [Електронний ресурс – Режим доступу: https://habr.com/ua/articles/560856/

10. J. J. Kuffner Jr., “RRT-Connect: an Efficient Approach to SingleQuery Path Planning,” in Proc of IEEE Int. Conf. of Robotics and Automation, San Fransisco, USA, Apr. 2000

11. CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework Eric Rohmer, Surya P. N. Singh and Marc Freese

12 Організація охорони праці на підприємстві [Електронний ресурс – Режим доступу: https://pro-op.com.ua/article/378-organzatsya-ohoroni-prats