Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп’ютерних систем та мереж

Пояснювальна записка
до кваліфікаційної роботи магістра
за спеціальністю 123 «Комп’ютерна інженерія»

на тему: РОЗРОБКА ДРОНУ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОМАТИЧНОЇ НАВІГАЦІЇ ТА ОБ'ЄКТНОГО ВИЗНАЧЕННЯ

Проєктував _____ Б. Г. Кісельов

Керівник роботи _____ А. О. Сенько

Нормоконтроль _____ Д. І. Кузнєцов

Завідувач кафедри _____ А. І. Купін

Кривий Ріг
2024

Криворізький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних систем та мереж

Ступінь вищої освіти магістр
Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри, голова циклової комісії

_____ А. І. Купін

"____" _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

_____
(прізвище, ім'я, по батькові)

1. Тема роботи _____
_____
_____
керівник роботи _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "___"_____ 20__року №__
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи _____
_____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____
_____
_____
_____
_____
_____
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: містить 100 сторінок, 33 рисунки, 15 таблиць, 99 використаних джерел. Мета проєкту – дослідити сучасні методи та алгоритми для створення системи автономної навігації дронів, що використовує штучний інтелект, алгоритми планування маршруту та побудову карт оточення.

Проєкт складається з чотирьох розділів.

У першому розділі подано ґрунтовний огляд сучасних підходів до навігації автономних дронів. Описано використання алгоритмів SLAM, фільтра Калмана та методів обробки даних із різноманітних сенсорів, таких як IMU, камери та LiDAR. Також детально проаналізовано переваги та недоліки існуючих технологій, з урахуванням їх застосування в реальних умовах.

Другий розділ присвячено розробці математичних моделей та алгоритмів для автономної навігації. Зокрема, описано принцип роботи EKF SLAM для побудови карт оточення, інтеграцію даних GPS та IMU із застосуванням фільтра Калмана, а також алгоритм A* для прокладання оптимальних маршрутів у динамічному середовищі.

У третьому розділі сформульовано основні завдання, визначено початкові умови та описано підходи до реалізації алгоритмів. Зосереджено увагу на математичних основах опрацювання сенсорних даних, моделюванні траєкторій та визначенні точного положення дрона. Окремо проведено тестування ключових компонентів системи для перевірки їхньої працездатності.

Четвертий розділ містить результати експериментальних досліджень, оцінку точності та стабільності роботи алгоритмів. Проведено аналіз результатів роботи системи у різних середовищах, включаючи складні сценарії з перешкодами. Оцінено точність позиціонування, стабільність побудови карт та ефективність маршрутного планування.

АВТОНОМНА НАВІГАЦІЯ, GPS, EKF SLAM, ФІЛЬТР КАЛМАНА, A* АЛГОРИТМ, ДРОНИ, СЕНСОРИ, МАПА ОТОЧЕННЯ.

Explanatory note: contains 100 pages, 33 figures, 15 tables, 99 references. The aim of the project is to investigate modern methods and algorithms for creating an autonomous drone navigation system that uses artificial intelligence, route planning algorithms, and environment mapping.

The project consists of four sections.

The first chapter provides a thorough overview of current approaches to autonomous drone navigation. It describes the use of SLAM algorithms, Kalman filter, and methods for processing data from various sensors such as IMUs, cameras, and LiDAR. The advantages and disadvantages of existing technologies are also analyzed in detail, taking into account their application in real-world conditions.

The second section is devoted to the development of mathematical models and algorithms for autonomous navigation. In particular, we describe the principle of EKF SLAM for building environment maps, integration of GPS and IMU data using the Kalman filter, and the A* algorithm for optimal routes in a dynamic environment.

The third section formulates the main tasks, defines the initial conditions, and describes the approaches to the implementation of the algorithms. Attention is focused on the mathematical foundations of sensor data processing, trajectory modeling, and determining the exact position of the drone. Separately, the key components of the system were tested to verify their performance.

The fourth section contains the results of experimental studies, an assessment of the accuracy and stability of the algorithms. The results of the system's operation in different environments, including complex scenarios with obstacles, are analyzed. Positioning accuracy, mapping stability, and route planning efficiency are evaluated.

AUTONOMOUS NAVIGATION, GPS, EKF SLAM, KALMAN FILTER, A* ALGORITHM, DRONES, SENSORS, ENVIRONMENT MAP.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

– **AI (Artificial Intelligence)** – Штучний інтелект;
– **A*** – Алгоритм пошуку найкоротшого шляху;
– **EKF (Extended Kalman Filter)** – Розширений фільтр Калмана;
– **FPS (Frames Per Second)** – Кількість кадрів на секунду;
– **GPS (Global Positioning System)** – Глобальна система позиціонування;
– **GNSS (Global Navigation Satellite System)** – Глобальна навігаційна супутникова система;
– **IMU (Inertial Measurement Unit)** – Інерційний вимірювальний блок;
– **KF (Kalman Filter)** – Фільтр Калмана;
– **LIDAR (Light Detection and Ranging)** – Технологія оптичного вимірювання відстаней;
– **MSE (Mean Squared Error)** – Середньоквадратична похибка;
– **RGB-D** – Камера, що використовує кольорове (RGB) та глибинне зображення;
– **SLAM (Simultaneous Localization and Mapping)** – Одночасна локалізація та побудова карти;
– **YOLO (You Only Look Once)** – Алгоритм глибокого навчання для виявлення об’єктів;
– **SSD (Single Shot Multibox Detector)** – Метод одноетапного визначення об’єктів;
– **RNN (Recurrent Neural Network)** – Рекурентна нейронна мережа;
– **CNN (Convolutional Neural Network)** – Згорткова нейронна мережа;
– **FOV (Field of View)** – Поле зору;
– **UKF (Unscented Kalman Filter)** – Нековаріаційний фільтр Калмана;
– **ROS2 (Robot Operating System 2)** – Операційна система для роботів.

ВСТУП

В останні десятиліття технологічний прогрес значно вплинув на розвиток безпілотних літальних апаратів (БПЛА), які відомі також як дрони. Їх використання стало поширеним у багатьох сферах людської діяльності, таких як військова справа, наукові дослідження, сільське господарство, промисловість, логістика та безпека. Завдяки своїй автономності та здатності до виконання широкого спектра завдань, дрони є інноваційним рішенням для автоматизації процесів у середовищах, де присутність людини є небезпечною або неможливою.

Однією з ключових особливостей сучасних дронів є можливість автономної навігації, яка базується на алгоритмах штучного інтелекту та математичних методах обробки даних. Інтеграція штучного інтелекту дозволяє дронам не лише переміщуватись у просторі без втручання людини, але й самостійно приймати рішення на основі аналізу даних з сенсорів, комп'ютерного зору та інших джерел інформації. Завдяки здатності дронів розпізнавати об'єкти в реальному часі, а також ефективно будувати та коригувати маршрут, вони можуть виконувати завдання спостереження, моніторингу інфраструктури, рятувальних операцій та дослідження навколишнього середовища.

Основною метою цієї роботи є підбір алгоритмів та дослідження перспективи об’єднання їх у автономну систему дрону, що поєднує штучний інтелект та математичні алгоритми для реалізації автоматичної навігації й розпізнавання об’єктів. Розробка передбачає використання сучасних підходів, зокрема комп’ютерного зору/методів машинного навчання для розпізнавання об’єктів, а також математичних алгоритмів, таких як фільтри для обробки сенсорних даних і технології одночасного локалізування та побудови карти (SLAM). Це дозволить забезпечити точне позиціонування та стабільну роботу дрону навіть за умови обмежених обчислювальних ресурсів або втрати GPS-сигналу.

Наукова новизна роботи полягає у поєднанні методів штучного інтелекту та математичних алгоритмів для вирішення завдання автономної навігації та розпізнавання об’єктів у складних умовах. В рамках даної роботи буде інтегровано методи комп’ютерного зору та математичні методи фільтрації для підвищення точності й стабільності роботи дрону у динамічному середовищі, що забезпечить високий рівень автономності в умовах мінливих зовнішніх факторів.

Актуальність роботи зумовлена зростаючою потребою у безпілотних системах, здатних працювати автономно й ефективно виконувати завдання в складних умовах. Запропоновані рішення можуть знайти застосування в цивільних і військових галузях, де необхідна висока автономність і точність. Підхід, що поєднує штучний інтелект і математичні алгоритми, дозволяє розробити дрон нового покоління, здатний виконувати складні завдання з мінімальним втручанням людини, що є важливим у науково-технічній та соціально-економічній сферах.

Таким чином, це дослідження спрямоване на створення автономного дрону з високим рівнем інтелектуальності, який зможе ефективно функціонувати в динамічному середовищі та вирішувати завдання навігації й об’єктного розпізнавання, покладаючись на можливості сучасних технологій.

1 СУЧАСНІ ПІДХОДИ ТА ВИКЛИКИ АВТОНОМНОЇ НАВІГАЦІЇ ДРОНІВ

З розвитком технологій автономної навігації дрони набувають дедалі ширшого застосування в різних галузях, таких як логістика, аграрний сектор, рятувальні операції та військова справа. Виконання дронами завдань у режимі автономної роботи стає пріоритетним завданням, адже це дозволяє зменшити потребу в постійному контролі з боку людини і значно підвищити ефективність їх застосування. У цьому розділі ми розглянемо сучасні технології, що використовуються для позиціонування дронів, виклики, пов'язані з автономною навігацією, а також можливості, які надають комп'ютерний зір і штучний інтелект для покращення точності та адаптивності навігації.

1.1. Огляд технологій позиціонування в автономних системах: IMU, GPS, компас, комп'ютерний зір, оптичний потік

Для забезпечення автономного польоту дронів необхідні системи, що дозволяють їм визначати своє положення у просторі. Сьогодні найпоширенішими засобами позиціонування є GPS, IMU (інерційна вимірювальна одиниця), компас, а також технології комп'ютерного зору. Розглянемо детальніше кожен із цих компонентів.

1.1.1. GPS (Глобальна система позиціонування)

GPS надає точні координати дрону у просторі. Це ключовий інструмент для навігації на відкритій місцевості, де сигнал стабільний. Однак GPS має свої обмеження, зокрема у міських умовах та в приміщеннях, де сигнал може бути значно ослабленим або повністю відсутнім [1, 2].

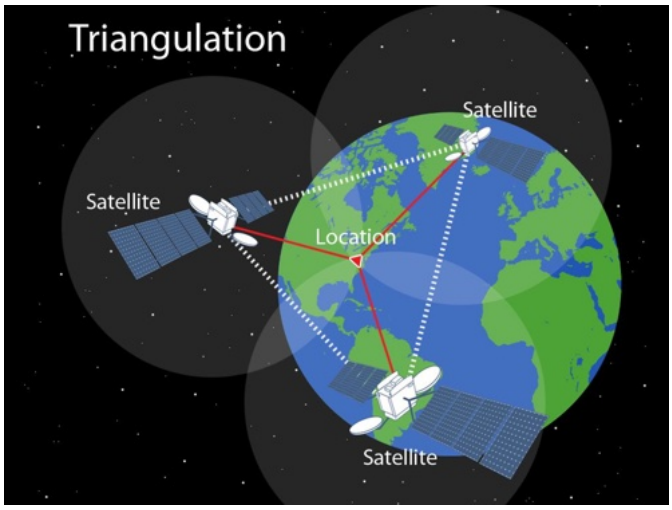


Рисунок 1.1 – Принцип роботи модуля GPS [1]

1.1.2. IMU (Інерційна вимірювальна одиниця)

IMU складається з акселерометра, гіроскопа і часто компаса. Вона вимірює прискорення, кутову швидкість і орієнтацію дрону, що дозволяє оцінювати його положення на основі інерційних даних. Проте через накопичення похибок з часом (дрейф) самостійно IMU не забезпечує достатньої точності, що потребує додаткових методів корекції [3].

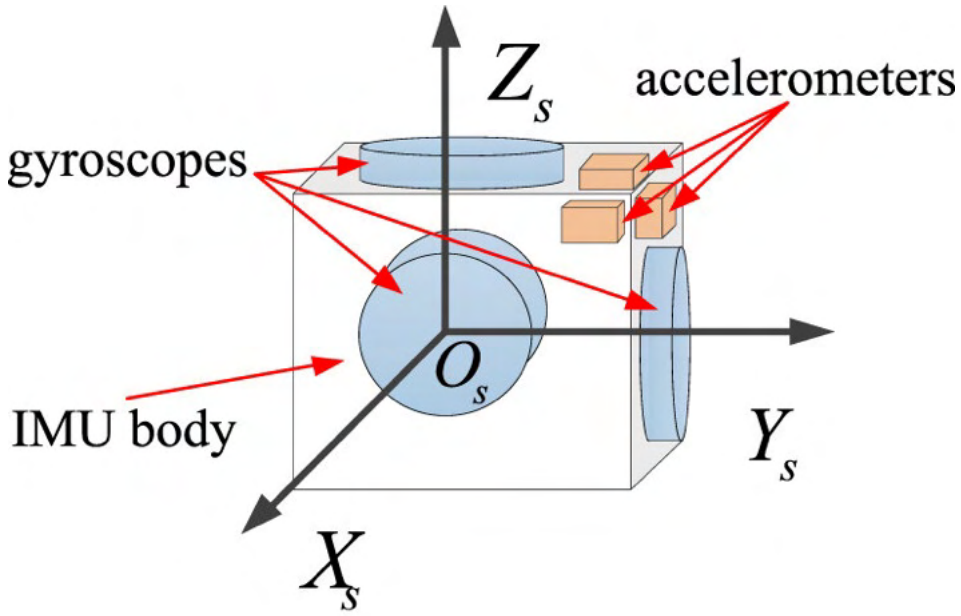


Рисунок 1.2 – Абстрактна будова IMU

1.1.3. Комп'ютерний зір

Комп'ютерний зір – це технологія, що дозволяє дрону сприймати навколишнє середовище за допомогою камер. Використання комп'ютерного зору, зокрема методів розпізнавання об'єктів і SLAM (Simultaneous Localization and Mapping), дає змогу дрону орієнтуватися у просторі навіть за відсутності GPS, створюючи карти оточення і визначати своє відносне положення [4, 5].

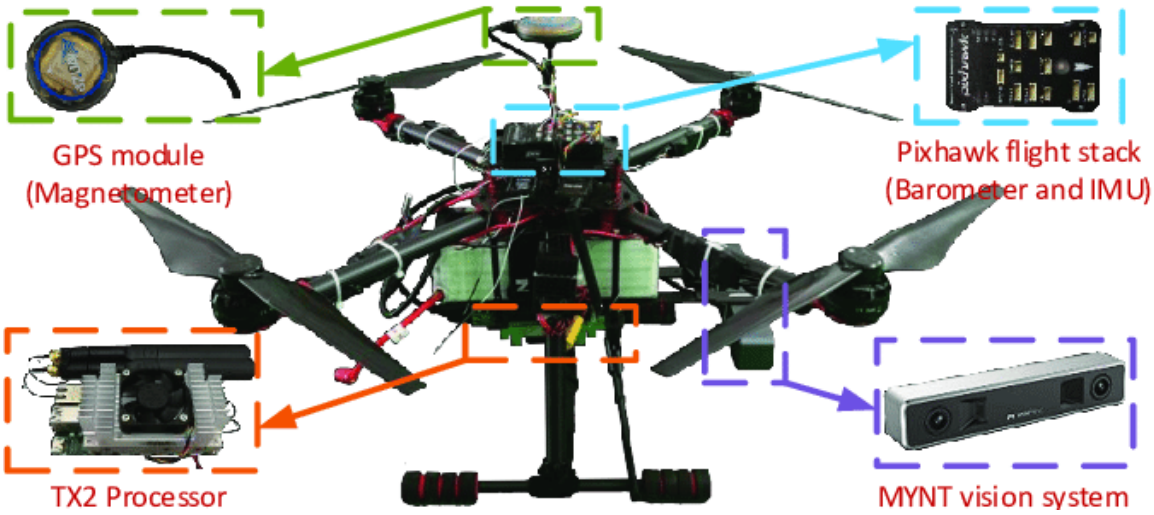


Рисунок 1.3 – Вигляд основних сенсорів дрона [6]

1.1.4 Оптичний потік

Оптичний потік є ключовим поняттям у комп'ютерному зорі, що описує видимий рух об'єктів у кадрі або зміну яскравості пікселів між послідовними зображеннями. Ця технологія широко використовується для визначення руху, оцінки напрямку та швидкості переміщення об'єктів, а також для аналізу сцени у динамічних умовах.

Принцип роботи оптичного потоку полягає у визначенні змін між двома або більше послідовними зображеннями, отриманими камерою. Для цього аналізується, як яскравість кожного пікселя змінюється з часом. Відповідно до цих змін формується векторне поле, яке відображає напрямок і величину руху.

Для обчислення оптичного потоку використовуються кілька підходів. Локальні методи, такі як метод Лукаса-Канаде, аналізують невеликі області зображення, припускаючи, що всі пікселі в межах вікна рухаються однаково. Глобальні методи, зокрема метод Хорна-Шунка, забезпечують цілісне уявлення про рух, але вимагають більше обчислювальних ресурсів. Сучасні нейронні мережі значно покращили точність і стійкість до шуму, дозволяючи використовувати оптичний потік у складних середовищах [7, 8].

Дані, отримані за допомогою оптичного потоку, дозволяють визначати напрямок і швидкість руху об'єктів або камери. У поєднанні з іншими даними ці результати можуть використовуватись для тривимірної реконструкції сцени або побудови моделей динамічного середовища (Рисунок 1.4).

Незважаючи на переваги, оптичний потік має свої обмеження. Наприклад, рівні або однорідні поверхні, такі як стіни або вода, не забезпечують достатньо текстур для точного аналізу. Також похибки можуть виникати через шум у зображеннях або різкі зміни освітлення. У випадках швидкого руху між кадрами обчислення потоку може бути ускладненим, що потребує застосування додаткових корекційних алгоритмів [9].

Використання оптичного потоку є важливим у системах автономної навігації, особливо у середовищах, де GPS недоступний. Наприклад, у дронах ця технологія допомагає визначати напрямок руху, уникати перешкод і коригувати траєкторію польоту. Завдяки здатності до інтеграції з іншими технологіями, такими як IMU або стереозір, оптичний потік є потужним інструментом для створення автономних систем [10].

1.2. Методи об'єднання даних з різних сенсорів для підвищення точності (фільтр Калмана, методи дедвекінгу)

Жоден окремий сенсор не може гарантувати повну точність і надійність через наявність шумів та похибок вимірювань та/або інших нюансів функціонування. Тому для забезпечення надійної та точної навігації дрону важливо об'єднувати дані з різних сенсорів. У цьому розділі розглянемо основні методи інтеграції даних, які дозволяють компенсувати недоліки кожного сенсора та підвищити загальну точність системи [11].

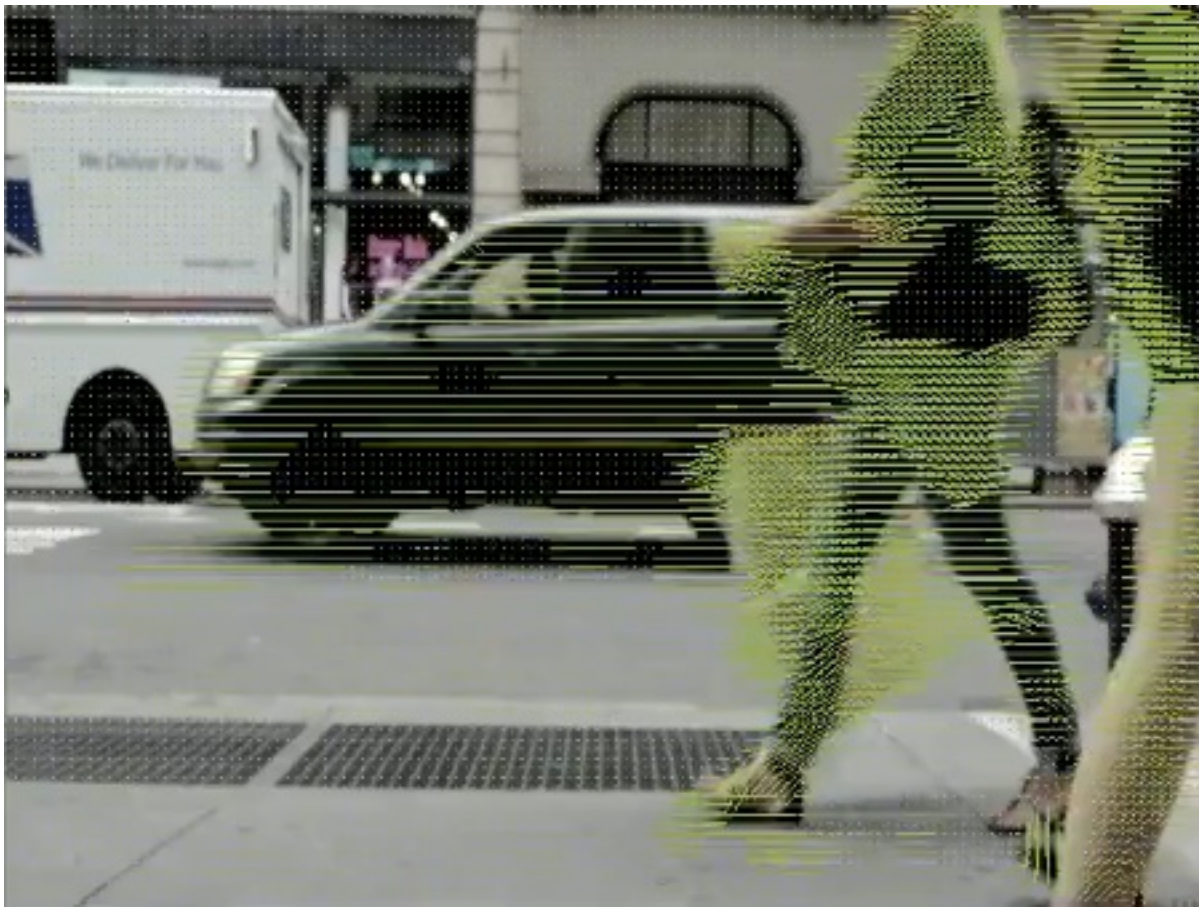


Рисунок 1.4 – Приклад векторного поля оптичного потоку, що показує напрямок і швидкість руху в кадрі [12]

1.2.1 Фільтр Калмана

Фільтр Калмана, названий на честь американського математика і інженера Рудольфа Е. Калмана, був розроблений в кінці 1950-х та початку 1960-х років. Рудольф Калман вперше опублікував свою роботу з фільтрації лінійних динамічних систем у статті "A New Approach to Linear Filtering and Prediction Problems" у 1960 році. Він співпрацював з Stanley F. Shmidt, який допоміг розширити теоретичну базу і застосувати ці концепції на практиці. Робота Калмана стала основою для подальших досліджень і розробок у цій галузі [13, 14].

Одними з перших сфер застосування фільтру Калмана були авіаційна та космічна галузі. У 1960-х роках NASA використовувала фільтр Калмана для навігації та керування космічними апаратами. Впровадження фільтру дозволило значно підвищити точність і надійність систем управління польотами, що було критично важливим для успішного виконання космічних місій. Наприклад, покращений фільтр Калмана застосовувався в програмі Apollo для точного визначення траєкторії польоту та посадки на Місяць [13, 14, 15, 16].

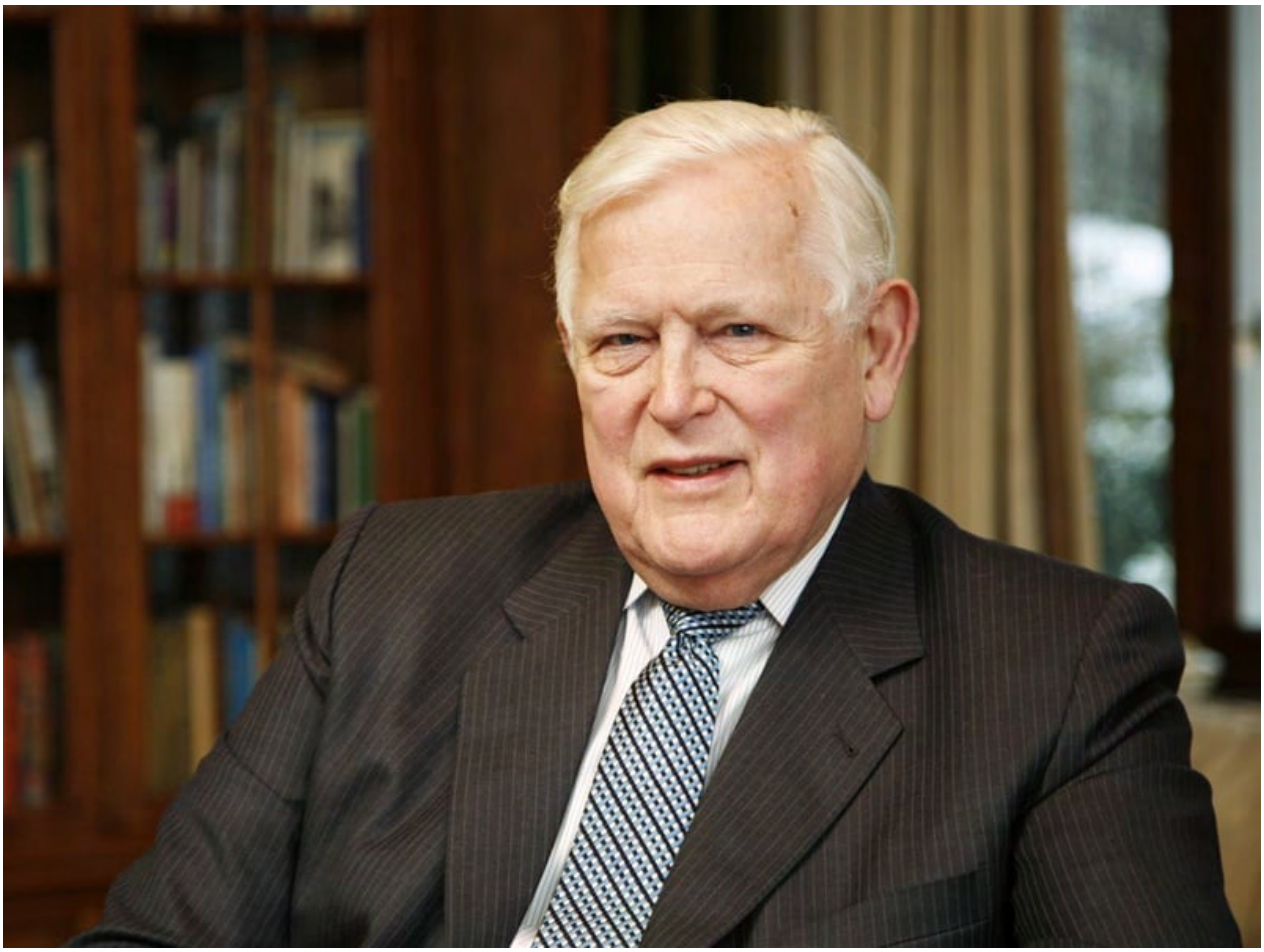


Рисунок 1.5 – Рудольф Калман [15]

Рисунок 1.6 – Логотип NASA [17]

Рисунок 1.7 – Програма Apollo [16]

Фільтр Калмана є оптимальним рекурсивним алгоритмом оцінки стану системи в умовах наявності шуму та похибок вимірювань. Його універсальність і ефективність зробили цей метод основним інструментом у задачах навігації, керування, робототехніки та обробки сигналів. Основною перевагою фільтра Калмана є його здатність комбінувати інформацію з різних сенсорів, враховуючи їх точність, а також згладжувати випадкові флуктуації даних, які виникають через шум.

Цей алгоритм ґрунтується на застосуванні моделей стану та вимірювань, які формалізують динаміку системи та залежність виміряних даних від стану. Завдяки рекурсивному характеру, фільтр Калмана дозволяє оцінювати поточний стан системи, використовуючи як попередні оцінки, так і нові дані, що надходять у реальному часі.

Основні застосування фільтра Калмана:

- **Згладжування даних від сенсорів**: Зменшення впливу випадкових шумів на вимірювання отриманих з сенсорів;
- **Корекція похибок інерційних вимірювань**: Компенсація дрейфу та накопичення похибок з часом у даних з IMU;
- **Інтеграція даних з різних джерел**: Поєднання даних з GPS, IMU, компаса, камер та інших сенсорів для отримання більш точної оцінки стану.

Фільтр Калмана є одним з найефективніших інструментів для обробки і фільтрації даних у реальному часі, що робить його незамінним у навігаційних

системах сучасних безпілотних літальних апаратів (БПЛА). Основною перевагою фільтра Калмана є його здатність надавати оптимальні оцінки стану системи навіть при наявності значних шумів і неточностей у вимірюваннях. Це особливо важливо для БПЛА, які широко використовуються у різних сферах, таких як логістика, моніторинг, зйомка та розвідка.

Інженерні підходи до застосування фільтра Калмана включають його інтеграцію у різні системи управління та навігації. У системах навігації дронів фільтр Калмана використовується для покращення точності позиціонування шляхом фільтрації шумових даних з GPS-навігаторів. Це дозволяє дрону підтримувати стабільну траєкторію польоту і точно виконувати поставлені завдання, навіть в умовах сильних електромагнітних перешкод або обмеженого доступу до супутникових сигналів.

У системах управління польотами фільтр Калмана дозволяє забезпечити високу точність і надійність контролю параметрів польоту, таких як висота, швидкість, орієнтація і положення в просторі. Наприклад, в авіаційних системах фільтр Калмана використовується для обробки даних з інерційних навігаційних систем (INS) та GPS, що дозволяє отримати точні координати літака навіть при втраті сигналу GPS.

Фільтр Калмана також використовується в системах стабілізації дронів, де він дозволяє точно оцінювати положення і орієнтацію дрону в просторі, забезпечуючи стабільний і плавний політ. Це особливо важливо для виконання складних маневрів та стабілізації дрону при зйомках або моніторингу. Застосування фільтра Калмана дозволяє значно зменшити вплив турбулентності та інших зовнішніх факторів на політ дрону.

У сфері робототехніки фільтр Калмана використовується для навігації автономних роботів, дозволяючи їм точно визначати своє положення і планувати траєкторію руху. Наприклад, у роботах для доставки вантажів фільтр Калмана дозволяє забезпечити точну навігацію навіть у складних міських умовах з великою кількістю перешкод і шумів. Завдяки цьому роботи можуть точно і надійно доставляти вантажі до місця призначення.

В автомобільній промисловості фільтр Калмана використовується для системи стабілізації курсу автомобіля (ESP), адаптивного круїз-контролю, системи допомоги при парковці та інших автоматичних систем керування. Використання фільтра Калмана дозволяє значно підвищити безпеку та комфорт водіння за рахунок точного контролю параметрів руху автомобіля і швидкого реагування на зміну умов дорожнього руху [18].

Реалізація фільтра Калмана у програмних системах для дронів включає використання мов програмування високого рівня, таких як Python або MATLAB. Ці мови дозволяють ефективно обробляти дані у реальному часі і забезпечують необхідну швидкість і точність обчислень. Використання фільтра Калмана у навігаційних системах дронів дозволяє значно підвищити точність їх позиціонування і маневреність, що особливо важливо для виконання завдань у складних умовах.

У мобільних роботах фільтр Калмана використовується для інтеграції даних з різних сенсорів, таких як Лідар, камери, інерційні вимірювальні одиниці (IMU) та GPS. Це дозволяє отримати точну оцінку положення і орієнтації робота в просторі, що є важливим для його автономного переміщення і виконання складних завдань. Наприклад, у роботах для дослідження підводного світу фільтр Калмана використовується для інтеграції даних з гідролокаторів і інерційних навігаційних систем, що дозволяє отримати точну карту дна і забезпечити безпечне переміщення робота під водою.

Фільтр Калмана також застосовується в галузі дистанційного зондування, де він використовується для обробки даних з супутників та літальних апаратів. Це дозволяє отримати точні дані про стан атмосфери, поверхні землі і океанів, що є важливим для прогнозування погоди, моніторингу кліматичних змін та управління природними ресурсами. Застосування фільтра Калмана дозволяє значно зменшити вплив шумів і помилок вимірювань, що дозволяє отримати більш точні і надійні дані.

Таким чином, фільтр Калмана є потужним і універсальним інструментом для обробки і фільтрації даних у реальному часі, що знайшов широке застосування у різних інженерних системах. Його здатність забезпечувати високу точність і надійність оцінок стану системи навіть при наявності значних шумів і неточностей робить його незамінним для вирішення складних завдань у різних галузях техніки.

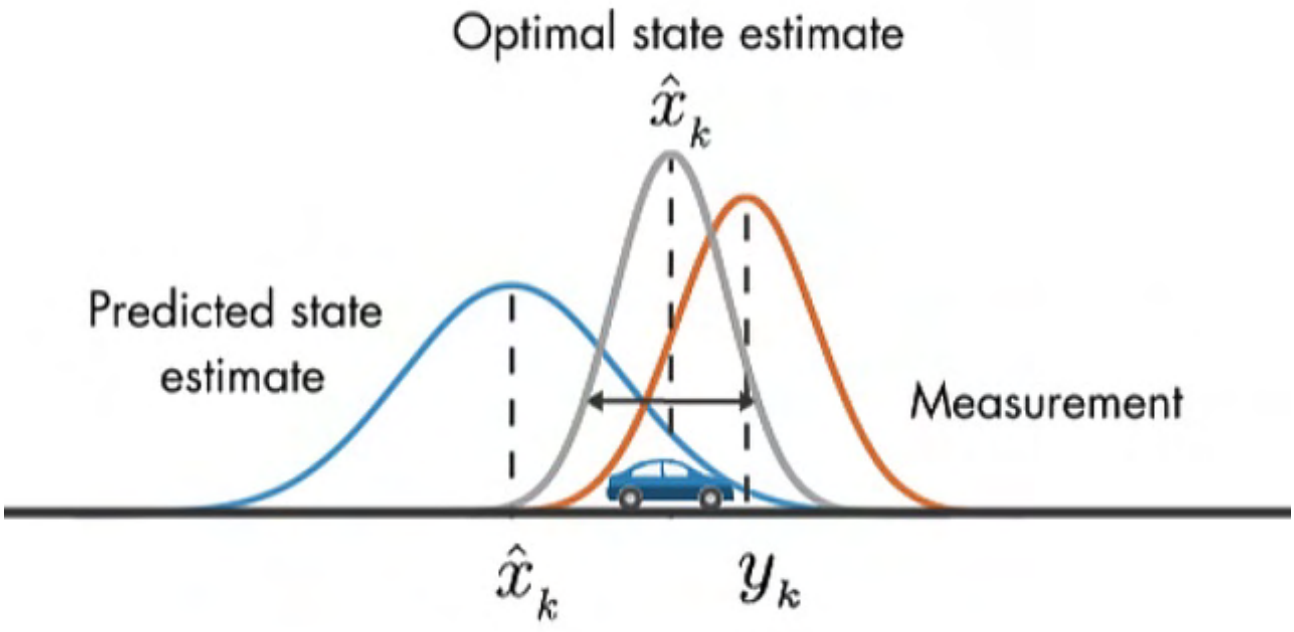


Рисунок 1.8 – Приклад використання фільтру у GPS-навігації [19]

Реалізація алгоритму фільтра Калмана включає кілька основних кроків:

1. **Прогноз стану**: Прогнозування наступного стану дрону на основі поточного стану і керування;

2. **Прогноз коваріаційної матриці**: Прогнозування коваріаційної матриці, яка відображає невизначеність прогнозу;

3. **Оновлення Калманівського посилення**: Розрахунок коефіцієнтів посилення фільтра Калмана, які визначають вплив нових вимірювань на корекцію стану;

4. **Оновлення стану**: Корекція прогнозованого стану дрону на основі нових вимірювань;

5. **Оновлення коваріаційної матриці**: Оновлення коваріаційної матриці після корекції стану для відображення нової невизначеності.

Розробка математичної моделі є ключовим етапом у формалізації задачі дослідження. Модель повинна включати рівняння стану та рівняння вимірювань, які описують динаміку руху дрону та взаємодію з навігаційними системами. Основні елементи моделі включають:

– **Рівняння стану**: Описують еволюцію стану дрону з часом. Вектор стану x_k може включати положення, швидкість та інші параметри, що характеризують рух дрону:

$$x_k = Ax_{k-1} + Bu_k + w_k$$

де A – матриця переходу стану, B – матриця керування, u_k – вектор керування, w_k – шум процесу.

– **Рівняння вимірювань**: Описують залежність між вимірюваннями та станом дрону. Вектор вимірювань z_k може включати дані з GPS-навігатора:

$$z_k = Hx_k + v_k$$

де H – матриця вимірювань, v_k – шум вимірювань.

Реалізація алгоритму фільтра Калмана включає кілька етапів:

1. **Прогноз стану**:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k$$

Це рівняння використовується для прогнозування наступного стану дрону на основі поточного стану і керування.

2. **Прогноз коваріаційної матриці**:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q$$

Де $P_{k|k-1}$ – прогнозована коваріаційна матриця, Q – матриця коваріації шуму процесу.

3. **Оновлення Калманівського посилення**:

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1}$$

Де K_k – Калманівське посилення, R – матриця коваріації шуму вимірювань.

4. **Оновлення стану**:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1})$$

Це рівняння використовується для корекції прогнозованого стану дрону на основі нових вимірювань.

5. **Оновлення коваріаційної матриці**:

$$P_{k|k} = (I - K_kH)P_{k|k-1}$$

Це рівняння використовується для оновлення коваріаційної матриці після корекції стану [20, 21].

Таким чином, загальний алгоритм фільтру Калмана виглядає наступним чином, зображеним на Рисунку 1.9:

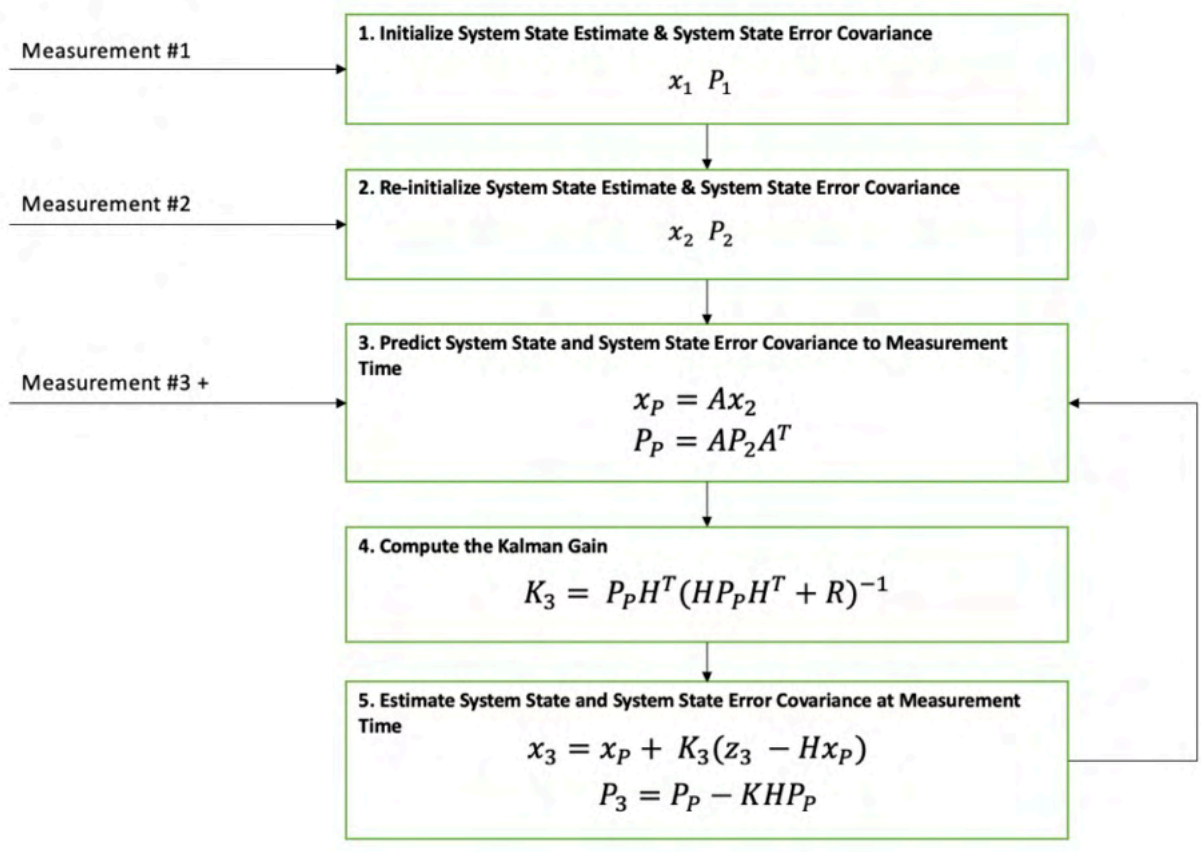


Рисунок 1.9 – Схема роботи фільтра Калмана [22]

Розширений фільтр Калмана (EKF):

Оскільки моделі руху дрону та вимірювань можуть бути нелінійними, застосовується розширений фільтр Калмана (EKF), який лінеаризує нелінійні функції за допомогою розкладу Тейлора [23, 24].

Застосування фільтра Калмана для інтеграції даних:

- **Поєднання IMU та GPS:** GPS надає абсолютні координати, але з меншою частотою та можливими затримками. IMU надає високочастотні дані про прискорення та кутову швидкість, але схильна до накопичення похибок. Фільтр Калмана дозволяє об'єднати ці дані для отримання точної та стабільної оцінки положення [25];
- **Інтеграція з даними SLAM:** Фільтр Калмана може поєднувати локальні відносні координати, отримані з SLAM, з абсолютними координатами GPS та даними з IMU, забезпечуючи точне позиціювання дрону [26].

Переваги використання фільтра Калмана:

- **Зменшення шуму та похибок:** Покращення якості вимірювань шляхом згладжування даних;
- **Об'єднання різних типів даних:** Можливість інтегрувати дані з різних сенсорів з урахуванням їх похибок;
- **Рекурсивність алгоритму:** Не потребує збереження всієї історії вимірювань, що економить обчислювальні ресурси.

Недоліки:

- **Потреба у точній моделі системи:** Неточності в моделі руху або вимірюваннях можуть призвести до некоректних оцінок;
- **Складність реалізації EKF:** Нелінійні системи потребують додаткових обчислень для лінеаризації [23, 24, 25, 27, 28, 29].

1.2.2 Методи дедвекінгу

Дедвекінг (Dead Reckoning) – це метод визначення поточного положення на основі відомого початкового положення та оцінки пройденого шляху за даними швидкості та напрямку руху. Дедвекінг широко використовується в навігаційних системах, де немає можливості отримувати зовнішні дані про положення [30].

Переваги:

- **Автономність:** Не залежить від зовнішніх сигналів або інфраструктури;
- **Швидка обробка даних:** Використовує лише внутрішні сенсори (IMU), що забезпечує високу частоту оновлення;

Недоліки:

- **Накопичення похибок з часом:** Помилки в вимірюваннях IMU призводять до поступового збільшення похибки позиціювання;

- **Потреба в корекції:** Для підтримання точності необхідно періодично коригувати положення за допомогою зовнішніх джерел (наприклад, GPS або SLAM) [31].

Таблиця 1.1 – Порівняння методів об'єднання даних [23, 24, 31]

Метод	Переваги	Недоліки
Фільтр Калмана	Зменшує шум, об'єднує різні дані	Потребує точної моделі системи
Дедвекінг	Автономність, швидкість	Накопичення похибок, потребує корекції

1.3. Застосування комп'ютерного зору для локалізації та розпізнавання об'єктів: SLAM, розпізнавання орієнтирів і об'єктів

Комп'ютерний зір є ключовим компонентом автономної навігації, особливо в умовах відсутності GPS сигналу. Однією з найбільш ефективних технологій для вирішення проблеми одночасного визначення місцезнаходження дрона та побудови карти навколишнього середовища є **SLAM (Simultaneous Localization and Mapping)**. SLAM дозволяє дрону орієнтуватися в невідомому середовищі, створюючи його карту та визначаючи своє положення на ній у реальному часі [32].

1.3.1 Основні принципи роботи SLAM

SLAM вирішує дві взаємозалежні задачі:

- **Локалізація**: визначення позиції та орієнтації дрона відносно навколишнього середовища;
- **Побудова карти**: створення та оновлення карти навколишнього середовища на основі отриманих даних.

SLAM алгоритми використовують сенсорні дані для побудови карти та визначення місцезнаходження. Вони враховують похибки вимірювань та шум, забезпечуючи точність локалізації [32, 33, 34].

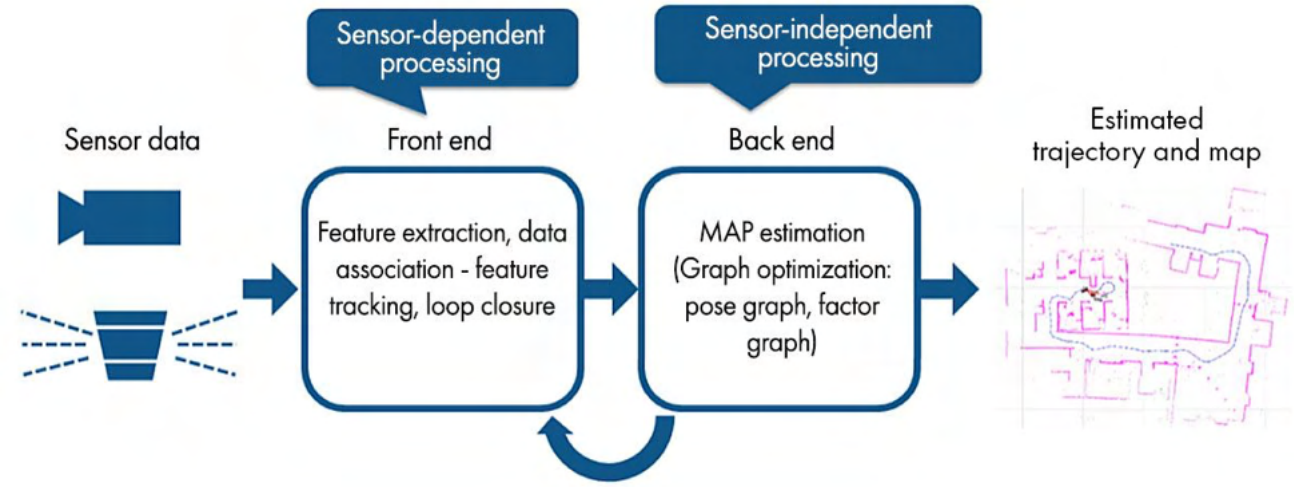


Рисунок 1.10 – Схема роботи алгоритму SLAM [32]

1.3.2 Типи SLAM алгоритмів [32, 33]

Існує кілька типів SLAM алгоритмів, які розрізняються за типом використовуваних сенсорів та методами обробки даних:

1. Візуальний SLAM (Visual SLAM):

Використовує зображення з однієї або кількох камер для побудови карти та локалізації. Візуальний SLAM може бути [35]:

- **Monocular SLAM**: використовує одну камеру. Викликає проблему масштабу, оскільки важко визначити абсолютні відстані;
- **Stereo SLAM**: використовує стереопару камер для отримання глибинної інформації;
- **RGB-D SLAM**: використовує камери, що надають дані про кольори та глибину (наприклад, Microsoft Kinect).

2. Лазерний SLAM (LiDAR SLAM) [36]:

Використовує дані з лазерних далекомірів (LiDAR) для високоточного вимірювання відстаней до об'єктів. LiDAR SLAM забезпечує високу точність, але LiDAR сенсори можуть бути дорогими та енерговитратними.

3. Інерційний SLAM (Inertial SLAM) [37]:

Об'єднує дані з IMU (акселерометри та гіроскопи) для покращення оцінки руху. Часто використовується в комбінації з візуальним SLAM для компенсації швидких рухів та покращення стабільності.

4. Multi-Sensor SLAM:

Поєднує дані з різних сенсорів (камери, LiDAR, IMU, ультразвукові датчики, GPS) для покращення точності та надійності локалізації та побудови карти.

1.3.3 Multi-Sensor SLAM [38]

Multi-Sensor SLAM є особливо актуальним для автономних дронів, оскільки дозволяє компенсувати недоліки одного типу сенсорів за рахунок використання інших. Поєднання даних з різних джерел підвищує точність та стійкість системи в умовах різних зовнішніх факторів.

1.3.4 Переваги Multi-Sensor SLAM:

- **Підвищена точність**: Комбінування даних з різних сенсорів дозволяє зменшити вплив шуму та похибок;
- **Стійкість до відмов сенсорів**: У разі відмови або погіршення роботи одного сенсора система може продовжувати працювати на основі даних з інших сенсорів;
- **Покращена робота в різних умовах**: Наприклад, в умовах поганого освітлення камера може давати неточні дані, але LiDAR або IMU можуть компенсувати цю недостовірність.

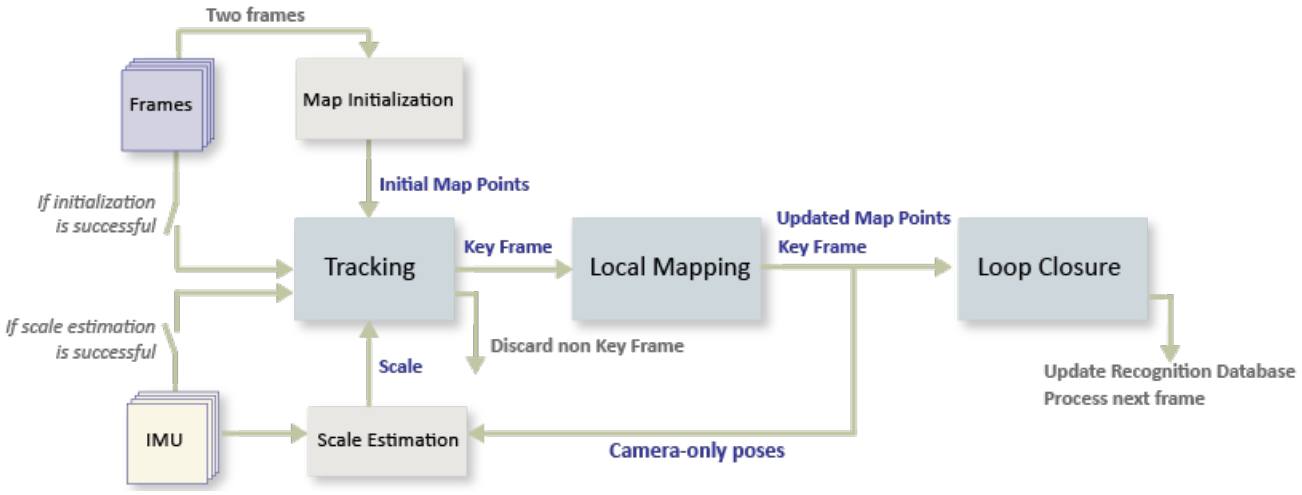


Рисунок 1.11 – Схема Multi-Sensor SLAM з поєднанням даних з камери, LiDAR та IMU [37]

1.3.5 Виклики при реалізації Multi-Sensor SLAM

Хоча Multi-Sensor SLAM має багато переваг, його реалізація супроводжується певними викликами:

- **Синхронізація даних**: Необхідно точно синхронізувати дані з різних сенсорів, що можуть мати різні частоти оновлення та затримки;
- **Калібрування сенсорів**: Важливо правильно калібрувати сенсори, щоб забезпечити коректне поєднання даних;
- **Обчислювальні ресурси**: Поєднання та обробка великої кількості даних вимагає значних обчислювальних ресурсів, що може бути проблемою для дронів з обмеженими можливостями;
- **Розробка алгоритмів поєднання даних**: Необхідно створювати ефективні алгоритми для інтеграції даних з різних сенсорів, враховуючи їх особливості та похибки (зокрема, фільтр Калмана один з них).

1.3.6 Застосування Multi-Sensor SLAM у дронах

Для автономних дронів, які працюють в різних середовищах, Multi-Sensor SLAM є оптимальним рішенням для забезпечення стабільної навігації.

Сенсори сприйняття для Multi-Sensor SLAM:

- **Камери**: для отримання візуальної інформації про оточення;
- **LiDAR**: для точного вимірювання відстаней до об'єктів, особливо в умовах поганої видимості;
- **IMU**: для вимірювання прискорень та кутових швидкостей, допомагає при швидких рухах та динамічних маневрах;
- **Ультразвукові датчики**: для вимірювання висоти над поверхнею землі або виявлення близьких перешкод [39].

Таблиця 1.2 – Переваги та недоліки різних сенсорів у Multi-Sensor SLAM

Сенсор	Переваги	Недоліки
Камера	Висока роздільна здатність, кольорова інформація	Залежність від освітлення, обмежена глибинна інформація
LiDAR	Точні вимірювання відстаней, незалежність від освітлення	Висока вартість, енергоспоживання
IMU	Висока частота оновлення, незалежність від зовнішніх умов	Накопичення похибок з часом (дрейф)
Ультразвукові датчики	Простота, низька вартість	Обмежений діапазон, чутливість до перешкод

1.3.7 Приклади реалізації Multi-Sensor SLAM

Деякі сучасні системи та платформи пропонують реалізацію Multi-Sensor SLAM для автономних дронів:

- **ORB-SLAM2**: Підтримує стерео та RGB-D камери, може бути розширений для використання з IMU [40];
- **Google Cartographer**: Підтримує 2D та 3D SLAM, може працювати з даними LiDAR та IMU [41];
- **ROVIO (Robust Visual Inertial Odometry)**: Поєднує дані з камери та IMU для забезпечення точного позиціонування; EKF-based [42].

Таблиця 1.3 – Порівняння популярних SLAM алгоритмів для дронів [40, 41,42]

Алгоритм	Сенсори	Особливості
ORB-SLAM2	Камери	Висока точність, відкритий код
Google Cartographer	LiDAR, IMU	Підтримка 2D/3D SLAM, розроблений для робототехніки
ROVIO	Камера, IMU	Швидкий, оптимізований для ресурсів

1.3.8 Зв'язок з фільтром Калмана:

Фільтр Калмана відіграє ключову роль у Multi-Sensor SLAM, оскільки він використовується для об'єднання даних з різних сенсорів з урахуванням їх похибок та невизначеностей.

У контексті SLAM фільтр Калмана допомагає:

- **Оцінювати стан дрону:** Прогнозувати та коригувати положення та орієнтацію на основі моделей руху та даних з сенсорів, зокрема додатково використовувати для використання GPS для додаткового позиціонування, якщо доступний;

- **Оновлювати карту середовища:** Коригувати положення орієнтирів та об'єктів на карті на основі нових вимірювань;
- **Інтегрувати дані з різних сенсорів:** Поєднувати відносні вимірювання з камер та LiDAR з абсолютними координатами з GPS та даними з IMU.

Таким чином, фільтр Калмана є одним з фільтрів, що використовується для об’єднання даних з різних сенсорів [43].

1.4. Роль штучного інтелекту в автоматичній навігації: обхід перешкод, адаптивність, навчання з підкріпленням

Штучний інтелект (ШІ) є ключовим елементом у розв’язанні задач автономної навігації дрону. Завдяки ШІ, дрон може приймати складні рішення в реальному часі, забезпечуючи точне позиціювання, обхід перешкод, адаптивність до змінного середовища та здатність до самонавчання [43]. Для покращення локалізації автономного дрону в цій роботі буде реалізовано інтеграцію multi-sensor SLAM, GPS та нейронної мережі з використанням фільтра Калмана. Такий підхід дозволяє об'єднувати дані з різних сенсорів і ШІ-моделей, що значно підвищує точність навігації в умовах обмеженого або нестабільного GPS [45].

1.4.1 Обхід перешкод

Одним із основних завдань ШІ для автономної навігації є виявлення та уникнення перешкод. У рамках цього процесу ШІ обробляє дані з різних сенсорів, таких як камери, LiDAR, IMU, що дозволяє аналізувати навколишнє середовище та планувати безпечні маршрути в реальному часі [45].

ШІ забезпечує такі функції для уникнення перешкод:

- **Аналіз даних з камер і сенсорів**: Дрон за допомогою комп'ютерного зору ідентифікує різноманітні об'єкти на шляху, включно з будівлями, деревами та іншими дронами. Це дозволяє йому створювати карту перешкод у реальному часі;
- **Прогнозування траєкторії рухомих об'єктів**: ШІ прогнозує рух інших об'єктів, таких як транспортні засоби чи люди, дозволяючи дрону передбачати можливі зіткнення та ухилятися від них;
- **Планування альтернативних маршрутів**: При виявленні перешкоди дрон обчислює безпечний маршрут, оптимізуючи його для збереження енергії та часу польоту [46].

1.4.2 Покращення локалізації за допомогою ШІ, SLAM, GPS та фільтра Калмана

Для забезпечення точної навігації використовується інтеграція multi-sensor SLAM, GPS і нейронної мережі, об'єднаних фільтром Калмана. Multi-sensor SLAM дозволяє дрону одночасно створювати карту навколишнього середовища та визначати своє положення в ньому, використовуючи дані з камер, LiDAR та IMU. Однак, такі дані є відносними і можуть накопичувати похибки з часом. GPS, у свою чергу, надає абсолютні координати, але може бути неточним або недоступним у

певних зонах. ШІ, зокрема нейронні мережі, надають можливість розпізнавати об’єкти та орієнтири, які можна використовувати для покращення локалізації у складних умовах. Зокрема, можливе управління дроном за відсутності зв’язку [46].

Нейронна мережа може виконувати кілька важливих функцій:

- **Прогнозування координат**: На основі даних з SLAM та IMU, нейронна мережа може прогнозувати положення дрону, компенсуючи накопичення похибок та знижуючи залежність від GPS;
- **Розпізнавання орієнтирів**: Використовуючи конволюційні нейронні мережі (CNN), дрон розпізнає орієнтири (наприклад, будівлі, дерева або стовпи), що допомагає уточнити своє положення, коли GPS сигнал недоступний;
- **Корекція похибок**: Нейронна мережа, навчена на історичних даних навігації, може виявляти та компенсувати закономірності у похибках SLAM, підвищуючи точність навігації [47, 48].

Фільтр Калмана забезпечує ефективну інтеграцію цих даних, дозволяючи дрону об'єднувати різні джерела для максимально точної оцінки свого положення. Процес виглядає наступним чином:

1. **Прогнозування з нейронної мережі та IMU**: Нейронна мережа та IMU надають початкову оцінку положення дрону. Це дозволяє компенсувати короткочасні маневри та зменшити залежність від GPS;
2. **Оновлення за даними SLAM**: SLAM надає відносні координати, які фільтр Калмана використовує для уточнення положення дрону. При цьому нейронна мережа додає інформацію про орієнтири, що дозволяє коригувати похибки та створювати більш надійну карту;
3. **Корекція за GPS**: Коли GPS сигнал доступний, фільтр Калмана використовує його для періодичної корекції глобального положення дрону, що компенсує накопичення похибок SLAM, у цьому випадку є можливість вимкнення системи штучного інтелекту, що може підвищити енергоефективність системи [49].

1.4.3 Адаптивність та навчання з підкріпленням

Навчання з підкріпленням є одним з основних методів, що дозволяють дрону адаптуватися до нових умов. Дрон накопичує власний досвід і використовує його для підвищення точності ухвалення рішень. Цей підхід дозволяє дрону адаптуватися до нових середовищ, покращуючи свої навички навігації та оптимізуючи використання ресурсів.

За допомогою навчання з підкріпленням дрон вирішує такі задачі:

- **Навчання на власному досвіді**: Дрон накопичує дані про навколишнє середовище, що дозволяє йому самостійно адаптуватися до змінних умов.
- **Оптимізація ухвалення рішень**: Дрон з часом стає здатним краще уникати перешкод, планувати маршрути та економити енергію.
- **Адаптація до нових середовищ**: Навчання з підкріпленням дозволяє дрону залишатися стабільним навіть за умов, що змінюються [50, 51].

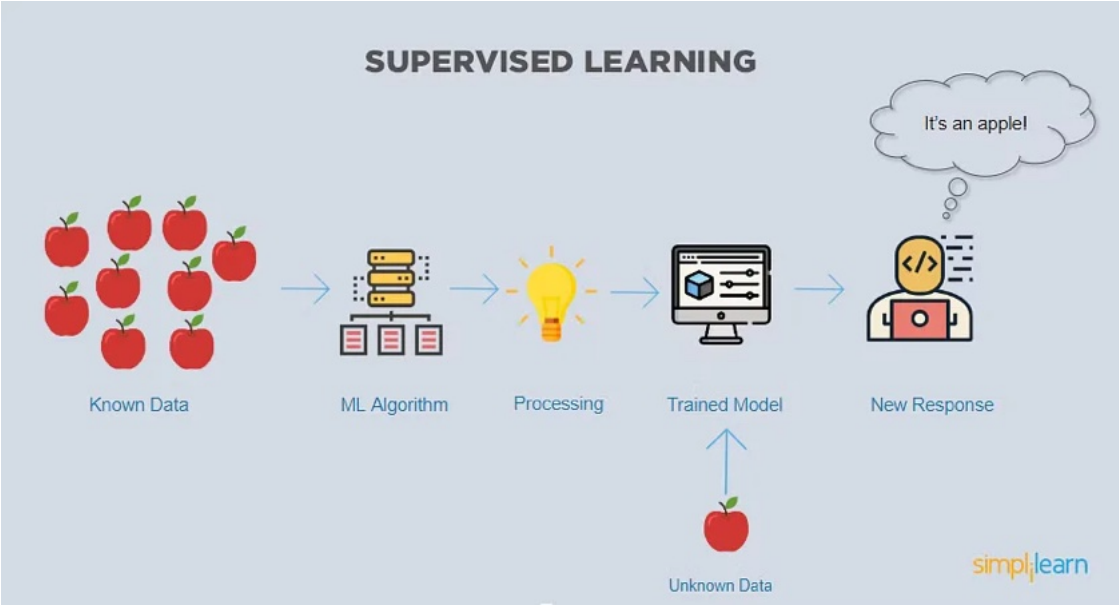


Рисунок 1.12 – Схема процесу навчання з підкріпленням [52]

1.4.4 Розпізнавання орієнтирів та об'єктів

Розпізнавання орієнтирів та об'єктів є важливою складовою автономної навігації дронів, оскільки дозволяє покращити точність локалізації та забезпечити безпечний рух у складних умовах. У рамках цієї роботи планується використання алгоритмів машинного навчання, зокрема конволюційних нейронних мереж (CNN), для виявлення та класифікації об'єктів та орієнтирів на шляху дрону. Завдяки CNN, дрон може:

- **Розпізнавати та класифікувати об'єкти**: Дрон буде здатен ідентифікувати різні об'єкти у навколишньому середовищі (наприклад, дерева, будівлі, транспортні засоби), що може бути корисним для прийняття рішень під час руху. Це допомагає дрону не лише зрозуміти, з чого складається середовище навколо нього, але й аналізувати відносну важливість об'єктів, наприклад, щоб уникати певних ділянок або розпізнавати заборонені зони.
- **Виявляти перешкоди**: Завдяки CNN, дрон зможе виявляти перешкоди, такі як стовпи, гілки дерев або інші об'єкти, які можуть загрожувати безпечному польоту, і автоматично ухилятися від них. Це забезпечує безпеку польоту та оптимальне планування маршруту для мінімізації ризику зіткнень.
- **Використовувати орієнтири для покращення локалізації**: Об'єкти та орієнтири, розташовані у навколишньому середовищі, слугуватимуть точками прив'язки для локалізації, зокрема в умовах, де GPS недоступний або має низьку точність. Це дозволить дрону визначати своє положення з більшою точністю, використовуючи локальні орієнтири на карті. Унікальні орієнтири, такі як великі будівлі чи окремі дерева, допоможуть дрону ідентифікувати своє місцезнаходження на карті та залишатися стабільним навіть у складних умовах.

- **Розпізнавати динамічні та статичні об’єкти**: Використовуючи додатковий шар класифікації, дрон зможе розрізняти об’єкти, що рухаються, і нерухомі об'єкти. Це дозволяє створити комплексну модель навколишнього середовища, враховуючи можливі зміни, які можуть вплинути на обраний маршрут, і оптимізувати ухилення від об'єктів з урахуванням їхньої динаміки [53, 54].

1.4.5 Моделі CNN для об’єктного розпізнавання

Для реалізації розпізнавання об’єктів будуть застосовуватися моделі CNN, такі як YOLO, SSD та Faster R-CNN, які забезпечують різні переваги залежно від швидкості та точності [55].

Таблиця 1.4 – Поширені моделі CNN для розпізнавання об'єктів [55]

Модель	Переваги	Недоліки
YOLO	Висока швидкість, реальний час	Менша точність для дрібних об'єктів
SSD	Баланс між швидкістю і точністю	Складніша реалізація
Faster R-CNN	Висока точність	Більш ресурсомістка, повільніша робота

1.4.6 Виклики та перспективи реалізації інтегрованої системи

Для реалізації штучного інтелекту на базі простих дронів, які не мають можливості використовувати потужні обчислювальні ресурси, слід звернути увагу на мікроконтролери та спеціалізовані модулі, оптимізовані для низького енергоспоживання та компактності. Нижче наведено детальніший опис таких платформ, а точніше деякі з них.

1. STM32 Microcontrollers з підтримкою Edge AI:

STM32 – це серія мікроконтролерів від компанії STMicroelectronics, які широко використовуються в вбудованих системах завдяки їхній енергоефективності та потужності. Нещодавно компанія представила можливості Edge AI на базі STM32, що дозволяє виконувати прості моделі машинного навчання безпосередньо на мікроконтролері.

Переваги:

- **Дуже низьке енергоспоживання:** Оптимізовані для роботи від батареї протягом тривалого часу;
- **Компактність і легкість:** Ідеально підходять для малих дронів;
- **Підтримка базових моделей ШІ:** Можливість виконання задач класифікації та виявлення простих патернів.

Обмеження:

- **Обмежена обчислювальна потужність:** Підходить лише для простих моделей;
- **Потреба в оптимізації моделей:** Моделі повинні бути спеціально адаптовані для роботи на мікроконтролері [56].

2. Arduino Nano 33 BLE Sense:

Arduino Nano 33 BLE Sense – це компактна плата з вбудованими сенсорами та можливістю виконання моделей машинного навчання за допомогою TensorFlow Lite для мікроконтролерів.

Переваги:

- **Вбудовані сенсори:** Акселерометр, гіроскоп, магнітометр, мікрофон та інші, що дозволяє збирати різноманітні дані;
- **Підтримка TensorFlow Lite:** Можливість виконання простих моделей ШІ безпосередньо на платі;
- **Компактні розміри:** Плата має розміри лише 45 x 18 мм.

Обмеження:

- **Обмежена обчислювальна потужність:** Підходить для невеликих моделей і простих задач;
- **Необхідність оптимізації коду:** Потрібна ефективна реалізація алгоритмів для роботи в режимі реального часу [57].

3. Espressif ESP32:

ESP32 – це мікроконтролер з вбудованим Wi-Fi та Bluetooth від компанії Espressif Systems. Він є популярним вибором для вбудованих систем завдяки своїй низькій вартості та потужності.

Переваги:

- **Низьке енергоспоживання:** Різні режими енергозбереження дозволяють оптимізувати споживання;
- **Бездротова комунікація:** Вбудований Wi-Fi та Bluetooth дозволяють передавати дані без додаткових модулів;
- **Підтримка машинного навчання:** Можливість виконання простих моделей за допомогою спеціалізованих бібліотек, таких як MicroML.

Обмеження:

- **Обмежена обчислювальна потужність:** Підходить лише для базових моделей;

- **Потреба в оптимізації алгоритмів:** Для забезпечення роботи в реальному часі [58].

Таблиця 1.5 – Альтернативні апаратні платформи для простих автономних дронів [56, 57, 58]

Платформа	Особливості	Енергоспоживання
STM32 (Edge AI)	Підтримка ML на мікроконтролері, енергоефективність	Дуже низьке
Arduino Nano 33 BLE Sense	Вбудовані сенсори, підтримка TensorFlow Lite	Дуже низьке
Espressif ESP32	Wi-Fi, Bluetooth, базове ML, низька вартість	Дуже низьке

Використання цих платформ дозволяє реалізувати базові функції штучного інтелекту, такі як:

- **Розпізнавання простих образів:** Виявлення об'єктів або патернів за допомогою оптимізованих моделей;
- **Обробка сенсорних даних:** Аналіз даних з акселерометрів, гіроскопів та інших сенсорів для стабілізації та навігації;
- **Прийняття рішень в реальному часі:** Реакція на зміни в навколишньому середовищі без затримок.

Переваги використання легких платформ:

- **Зменшення ваги дрона:** Менша вага платформи дозволяє збільшити час польоту;
- **Економія енергії:** Низьке енергоспоживання продовжує термін роботи від батареї;
- **Простота інтеграції:** Менші розміри та кількість компонентів спрощують конструкцію дрона.

Обмеження та виклики:

- **Обмежена складність моделей ШІ:** Можна виконувати лише прості алгоритми, що може не задовольнити вимоги складних застосувань;
- **Необхідність оптимізації коду:** Розробники повинні ретельно оптимізувати код для ефективної роботи на мікроконтролері;
- **Відсутність потужної підтримки для складних алгоритмів комп'ютерного зору:** Наприклад, виконання повноцінного SLAM на таких платформах є складним.

1.4.7 Покращення локалізації за допомогою об’єктного розпізнавання та нейронних мереж

Розпізнавання орієнтирів і об’єктів допомагає дрону не лише покращити точність навігації, але й адаптуватися до мінливих умов середовища. Ця

можливість стане особливо корисною для виконання складних маневрів у межах міських середовищ, підвищення стабільності локалізації, коли GPS недоступний, та збільшення надійності уникнення перешкод у динамічних середовищах [47].

Висновки за розділом

У першому розділі було розглянуто сучасні технології та методи, які використовуються для забезпечення точного позиціонування та навігації дронів у різних умовах. Розгляд ключових систем позиціонування, таких як GPS, IMU і комп'ютерного зору, дав змогу проаналізувати їхні сильні та слабкі сторони. Зокрема, було відзначено, що GPS забезпечує надійну навігацію на відкритих просторах, але має обмеження у приміщеннях та в міських умовах, тоді як IMU дозволяє контролювати орієнтацію дрона, але потребує корекції через накопичення похибок. Комп'ютерний зір, зокрема технологія SLAM, надає можливості орієнтації у просторі навіть без GPS, використовуючи візуальні дані для створення карти оточення та визначення положення дрону.

Особливу увагу було приділено методам об’єднання даних з різних сенсорів, таких як фільтр Калмана, який забезпечує згладжування шумів і підвищує точність навігації завдяки інтеграції даних від GPS, IMU та камер. Завдяки цьому дрон може стабільно та точно орієнтуватися у складних умовах, навіть за відсутності одного із джерел навігаційних даних.

Також було розглянуто роль комп'ютерного зору та штучного інтелекту для розпізнавання об'єктів і адаптації до мінливих умов середовища. Комп'ютерний зір дозволяє дрону виявляти та уникати перешкод у режимі реального часу, що значно підвищує безпеку польоту. Штучний інтелект забезпечує адаптивне управління польотом, аналізуючи умови та змінюючи параметри польоту для оптимальної продуктивності.

2 ТЕОРЕТИЧНІ ОСНОВИ АВТОМАТИЧНОЇ НАВІГАЦІЇ ДРОНА

2.1 Постановка задачі автоматичної навігації дрона

Автономна навігація дронів сьогодні має велике значення у зв'язку з розвитком технологій та зростанням потреб у безпілотних системах. Основна мета цієї задачі – забезпечення можливості автономного польоту від початкової до цільової точки без втручання людини з мінімізацією можливих помилок, зокрема втрати сигналу GNSS. Такий підхід усуває вплив людського фактору, мінімізує ймовірність помилок та підвищує ефективність дронів.

Для досягнення цієї мети система навігації повинна відповідати низці вимог, що включають точне позиціонування, безпеку польоту, адаптивність до умов навколишнього середовища, надійність і оптимізацію енергоспоживання. Точність позиціонування важлива для виконання завдань у таких сферах, як доставка вантажів чи моніторинг територій, де потрібна висока деталізація місцезнаходження. Безпека польоту забезпечується системами, що допомагають уникати зіткнень із перешкодами, знижуючи ризик аварійних ситуацій. Адаптивність навігаційної системи дозволяє дрону відповідати на зміни у навколишньому середовищі, що особливо корисно при зміні погодних умов або появі нових перешкод [59].

Таблиця 2.1 – Вимоги до автономної навігаційної системи

Вимога	**Опис**
Точність позиціонування	Забезпечення високої точності при визначенні місцезнаходження
Безпека польоту	Виявлення і уникнення перешкод для зниження ризиків зіткнення
Адаптивність	Реакція на змінні умови середовища
Надійність	Стійка робота навіть при втраті або зниженні якості сигналу GNSS/GPS
Енергоефективність	Оптимізація маршруту для зниження витрат енергії

2.2 Огляд методів та технологій для автоматичної навігації

Автоматична навігація дронів включає використання різноманітних технологій і методів для досягнення автономності, точного позиціонування та безпечного управління. Для досягнення цих цілей важливо інтегрувати кілька підходів до позиціонування, обробки даних і адаптивного управління, що дозволить дрону працювати стабільно та безпечно в різних умовах.

2.2.1 Системи позиціонування та їх роль в автономній навігації

Навігаційна система дрона використовує кілька ключових компонентів, таких як GPS, IMU, компас і комп'ютерний зір.

GPS є основним засобом для глобального позиціонування, що забезпечує точність місцезнаходження, але може мати обмеження у міських умовах та закритих просторах. IMU, яка складається з акселерометрів і гіроскопів, забезпечує вимірювання прискорення та кутової швидкості, необхідні для контролю орієнтації дрона. Комп'ютерний зір дозволяє дрону орієнтуватися у просторі навіть без доступу до GPS, використовуючи камери для аналізу об'єктів і орієнтирів.

Таблиця 2.2 – Порівняння технологій для навігації дронів

Технологія	**Особливості**
GPS	Висока точність, обмеження в закритих просторах
IMU	Збір даних про орієнтацію і прискорення
Комп'ютерний зір	Робота за умов відсутності GPS
Програмне забезпечення	Інтеграція даних і реалізація алгоритмів навігації

2.2.2 Усунення людського фактору в управлінні дроном

Автоматична навігація дронів покликана мінімізувати або повністю усунути потребу в постійному контролі з боку оператора, що досягається шляхом автоматизації ключових процесів управління. Це дозволяє дрону приймати рішення в реальному часі та адаптуватися до зміни умов, роблячи його незалежним від втручання людини.

Основні підходи автоматичної навігації включають:

- **Автоматичне планування маршруту**: Автономне планування маршруту є одним з ключових елементів автоматичної навігації. Алгоритми побудови оптимального маршруту дозволяють дрону самостійно прокладати курс між двома точками, обираючи найкоротший і найефективніший шлях з урахуванням перешкод та інших факторів (наприклад, прогнозованих умов навколишнього середовища);
- **Система уникнення перешкод**: Дрони, оснащені сенсорами, такими як камери, LiDAR або ультразвукові датчики, здатні виявляти об'єкти на своєму шляху та автоматично ухилятися від них. Такі системи працюють у реальному часі, що дозволяє уникати зіткнень із раптовими перешкодами. Для забезпечення надійного уникнення перешкод використовуються алгоритми машинного навчання, зокрема нейронні мережі, які допомагають дрону розпізнавати та класифікувати об'єкти на шляху;
- **Адаптивне управління**: Завдяки машинному навчанню дрони можуть адаптувати свої алгоритми управління до поточних умов. Наприклад, під час вітряної погоди алгоритм стабілізації польоту автоматично коригується, щоб підтримувати стабільність дрона, мінімізуючи вплив вітру на його траєкторію. Подібна адаптивність дозволяє дрону залишатися стійким у мінливих умовах, а також виконувати складні маневри без потреби в зовнішньому контролі [60, 61].

2.2.3 Роль програмного забезпечення в автоматичній навігації

Програмне забезпечення забезпечує обробку даних, отриманих із сенсорів, і реалізацію алгоритмів навігації. Програмні алгоритми, такі як Dijkstra або A*, дозволяють дрону обирати оптимальний маршрут, інтегруючи дані з камер, GPS та IMU. Це створює комплексну систему управління, що дозволяє дрону здійснювати навігацію за умов мінливих факторів навколишнього середовища [61, 62].

Таблиця 2.3 – Ключові функції програмного забезпечення в проєкті [61, 62]

Компонент	**Функція**	**Опис**
Обробка даних сенсорів	Згладжування та фільтрація даних	Використання фільтра Калмана для зменшення шумів
Навігаційні алгоритми	Розрахунок маршруту та корекція траєкторії	Уникнення перешкод, оптимізація маршруту
Інтеграція компонентів	Узгодженість роботи системи	Синхронізація всіх модулів для безперебійної роботи
Система уникнення перешкод	Виявлення та обхід перешкод	Використання комп'ютерного зору та алгоритмів обробки даних
Адаптивне управління	Зміна параметрів залежно від умов	Корекція польоту під впливом зовнішніх факторів

2.3 Вибір програмних засобів та обґрунтування вибору Python

Ефективне програмне забезпечення для автономної навігації дронів базується на мові програмування, яка забезпечує оптимальну швидкість розробки, гнучкість і широкий набір інструментів.

2.3.1 Аналіз доступних програмних платформ та фреймворків

У сучасних навігаційних системах для дронів використовуються мови програмування, такі як C++, Java та Python, кожна з яких має свої особливості. C++ забезпечує високу продуктивність та оптимізацію пам'яті, що ідеально підходить для систем реального часу. Однак її складний синтаксис і відносно тривалий час розробки можуть ускладнювати створення проєктів, орієнтованих на швидкий розвиток. Java, завдяки віртуальній машині JVM, надає переваги у кросплатформеності та стабільності, хоча її продуктивність у реальному часі може бути нижчою, ніж у C++ та Python.

Python виділяється своєю простотою та гнучкістю, що дозволяє швидко реалізовувати прототипи і проводити тестування алгоритмів. Крім того, вона підтримує великий набір бібліотек для машинного навчання, комп'ютерного зору, обробки сенсорних даних, що робить її ідеальним вибором для розробки автономних систем навігації [62, 63, 64].

Таблиця 2.4 – Порівняння мов програмування для програмування системи дрону та алгоритмів

Мова	Переваги	Недоліки
C++	Висока продуктивність, контроль пам'яті	Складний синтаксис, вимагає більше часу
Java	Кросплатформеність, стабільність	Нижча продуктивність у реальному часі
Python	Простота синтаксису, широкий набір бібліотек, швидка розробка	Нижча продуктивність для обчислювально інтенсивних завдань

До найважливіших фреймворків для автономної навігації належать:

- **ROS (Robot Operating System)**: Фреймворк для задач робототехніки, що підтримує C++ і Python. Він дозволяє ефективно керувати сенсорами, об'єднувати різні системи навігації та інтегруватися з симуляторами [65];
- **OpenCV**: Бібліотека з відкритим кодом для реалізації комп'ютерного зору. Завдяки ній можна створювати алгоритми для розпізнавання об'єктів, слідування за маркерами та аналізу простору, що важливо для орієнтації дрона у просторі [66];
- **TensorFlow та PyTorch**: Фреймворки для машинного навчання, які надають готові моделі та інструменти для створення та тренування нейронних мереж, зокрема для задач розпізнавання об'єктів [67, 68].

2.3.2 Переваги використання Python для розробки систем автоматичної навігації

Python обрано як основну мову програмування для розробки навігаційної системи дрону завдяки її простоті, потужному набору бібліотек та широкій підтримці серед розробників. Бібліотеки на зразок NumPy та SciPy забезпечують обробку великих обсягів даних з сенсорів, а OpenCV дозволяє працювати з комп'ютерним зором та розпізнаванням об'єктів.

Простота синтаксису Python спрощує процес розробки та дозволяє швидко тестувати нові алгоритми, що особливо важливо при роботі з динамічними даними від сенсорів дрона. Python має багато готових рішень та активну підтримки зі сторони спільноти, що значно прискорює процес розробки. Інтеграція з апаратними платформами також не викликає труднощів, оскільки Python підтримує бібліотеки, такі як PySerial та pyusb, які забезпечують зручну взаємодію з сенсорами і пристроями, зокрема з камерами, IMU, GPS та LiDAR.

Рисунок 2.1 – Використання Python для сенсорних даних дрона та комп'ютерного зору

2.3.3 Огляд бібліотек та інструментів на Python

Python пропонує широкий вибір бібліотек, необхідних для роботи з системами автоматичної навігації:

- **NumPy та SciPy**: Бібліотеки для наукових та математичних розрахунків, які підтримують роботу з багатовимірними масивами та матрицями. Це особливо важливо для обробки даних з сенсорів та для побудови алгоритмів фільтрації;
- **OpenCV**: Потужна бібліотека для роботи з комп'ютерним зором. Вона дозволяє розпізнавати об'єкти та орієнтири, здійснювати обробку зображень і відео в реальному часі, що значно підвищує ефективність навігації дрона [66];
- **TensorFlow**: Цей фреймворк машинного навчання використовується для побудови та тренування нейронних мереж, необхідних для розпізнавання об'єктів та аналізу даних. Вони забезпечують інструменти для роботи з великими наборами даних, а також надають можливість використання попередньо навчених моделей [67];
- **Matplotlib**: Бібліотека для візуалізації даних, яка допомагає аналізувати результати навігаційних алгоритмів, відображати дані про траєкторії руху дрона та сенсорні показники, що є важливим аспектом під час налагодження та оптимізації системи [69].

Таблиця 2.5 – Узагальнення наведених бібліотек Python

Бібліотека	**Призначення**
NumPy, SciPy	Наукові та математичні розрахунки
OpenCV	Обробка зображень, комп'ютерний зір
TensorFlow, Keras	Машинне навчання, нейронні мережі
Matplotlib	Візуалізація даних

Завдяки широкому вибору доступних інструментів та зручності роботи, Python забезпечує ефективне середовище для розробки складних систем навігації та дозволяє впроваджувати інноваційні рішення з використанням комп'ютерного зору і штучного інтелекту, що є ключовими елементами в автоматичній навігації дронів.

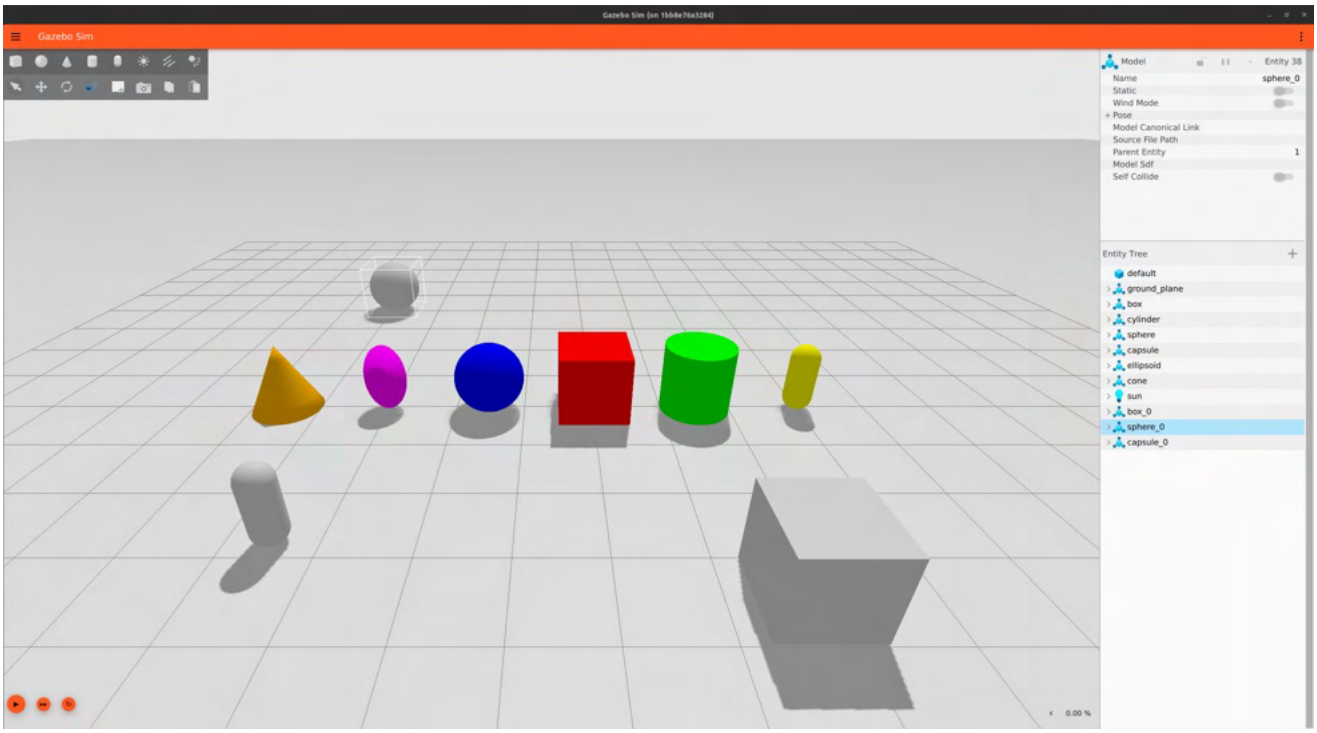
2.4 Інструменти для симуляції та тестування

Симуляція та тестування – ключові етапи для вдосконалення автоматичної навігації дронів. Симуляція та тестування є критично важливим моментом для забезпечення безпеки та надійності автономної навігаційної системи дронів. Завдяки симуляторам та спеціалізованим інструментам, можна перевірити різноманітні сценарії, оцінити ефективність алгоритмів, оптимізувати налаштування системи та знизити ризик можливих помилок перед випробуванням у реальних умовах [70].

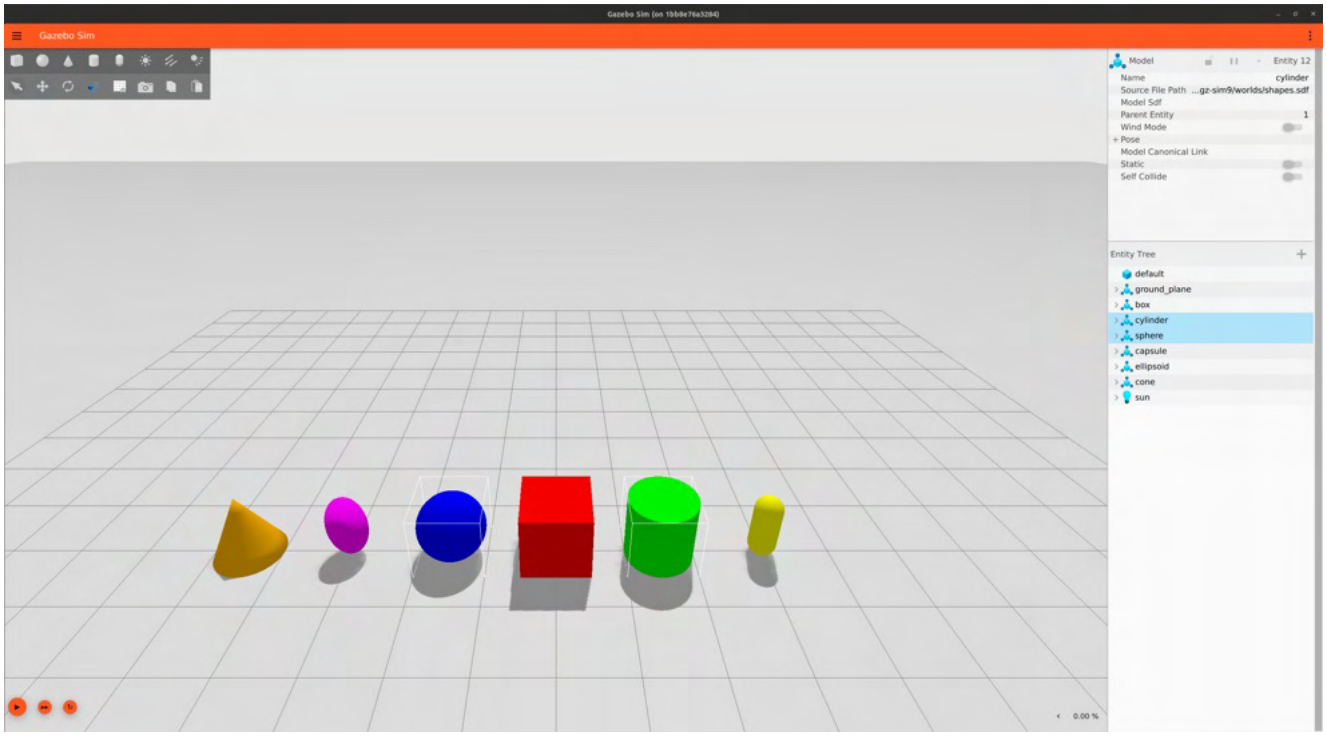
2.4.1 Огляд симуляторів для безпілотних літальних апаратів

Симулятори є основним інструментом для тестування поведінки дронів в умовах, наближених до реальних. Серед поширених симуляторів виділяються:

- **Gazebo**: популярний симулятор для робототехніки, що підтримує складне 3D-моделювання і працює у зв'язці з ROS (Robot Operating System). Gazebo дозволяє створювати складні сцени та інтегрувати різноманітні сенсори, що забезпечує деталізовану перевірку систем навігації дронів;
- **AirSim**: симулятор від Microsoft, який надає реалістичні фізичні моделі, високу деталізацію середовища та підтримує сценарії для тестування дронів;
- **RotorS**: платформа для симуляції мультикоптерів, що активно використовується разом з ROS. Вона забезпечує точне моделювання динаміки дронів і дозволяє налаштовувати параметри польоту;
- **FlightGear**: авіаційний симулятор з відкритим кодом, який використовується для базових симуляцій польоту дронів і інших літальних апаратів [70, 71].



а



б

Рисунок 2.2 – Приклад інтерфейсу симулятора Gazebo для навігації дрона [72]

2.4.2 Вибір симулятора та його інтеграція з Python

Gazebo був обраний як основний симулятор для тестування алгоритмів навігації та автономного управління дроном завдяки його інтеграції з ROS та розширеними можливостями для налаштування сенсорів і середовища. Платформа Gazebo підтримує Python API, що дозволяє контролювати дрон, збирати дані із сенсорів та перевіряти навігаційні алгоритми в реальному часі [73].

2.4.3 ROS – Robot Operating System

ROS, або Robot Operating System, є відкритою операційною системою для робототехніки, яка забезпечує набір інструментів і бібліотек, необхідних для розробки роботизованих систем, зокрема, безпілотних літальних апаратів. ROS полегшує процес інтеграції апаратних компонентів дронів, таких як камери, IMU, GPS, та обробку отриманих даних у реальному часі [74].

Основні особливості ROS:

- **Модульна архітектура**: ROS дозволяє створювати додатки з незалежних модулів (нодів), що обмінюються даними через повідомлення, забезпечуючи високу гнучкість;
- **Інструменти для сенсорних даних**: забезпечує підтримку стандартних сенсорів (камери, LiDAR, IMU) та засоби для обробки інформації з них;
- **Вбудовані алгоритми**: ROS має багатий набір бібліотек для управління рухом, SLAM, обробки зображень і комп'ютерного зору;
- **Сумісність з симуляторами**: ROS інтегрується з різними симуляторами, такими як Gazebo, що дозволяє тестувати алгоритми віртуально, перш ніж запускати їх на реальних дронах [74].

Таблиця 2.6 – Основні компоненти ROS для безпілотних апаратів [74]

Компонент	Опис
Ноди (Nodes)	Незалежні процеси, що виконують конкретні завдання, наприклад, управління дроном або обробка даних
Повідомлення (Messages)	Система передачі даних між нодами через певні канали
Теми (Topics)	Канали, якими ноди обмінюються повідомленнями
Сервіси (Services)	Запити і відповіді, що використовуються для комунікації між нодами
Дії (Actions)	Більш складні команди з можливістю відстежувати прогрес виконання

2.4.4 Методологія симуляції та тестування автоматичної навігації

Процес симуляції і тестування в середовищі Gazebo і ROS складається з таких основних етапів:

1. **Визначення критеріїв успішності**: Основні показники, за якими оцінюється якість роботи системи – це точність позиціонування, стабільність польоту та ефективність обльоту перешкод;
2. **Створення тестових сценаріїв**: Включає симуляцію різних середовищ, що відображають реальні умови, у яких дрон може опинитися, таких як міські каньйони, закриті простори та об'єкти з високою перешкоджувальною здатністю;
3. **Аналіз результатів**: Зібрані дані аналізуються для виявлення недоліків і вдосконалення алгоритмів навігації та уникнення перешкод.

ROS та Gazebo забезпечують комплексну систему для відлагодження, що дозволяє перевіряти працездатність системи та забезпечувати її належну роботу перед запуском у реальних умовах.

2.5 Усунення людського фактору та автоматизація управління

Розробка систем автоматичної навігації має на меті мінімізувати або повністю усунути необхідність втручання людини, що дозволяє зменшити помилки, пов’язані з людським фактором, та забезпечити стабільну, надійну та безпечну роботу дрону навіть в умовах складних середовищ. Основними аспектами такої автоматизації є ефективні алгоритми управління, обробка даних від сенсорів, використання зворотного зв'язку та адаптація до динамічно змінних умов.

2.5.1 Принципи автоматичного управління без втручання людини

Автоматичне управління вимагає створення замкнених систем керування, де рішення приймаються на основі постійного моніторингу сенсорних даних та моделі оточення. У такій системі управління використовуються алгоритми, що забезпечують стабільність дрону та оптимізують його рух. Основою цього підходу є використання зворотного зв'язку, завдяки якому дрон може коригувати свої дії на основі поточних умов, наприклад, для стабілізації висоти, збереження напрямку чи уникнення перешкод.

Алгоритми контролю, такі як PID-регулятори (пропорційно-інтегрально-диференціальні контролери), виконують важливу функцію підтримки стабільності польоту. PID-регулятор коригує швидкість двигунів дрону для забезпечення заданого положення, компенсуючи вплив зовнішніх факторів, як-от вітер або турбулентність. Окрім того, автоматичне прийняття рішень дозволяє дрону самостійно змінювати свій маршрут чи швидкість у відповідь на зміну умов середовища, забезпечуючи адаптивність до нових ситуацій.

PID-контролер (Пропорційно-Інтегрально-Диференційний контролер) – це автоматичний регулятор, який використовується для керування різними процесами. Він широко застосовується у промислових системах для підтримки бажаних значень таких параметрів, як температура, швидкість, тиск і положення. Його назва походить від трьох компонентів, які визначають його роботу:

1. **Пропорційний (P) компонент** – цей компонент реагує пропорційно на поточну похибку (різницю між бажаним і фактичним значенням параметра). Чим більша похибка, тим сильніший вихід контролера;
2. **Інтегральний (I) компонент** – реагує на накопичену суму похибок за певний час. Він усуває залишкові помилки, які можуть залишатися від пропорційного компонента, поступово компенсуючи систематичні відхилення;
3. **Диференційний (D) компонент** – відповідає за швидкість зміни похибки, реагуючи на тенденцію зміни параметра. Це допомагає зменшити коливання і стабілізувати систему, особливо під час швидких змін.

У підсумку, PID-контролер коригує вихід системи таким чином, щоб зменшити похибку до мінімуму. Він особливо корисний у випадках, де потрібно

точне та стабільне регулювання, наприклад, у дронах, де PID-контролери можуть використовуватися для підтримки стабільності і керування рухом [75, 76].

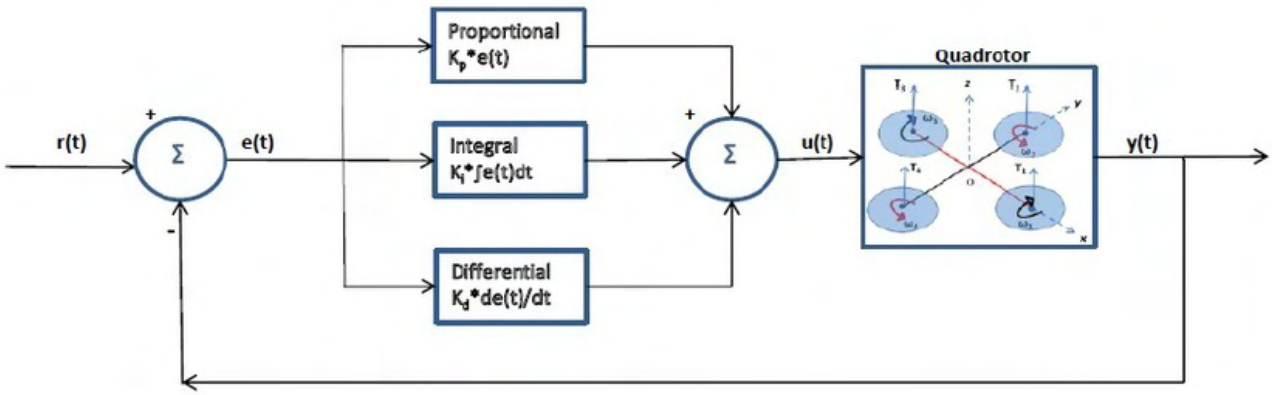


Рисунок 2.3 – Схема управління дроном із замкненим зворотним зв'язком [77]

2.5.2 Технології та методи для автономної роботи дрона

Для досягнення повної автономності в роботі дрону використовуються методи автоматичного планування маршруту, уникнення перешкод та адаптації до нових умов. Під час планування маршруту дрон самостійно визначає оптимальний шлях, враховуючи такі фактори, як енергоефективність, швидкість та наявність перешкод. Використання алгоритмів пошуку, наприклад, алгоритму Дейкстри або алгоритмів на основі A*, дозволяє дрону знайти найкоротший шлях до цілі, при цьому враховуючи обмеження по енергоспоживанню.

Уникнення перешкод – це ключова функція автономної навігації, що дозволяє дрону безпечно переміщатися у складних середовищах. Сучасні дрони можуть використовувати технології комп'ютерного зору та обробки даних від сенсорів, як-от LiDAR чи ультразвукові датчики, для визначення об'єктів на шляху. Завдяки цим технологіям дрон виявляє перешкоди заздалегідь та обирає оптимальний маршрут для обльоту об'єктів. Окрім цього, обробка непередбачуваних ситуацій дозволяє дрону реагувати на раптові зміни в оточенні, як-от поява іншого об'єкта на шляху, зміна погодних умов або збої в роботі сенсорів.

2.5.3 Використання машинного навчання для підвищення автономності

Для подальшого зниження необхідності втручання людини в процес управління дронами використовуються методи машинного навчання, зокрема навчання з підкріпленням, що дозволяє дрону навчатися на власному досвіді. У процесі навчання з підкріпленням дрон набуває навичок, що дозволяють йому приймати оптимальні рішення в різних ситуаціях. Наприклад, дрон може навчитися уникати перешкод, оптимізувати маршрут для економії енергії або адаптувати свою поведінку до змін в оточенні.

Використання конволюційних нейронних мереж (CNN) для розпізнавання об'єктів дозволяє дрону ефективно визначати перешкоди та орієнтири на шляху,

що є критично важливим для точного позиціонування та ухилення від небезпечних зон. Завдяки CNN дрон може ідентифікувати об'єкти, такі як дерева, будівлі чи інші транспортні засоби, що дозволяє йому коригувати свій маршрут залежно від розташування цих об'єктів. Адаптивні системи машинного навчання також сприяють підвищенню автономності, дозволяючи дрону самостійно налаштовувати параметри польоту відповідно до умов, що змінюються.

Таблиця 2.7 – Основні методи машинного навчання для автоматизації управління дроном

Метод машинного навчання	**Застосування**
Навчання з підкріпленням	Оптимізація маршрутів, уникнення перешкод
Конволюційні нейронні мережі	Розпізнавання об'єктів та орієнтирів
Адаптивні системи	Адаптація до умов навколишнього середовища

Ці методи машинного навчання забезпечують гнучкість та адаптивність навігаційної системи дрона, знижуючи залежність від попередньо встановлених правил і дозволяючи дрону приймати рішення в умовах невизначеності

Висновки за розділом

У цьому розділі детально розглянуто основні теоретичні засади автоматичної навігації дронів, що дозволяють досягти повної автономності без необхідності втручання людини. Обговорено ключові вимоги до систем навігації, серед яких точне позиціонування, адаптивність до змінних умов, безпека польоту та енергоефективність. Аналіз сучасних технологій автоматичної навігації підкреслив важливість інтеграції різних сенсорних систем, таких як GPS, інерціальні вимірювальні одиниці (IMU) та системи комп'ютерного зору, для забезпечення надійного функціонування дронів у різних умовах експлуатації.

Автоматизація управління реалізується через замкнені системи контролю, що працюють на основі алгоритмів обробки сенсорних даних та технологій машинного навчання. Використання PID-регуляторів та адаптивних алгоритмів, зокрема навчання з підкріпленням, сприяє точному контролю польоту та уникненню перешкод, підвищуючи автономність дронів і їх здатність реагувати на зовнішні фактори. Завдяки конволюційним нейронним мережам (CNN), дрони можуть ідентифікувати об'єкти на своєму шляху, аналізувати складні візуальні сцени та оптимізувати траєкторію польоту.

Розгляд програмних засобів показав, що мова програмування Python, разом із широким набором бібліотек, таких як OpenCV для комп'ютерного зору та TensorFlow для машинного навчання, є ефективним інструментом для розробки систем автоматичної навігації. Використання цих інструментів спрощує процес розробки та дозволяє швидко впроваджувати нові алгоритми. Крім того,

інструменти для симуляції, такі як Gazebo у поєднанні з Robot Operating System (ROS), дають можливість тестувати алгоритми в умовах, наближених до реальних, що мінімізує ризики при переході до практичного використання та підвищує надійність системи.

Підсумовуючи, поєднання сучасних сенсорних технологій, алгоритмів управління та програмних засобів створює міцну основу для розробки автономних дронів, здатних виконувати складні завдання в різноманітних умовах експлуатації.

3 РОЗРОБКА КОМПЛЕКСНОЇ СИСТЕМИ АВТОНОМНОЇ НАВІГАЦІЇ ДРОНУ

3.1. Архітектура системи: компоненти і взаємозв’язок. Архітектура системи: компоненти і взаємозв’язок.

Архітектура системи автономної навігації дрона є комплексним поєднанням апаратних і програмних компонентів, які забезпечують безперервне сприйняття середовища, аналіз отриманих даних, планування маршруту та реалізацію руху. Основні етапи роботи системи включають сканування, інтерпретацію даних, планування та виконання. У цьому розділі описується структура системи, перелік основних компонентів та моделей і їх взаємозв’язок.

Система починає свою роботу з фізичного середовища, яке включає всі процеси, що потрібно відстежувати. Дрон сприймає дані про навколишнє середовище за допомогою низки сенсорів, які забезпечують отримання інформації про статичні та динамічні об’єкти, перешкоди та параметри середовища. Дані передаються до модуля обробки, який інтерпретує інформацію, визначає важливі об’єкти, локалізує дрон і створює модель навколишнього середовища. На основі цієї моделі здійснюється планування руху, що включає побудову оптимального маршруту. Завершальний етап – виконання маршруту з урахуванням оновлених даних у реальному часі (Рисунок 3.1).

Рисунок 3.1 – Кроки до автономної навігації дрону

3.1.1 Основні сенсори і їх взаємозв’язок

Система автономної навігації дрона використовує широкий набір сенсорів, які працюють разом для забезпечення точності та стабільності. До них належать:

1. **IMU (інерційна вимірювальна одиниця):** Визначає прискорення і кутову швидкість, забезпечуючи дані для орієнтації та базового переміщення. Використовується у всіх фазах руху, але може накопичувати похибки;
2. **Lidar або стереокамери (RGBD):** Вимірюють відстань до об’єктів, забезпечують деталізовану карту середовища. Їх перевагою є висока точність у статичних умовах, а недоліком – висока вартість та чутливість до погодних умов;
3. **Барометр:** Забезпечує вимірювання висоти, що особливо важливо у складних топографічних умовах;
4. **Сенсори оптичного потоку:** Дозволяють оцінювати рух дрона відносно поверхонь, забезпечуючи корекцію маршруту у відсутності GPS;
5. **Камери (зокрема з підтримкою Yolov5):** Використовуються для визначення важливих об’єктів та класифікації динамічних і статичних перешкод;
6. **Вітрові сенсори:** Можуть бути корисними для корекції траєкторії у вітряних умовах.

Таблиця 3.1 ілюструє основні переваги та виклики використання цих сенсорів.

Таблиця 3.1 – Основні переваги та виклики використання наведених сенсорів

Сенсор	Переваги	Недоліки
IMU	Простота, надійність, незалежність від зовнішніх умов	Накопичення похибок, потребує корекції даних
Lidar	Висока точність, незалежність від освітлення	Висока вартість, обмеження в складних погодних умовах
Стереокамери (RGBD)	Деталізована глибина, можливість класифікації об’єктів	Потребує високої обчислювальної потужності
Барометр	Точне визначення висоти	Чутливість до змін тиску
Камери (Yolov5)	Універсальність, можливість обробки великого обсягу даних	Залежність від освітлення, складність налаштування
Сенсори оптичного потоку	Компактність, ефективність в умовах відсутності GPS	Може мати похибки на нерівних поверхнях
Вітрові сенсори	Визначення напрямку і швидкості вітру	Залежність від точності калібрування, обмеженість функціоналу

3.1.2 Вибір сенсорів

Вибір сенсорів для дрону є важливим етапом проєктування системи, адже саме вони забезпечують збір критично важливих даних для навігації, аналізу середовища та виконання завдань. Основними факторами, які впливають на вибір сенсорів, є:

1. **Дальність польоту з огляду на заряд:** Сенсори, що встановлюються на дрон, споживають енергію, тому їх ефективність має бути оптимальною. Вибір сенсорів з низьким енергоспоживанням, таких як IMU чи оптичний потік, дозволяє зберігати заряд батареї, забезпечуючи триваліший час польоту. У разі використання енергоємних сенсорів, як-от Lidar, потрібно враховувати вплив їхньої роботи на загальну тривалість польоту та розглядати можливість підвищення ємності батареї або оптимізації роботи інших компонентів дрону.

2. **Бажана точність:** Точність вимірювань залежить від конкретних завдань дрону. Наприклад:

- Для задач високоточної локалізації та побудови карти використовуються сенсори з високою роздільною здатністю, такі як Lidar або стереокамери (RGBD). Вони забезпечують детальну модель середовища, що необхідно для точного позиціонування;
- Для стабільності польоту в умовах відсутності GPS (GNS-Denied) використовуються IMU та сенсори оптичного потоку, які дозволяють забезпечити базову орієнтацію;
- Для визначення висоти можуть застосовуватися барометри, які є енергоефективними, хоча і менш точними порівняно з лазерними або ультразвуковими датчиками.

3. **Вартість дрону:** Вибір сенсорів також визначається економічною доцільністю, особливо якщо система має бути масово виробленою. Наприклад:

- Використання дорогих сенсорів, таких як Lidar, може значно підвищити вартість дрону, тому їх застосування обмежується ситуаціями, де критично необхідна висока точність;
- Оптичний потік або стереокамери пропонують компроміс між вартістю та функціональністю, забезпечуючи прийнятну точність при помірній ціні;
- IMU є стандартним компонентом, доступним за відносно низькою ціною, і його використання є майже обов’язковим у будь-якому дроні [78].

3.1.3 Баланс між характеристиками

Оптимальний вибір сенсорів базується на компромісі між дальністю польоту, точністю та вартістю. Наприклад, для завдань моніторингу у відкритих просторах дрон може використовувати базові сенсори з акцентом на енергоефективність, тоді як для складних умов, таких як міські середовища або закриті приміщення, знадобляться високоточні сенсори.

Цей баланс дозволяє створювати дрони, які відповідають конкретним потребам користувача, забезпечуючи необхідну функціональність за прийнятну вартість та збереження ресурсу батареї.

Рисунок 3.2 – Можливі компоненти надійної навігаційної системи дрону

3.1.4 Загальна схема роботи системи

Алгоритм роботи дрона побудований за принципом безперервного циклу, що включає наступні етапи (Рисунок 3.3):

1. **Сканування середовища:** Сенсори збирають інформацію про навколишнє середовище. Lidar або камери створюють модель простору, а IMU забезпечує стабільність польоту;
2. **Інтерпретація даних:** Отримана інформація обробляється для визначення статичних і динамічних перешкод. Камери з підтримкою Yolov5 аналізують об’єкти, визначаючи їх значущість. SLAM (Simultaneous Localization and Mapping) інтегрує дані для створення карти;
3. **Планування маршруту:** На основі побудованої карти алгоритм A* планує найкоротший шлях до цілі, враховуючи перешкоди та обмеження;
4. **Виконання руху:** Контролери забезпечують виконання траєкторії, стабілізацію польоту та корекцію маршруту у реальному часі [60, 79].

Реалізація цих етапів забезпечує точну та надійну автономну навігацію навіть у складних умовах. SLAM дозволяє будувати карту в реальному часі, а алгоритм A* оперативно коригує маршрут, знижуючи ризик зіткнень, таким чином наближаючи його до нуля [60, 79].

3.1.5 Динамічна перебудова маршруту

У процесі польоту система може виявляти нові перешкоди, які не були враховані під час початкового планування. У таких випадках дані про нові об’єкти

інтегруються в SLAM, після чого алгоритм A* перебудовує маршрут, забезпечуючи безпеку руху.

Така архітектура дозволяє дрону адаптуватися до змін середовища, що робить його надзвичайно ефективним для виконання складних завдань у реальному часі.

Рисунок 3.3 – Схема роботи системи

3.2. Використання даних з IMU, компаса та сенсорів оптичного потоку для підтримання орієнтації та приблизної траєкторії

Система автономного управління дронами вимагає високоточної орієнтації та ефективного визначення траєкторії польоту. Для цього використовуються такі сенсори, як інерційні вимірювальні одиниці (IMU), магнітні компаси та сенсори оптичного потоку. Кожен із цих сенсорів має свої переваги, недоліки та особливості роботи, які впливають на загальну продуктивність системи.

3.2.1 IMU (інерційні вимірювальні одиниці)

IMU включає акселерометри та гіроскопи, які вимірюють прискорення та кутову швидкість дрону. Дані з IMU дозволяють визначати орієнтацію дрону, швидкість руху та його положення.

Принцип роботи IMU (Рисунок 3.4):

- **Акселерометр** вимірює лінійне прискорення уздовж осей X, Y, Z.
- **Гіроскоп** визначає кутові швидкості, що дозволяє розрахувати нахили дрону.
- Дані інтегруються, щоб оцінити траєкторію.

Типова похибка IMU:

- **Дрейф:** помилка накопичується через постійну інтеграцію даних.
- **Залежність від температури:** зовнішні умови можуть впливати на точність показів.

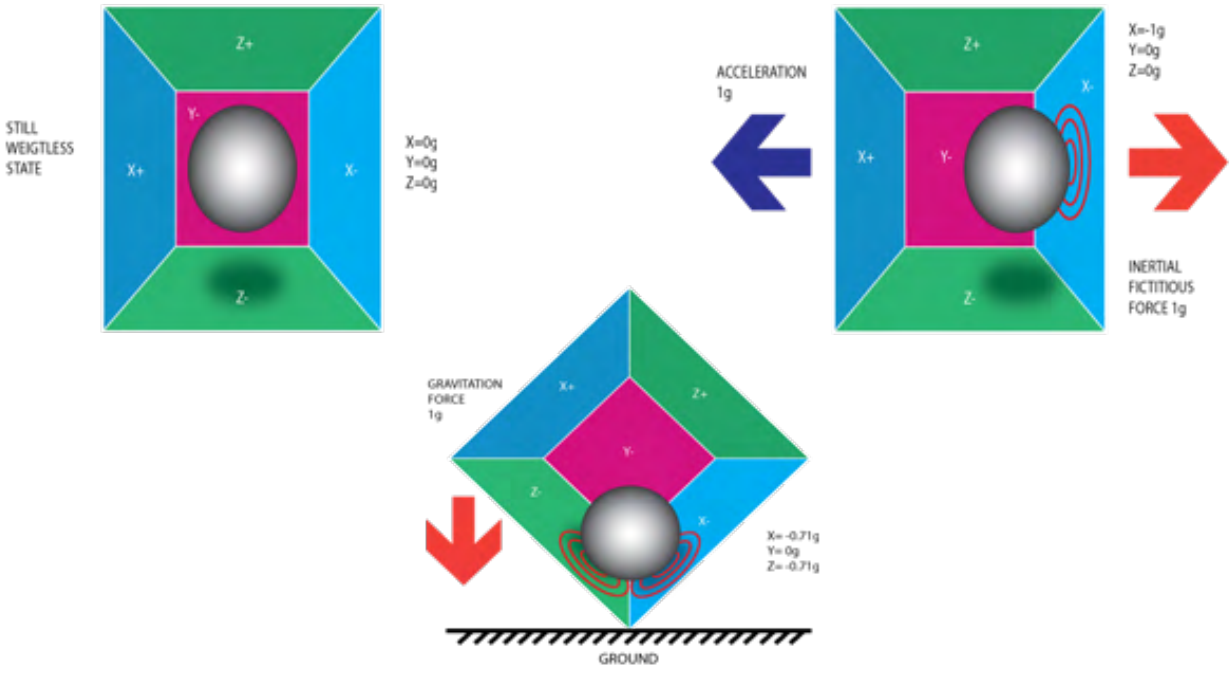


Рисунок 3.4 – Принцип роботи IMU та обчислення орієнтації [80]

Таблиця 3.2 – Основні характеристики IMU

Параметр	**Значення**
Дані	Лінійне прискорення, кутова швидкість
Переваги	Швидка реакція, незалежність від зовнішніх умов
Недоліки	Накопичення похибок, чутливість до вібрацій

3.2.2 Компас

Компас використовується для визначення орієнтації дрону відносно магнітного поля Землі. Це надає важливу інформацію для стабілізації польоту.

Принцип роботи (Рисунок 3.5):

- Орієнтація визначається за допомогою вимірювання магнітних векторів;
- Використовується для стабільного напрямку польоту.

Похибки компаса:

- **Магнітні завади:** електромагнітні поля та об’єкти можуть спотворювати показники;
- **Девіація:** відхилення через локальні аномалії магнітного поля.

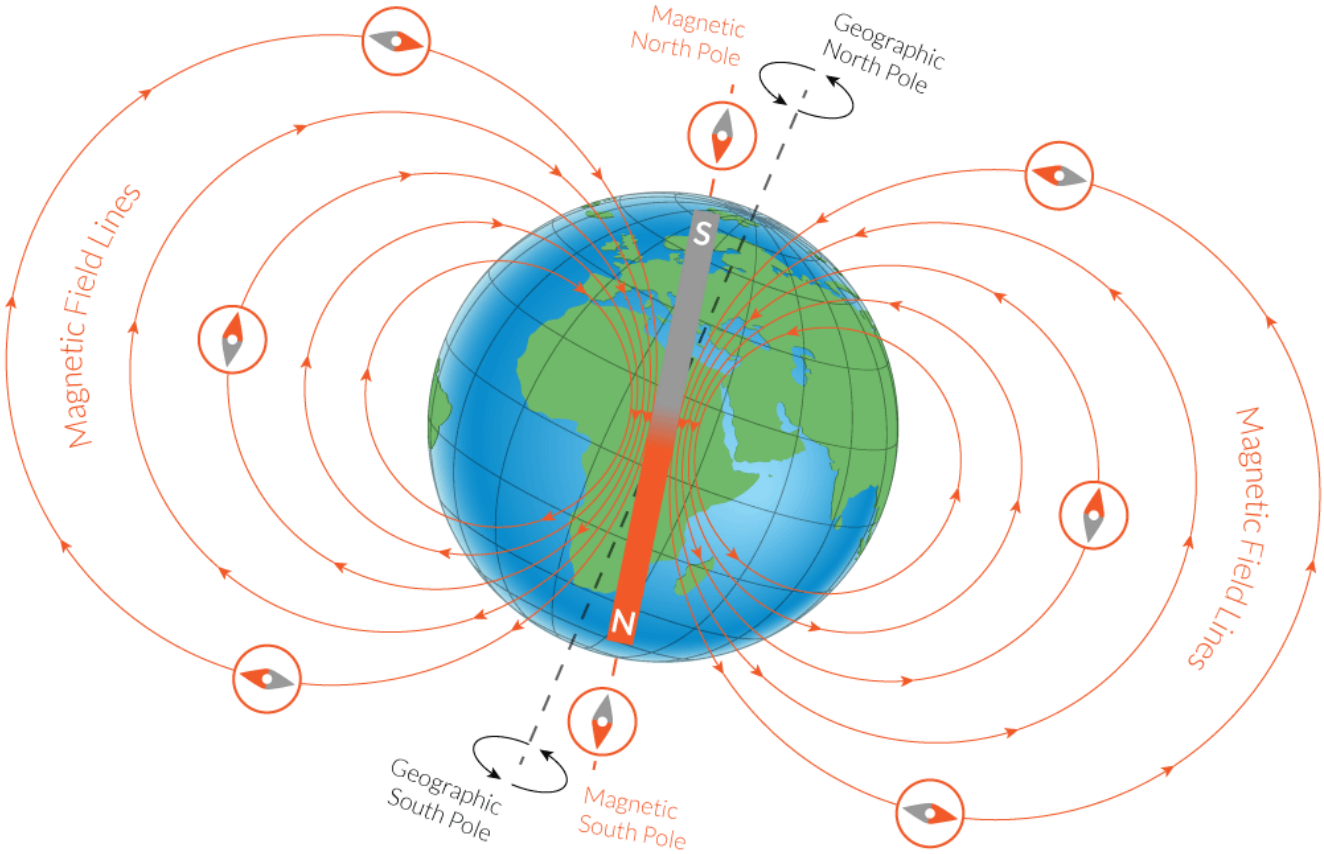
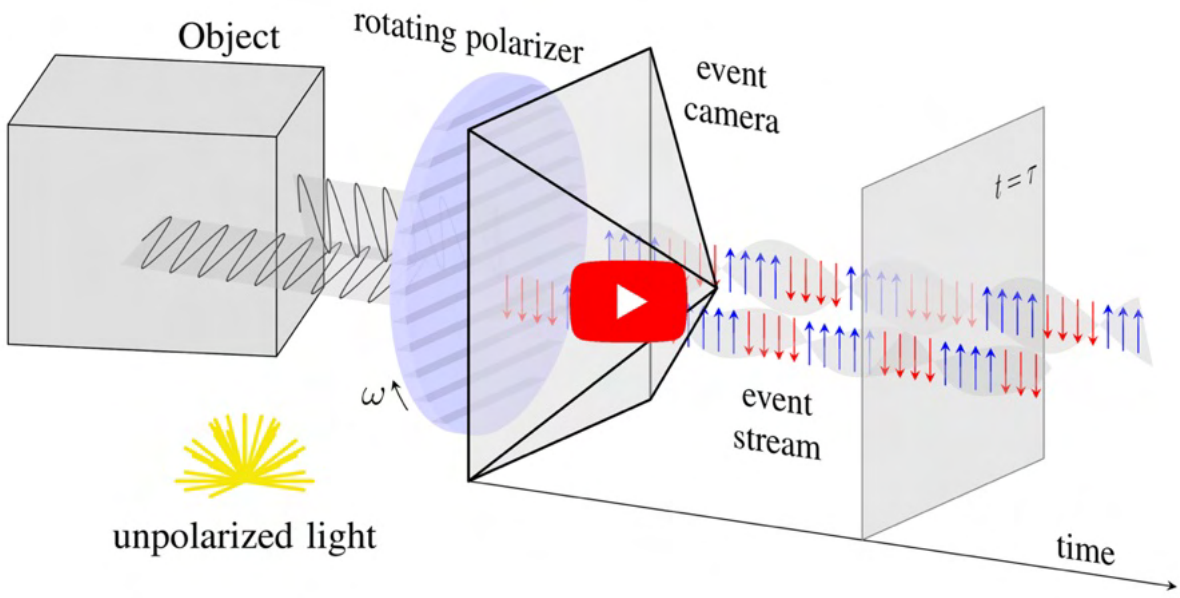


Рисунок 3.5 – Визначення орієнтації компасом відносно магнітного поля [81]

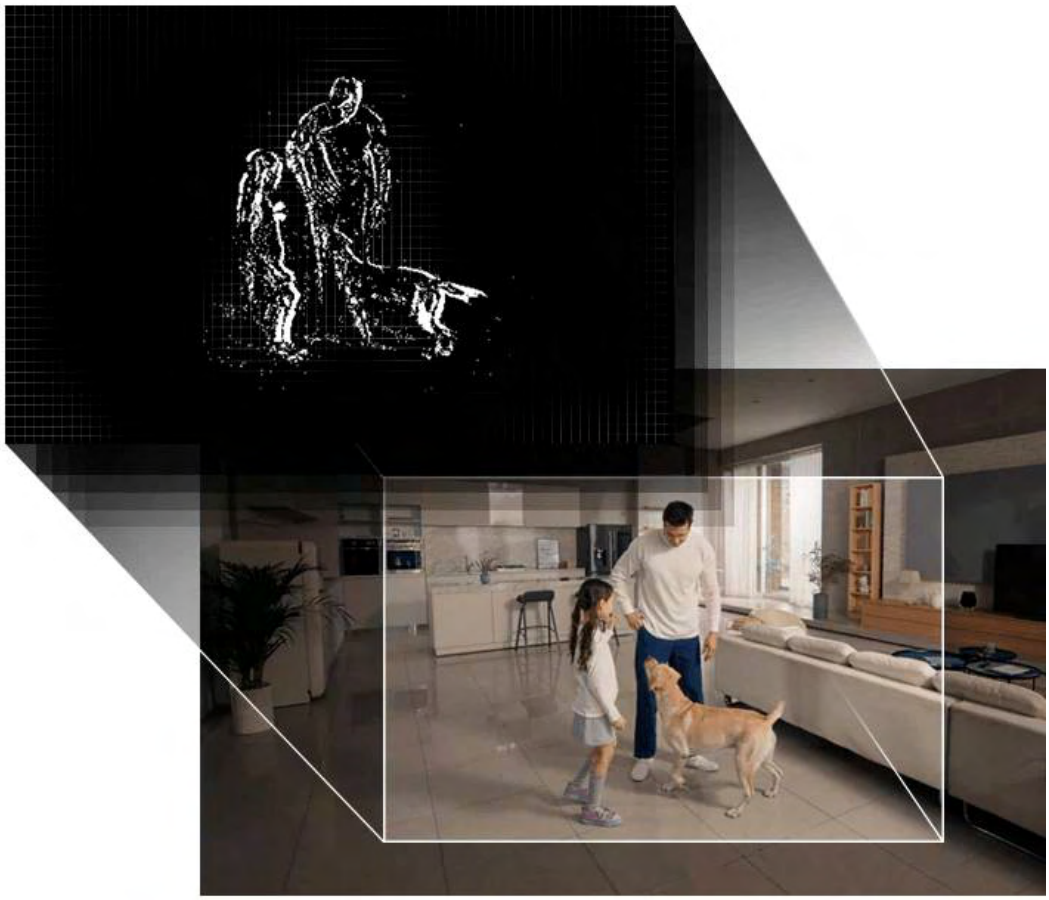
3.2.3 Сенсори оптичного потоку

Сенсори оптичного потоку використовуються для визначення відносного руху дрону. Вони аналізують зміщення пікселів між послідовними кадрами, щоб обчислити швидкість і напрямок руху.

Можлива реалізація на основі камер, а також камери подій (Event Cameras), що відловлюють зміни в інтенсивності потоку світла (Рисунок 3.6).



а [82]



б [83]

Рисунок 3.6 – Схема роботи сенсора оптичного потоку

Особливості:

- Працюють без GPS, що особливо важливо в умовах його відсутності;
- Забезпечують високу точність у середовищах з текстурованою поверхнею.

Похибки сенсорів оптичного потоку:

- **Освітлення:** недостатнє або надмірне освітлення може впливати на якість аналізу кадрів;
- **Однорідна текстура:** складно визначити рух у середовищах без чітких орієнтирів.

3.2.4 Інтеграція даних

Для забезпечення точності системи дані з усіх сенсорів інтегруються за допомогою алгоритмів, таких як фільтр Калмана. Це дозволяє компенсувати похибки кожного сенсора окремо.

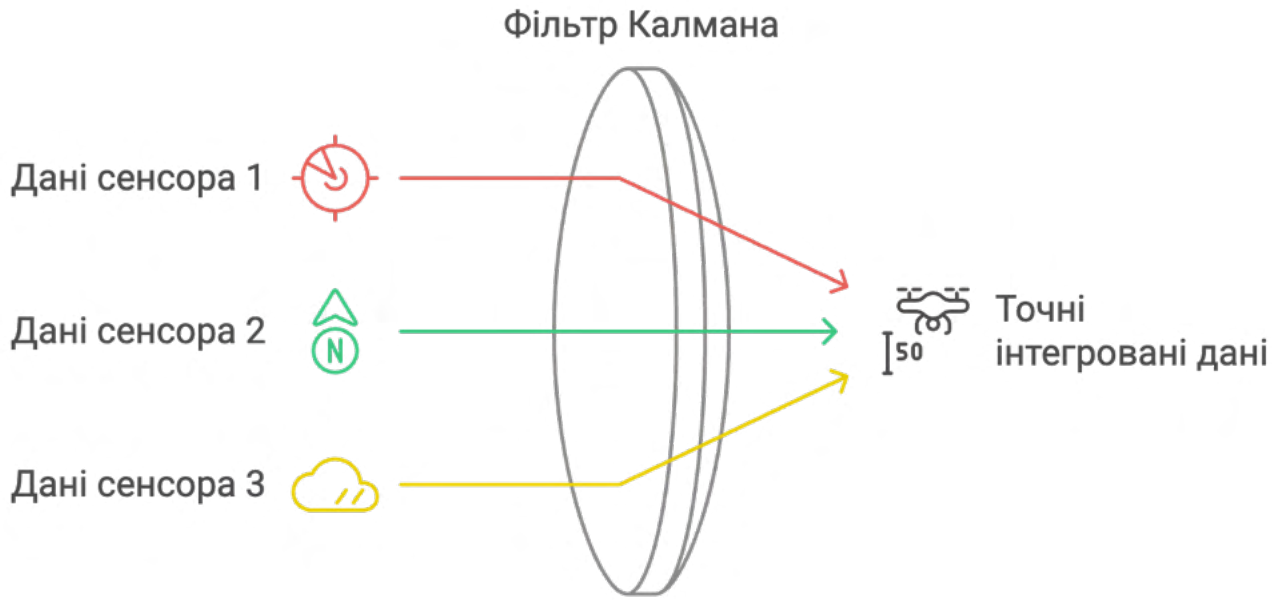


Рисунок 3.7 – Система інтеграції даних з сенсорів

3.2.5 Порівняння сенсорів

Таблиця 3.3 – Порівняння характеристик сенсорів

Сенсор	Дані	Переваги	Недоліки
IMU	Прискорення, кутова швидкість	Робота в будь-якому середовищі, швидка реакція	Накопичення похибок, вплив вібрацій

Сенсор	Дані	Переваги	Недоліки
Компас	Напрямок на північ	Простота, низька енергозатратність	Магнітні завади, девіація
Сенсори оптичного потоку	Відносна швидкість, зміщення	Відсутність залежності від GPS	Залежність від освітлення та текстури

3.3. Реалізація фільтра Калмана для згладжування та корекції даних з датчиків в реальному часі

Фільтр Калмана є одним із найпоширеніших алгоритмів для роботи з даними, отриманими від сенсорів у реальному часі. Його головна мета – згладжування шумів, корекція вимірювань та отримання більш точних оцінок стану системи, наприклад, положення чи орієнтації дрона. Для нелінійних систем, таких як навігаційні системи дронів, ефективним інструментом є розширений фільтр Калмана (EKF). Цей алгоритм адаптовано для роботи з нелінійними моделями через використання процесу лінеаризації.

Робота EKF відбувається у два основні етапи: прогнозування та оновлення. На етапі прогнозування обчислюється стан системи та її похибки на основі попередніх даних і моделей динаміки. Для цього нелінійні моделі замінюються лінійними за допомогою матриці Якобіана, яка описує взаємозв’язок між змінними. На етапі оновлення використовуються вимірювання з сенсорів, таких як IMU, компас, оптичний потік або барометр, для корекції прогнозованих даних. Це дозволяє отримати більш точні оцінки, враховуючи як похибки вимірювань, так і моделі системи.

Розширений фільтр Калмана обробляє дані з різних сенсорів: IMU забезпечує інформацію про прискорення та кутову швидкість, компас дає орієнтацію відносно магнітного поля Землі, а оптичний потік надає дані про зміщення у просторі. Барометр використовується для оцінки висоти польоту. Об’єднання цих даних дозволяє не лише знизити вплив шумів, а й компенсувати похибки, характерні для окремих сенсорів.

Проте лінеаризація, яка є основою роботи EKF, має свої обмеження. Вона призводить до помилок апроксимації, які можуть накопичуватись у високонелінійних системах. Це ускладнює точне оцінювання стану, особливо у складних умовах роботи. Також EKF чутливий до початкових умов, і неправильна ініціалізація може значно знизити точність результатів.

Для порівняння, нерозширений фільтр Калмана (UKF) не потребує лінеаризації. Він використовує сигма-точки для моделювання розподілу станів, що дозволяє краще працювати з нелінійними системами. Проте UKF вимагає більших обчислювальних ресурсів, що робить його менш придатним для пристроїв із обмеженими потужностями, таких як дрони.

Незважаючи на існуючі обмеження, EKF залишається ефективним інструментом для автономної навігації. Він дозволяє інтегрувати дані з кількох сенсорів, забезпечуючи високу точність оцінювання стану.

Повний алгоритм Фільтру Калману виглядає наступним чином (Рисунок 3.8).

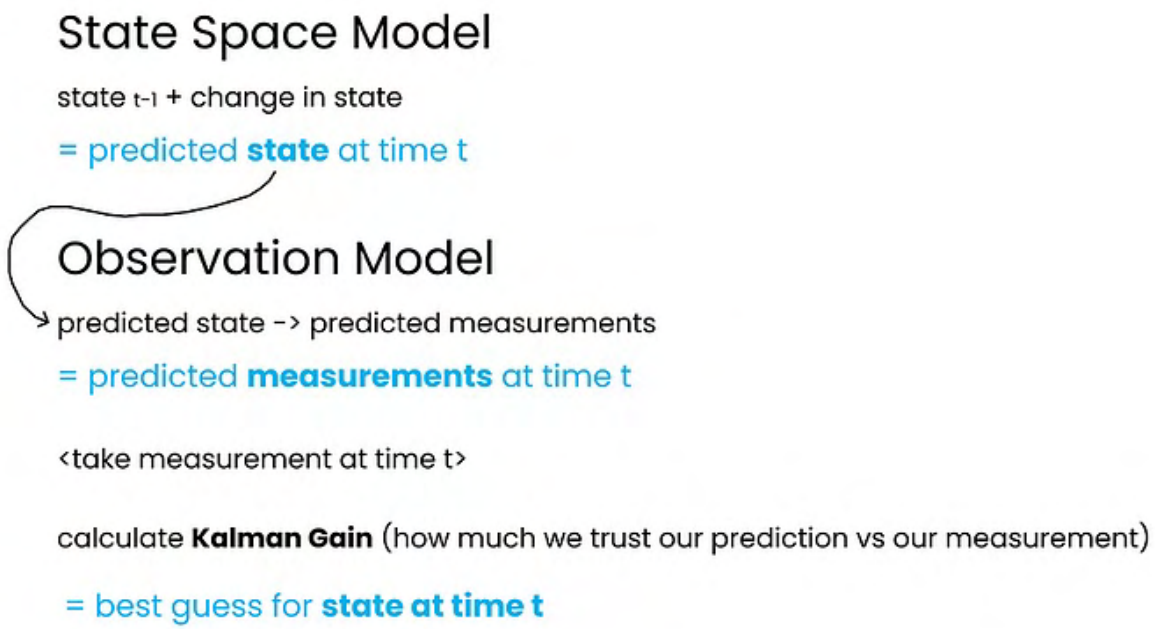


Рисунок 3.8 – Алгоритм фільтру Калмана [28]

3.3.1 Порівняння математичної складової EKF та KF

Математичний вигляд фільтру Калмана було детально розглянуто у Розділі 1, Пункті 1.2. Однак, ключова різниця між класичним фільтром Калмана (KF) та розширеним фільтром Калмана (EKF) полягає у їх здатності працювати з різними типами систем. KF призначений для роботи з лінійними системами, тоді як EKF дозволяє враховувати нелінійності, що є критично важливим для багатьох реальних застосувань, зокрема, для автономної навігації дронів.

KF працює з лінійними рівняннями стану та спостереження, які описуються у вигляді матриць:

$$x_k = Ax_{k-1} + Bu_k + w_k$$
$$z_k = Hx_k + v_k,$$

де A – матриця переходу стану, B – матриця керування, u_k – вектор керування, w_k – шум процесу, H – матриця вимірювань, v_k – шум вимірювань.

EKF, на відміну від KF, працює з нелінійними моделями:

$$x_k = f(x_{k-1}, u_k) + w_k$$
$$z_k = h(x_k) + v_k,$$

де $f(x)$ – нелінійна функція динаміки системи, $h(x)$ – нелінійна функція спостережень.

Зокрема, під функцією $f(x_{k-1}, u_k)$ прихована та сама лінійна функція, різниця лише у матрицях переходу станів та матриці керування, де вони представлені, як матриці Якобіана, тобто матриці часткових похідних:

$$A_{t-1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_{t-1}} & \frac{\partial f_1}{\partial y_{t-1}} & \frac{\partial f_1}{\partial \gamma_{t-1}} \\ \frac{\partial f_2}{\partial x_{t-1}} & \frac{\partial f_2}{\partial y_{t-1}} & \frac{\partial f_2}{\partial \gamma_{t-1}} \\ \frac{\partial f_3}{\partial x_{t-1}} & \frac{\partial f_3}{\partial y_{t-1}} & \frac{\partial f_3}{\partial \gamma_{t-1}} \end{bmatrix}$$

а

$$B_{t-1} = \begin{bmatrix} \frac{\partial f_1}{\partial v_{t-1}} & \frac{\partial f_1}{\omega_{t-1}} \\ \frac{\partial f_2}{\partial v_{t-1}} & \frac{\partial f_2}{\omega_{t-1}} \\ \frac{\partial f_3}{\partial v_{t-1}} & \frac{\partial f_3}{\omega_{t-1}} \end{bmatrix}$$

б

Рисунок 3.9 – Матриці нелінійної системи [28]

А також оновлення стану відбувається наступним чином

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_{k|k-1})),$$

де $h(x)$ – матриця спостереження, або прогнозоване вимірювання на основі нелінійної моделі, у лінійній системі вона зводиться до Hx. Вона визначає, як вектор стану (x_k) трансформується для отримання прогнозованого вимірювання (z_k). Тобто, яким чином лінеаризувати вектор станів для отримання прогнозованого вимірювання.

Математичним способом шляхом розкладу на ряд Тейлора дана матриця спостереження задається, як:

$$h(x_k) = h\left(x_k^f\right) + J_h(x_k^f)(x_k - x_k^f) \text{ [84]},$$

де $J_h(x)$ – матриця Якобіана.

Інакше кажучи, ми апроксимуємо нелінійні значення у їх лінійне представлення на основі попереднього значення та зміни, що задана матрицею Якобіана [18-28].

3.3.2 Матриця Якобіана: Пояснення та приклади

Матриця Якобіана – це математичний інструмент, який використовується для лінеаризації нелінійних функцій у задачах, таких як розширений фільтр Калмана (EKF). Вона складається з часткових похідних багатовимірної функції і описує, як змінюється кожна компонента функції щодо кожної змінної вектору вхідних даних. У контексті EKF матриця Якобіана дозволяє представити нелінійну залежність між станом системи і вимірюваннями в лінійній формі, необхідній для розрахунків фільтру.

Якщо ми маємо векторну функцію $h(x)$, де:

$$f(x) = \begin{bmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_m(x_1, x_2, \dots, x_n) \end{bmatrix},$$

то матриця Якобіана J цієї функції визначається як [85]:

$$J = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

3.3.3 Реалізація фільтру Калмана

Для ілюстрації роботи фільтра Калмана розглянемо просту одновимірну систему. Ця система описує стан об’єкта, який рухається прямолінійно з певною швидкістю, під впливом шуму. Фільтр Калмана буде використовуватися для оцінки поточного стану об’єкта, враховуючи дані з вимірювань, які містять похибки.

Для даної системи задамо наступні параметри:

3.3.4 Постановка задачі:

Фільтр Калмана використовує математичну модель станів для опису змінних системи (наприклад, положення, швидкості, прискорення). У нашому випадку стан системи описується положенням x та прискоренням a.

1. Модель станів

Зміна положення x системи залежить від початкового положення, прискорення, і часу t:

$$x = x_0 + \frac{1}{2}at^2$$

Оскільки швидкість v прямо залежить від прискорення, можемо виразити її як:

$$v = v_0 + at$$

Ці рівняння описують динаміку системи. Таким чином, матриця переходу станів моделі:

$$A = \begin{bmatrix} 1 & 0.5t^2 \\ 0 & 1 \end{bmatrix}$$

Виходячи з наведених рівнянь, матриця переходу станів A визначає, як кожна змінна стану залежить від попереднього стану:

$$\begin{bmatrix} x \\ a \end{bmatrix} = \begin{bmatrix} 1 & 0.5t^2 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ a \end{bmatrix}$$

Управління не задається (u відслідковуватися не буде):

$$B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Матриця спостереження:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

GPS – вимірювання положення, з шумом $\sigma_{GPS}^2 = 9$.

IMU – вимірювання прискорення, з шумом $\sigma_{GPS}^2 = 0.01^2$.

Таким чином, шуми та параметри системи будуть встановлені наступні:

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$R = \begin{bmatrix} 9 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

Початкові значення:

$$x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Коваріація стану:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

У цьому підрозділі представлено лише математичний опис моделі та частина коду. Дослідження даної тестової системи буде наведена у наступному розділі, де здійснюватиметься його тестування на основі симульованих даних від сенсорів IMU та GPS.

3.4. Застосування Multi-sensor EKF SLAM для побудови карти оточення і визначення локальної позиції

3.4.1 Загальний огляд EKF SLAM

Розширений фільтр Калмана для одночасної локалізації та побудови карти (EKF SLAM) є одним із найпоширеніших алгоритмів для автономної навігації. Його мета полягає у вирішенні двох основних задач: визначення позиції дрона у просторі та побудови карти оточення на основі спостережень орієнтирів. Цей алгоритм забезпечує точну локалізацію навіть у середовищах із обмеженим використанням GPS, інтегруючи дані з різноманітних сенсорів, таких як IMU, Lidar або стереокамери.

EKF SLAM працює на основі розширення стандартного фільтра Калмана для нелінійних систем, враховуючи, що більшість моделей руху та вимірювання є нелінійними.

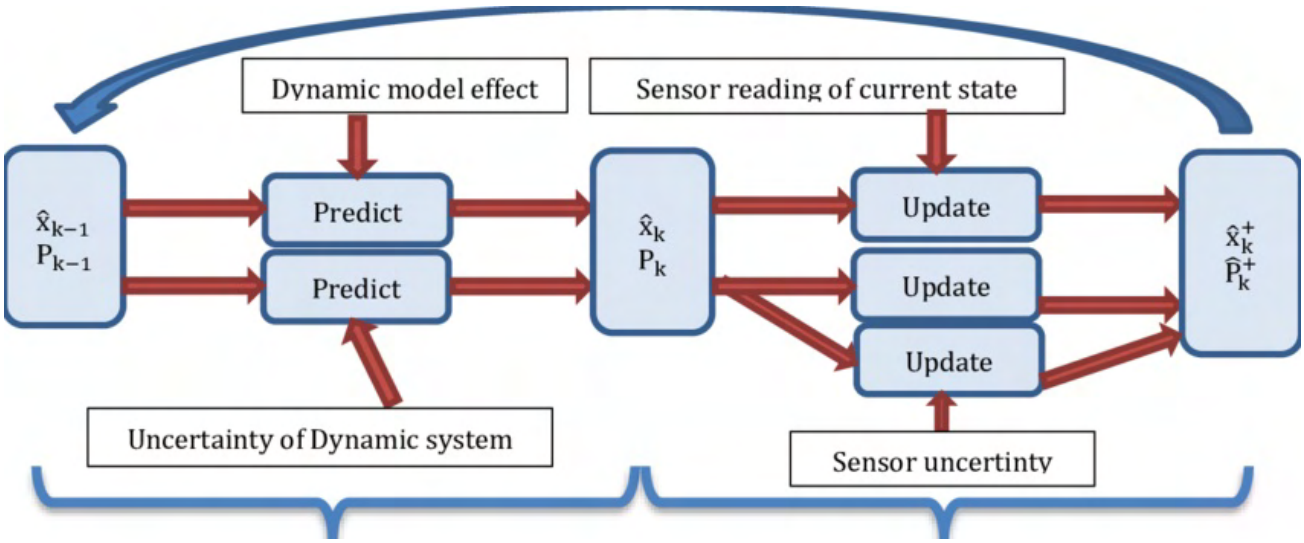


Рисунок 3.9 – Алгоритм роботи EKF SLAM майже ідентичний до алгоритму EKF [86]

3.4.2 Постановка задачі

EKF SLAM включає два основні об'єкти:

1. **Локалізація дрона**: визначення його положення (x, y) та орієнтації (θ) у просторі;
2. **Побудова карти оточення**: визначення положення орієнтирів (маяків) у глобальній системі координат.

Вектор стану системи (μ) складається з положення дрона та позицій орієнтирів:

$$\mu = \begin{bmatrix} x \\ y \\ \theta \\ l_{x1} \\ l_{y1} \\ \vdots \\ l_{xn} \\ l_{yn} \end{bmatrix}$$

Матриця коваріації $(\sum)$ описує невизначеність як стану дрона, так і орієнтирів, а також кореляцію між їхніми станами:

$$P = \begin{bmatrix} P_{rr} & P_{rl} \\ P_{lr} & P_{ll} \end{bmatrix}$$

де:
– P_{rr} – невизначеність стану дрона;
– P_{ll} – невизначеність позицій орієнтирів;
– P_{rl}, P_{lr} – кореляція між невизначеністю дрона та орієнтирів.

Та відповідні коваріаційні матриці представлені наступним чином:

Початкова невизначеність стану дрона:

$$P_{rr} = \begin{bmatrix} \sigma_{x^2} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_{y^2} & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_{\theta^2} \end{bmatrix}$$

Початкова невизначеність позицій орієнтирів:

$$P_{ll} = \begin{bmatrix} \sigma_{x^2} & 0 \\ 0 & \sigma_{x^2} \end{bmatrix}$$

Початкова кореляція між невизначеністю дрона та орієнтирів:

$$P_{rl} = P_{lr}^{T} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

На початковому етапі роботи алгоритму EKF SLAM матриця коваріації P ініціалізується так, щоб крос-кореляція між роботом і орієнтирами дорівнювала нулю. Це означає, що робот і орієнтири не мають початкової кореляції. Такий підхід є природним, оскільки позиція орієнтирів на момент їх ініціалізації невідома, і невизначеність їхнього розташування встановлюється дуже високою. З часом, у процесі роботи алгоритму, кореляція між роботом і орієнтирами виникає.

Кореляція з’являється під час роботи алгоритму через два основних етапи: передбачення та оновлення вимірювань. На етапі передбачення, коли робот здійснює рух, його позиція та відповідна невизначеність оновлюються відповідно до моделі руху. Якщо орієнтири вже включені до моделі, то їх невизначеність починає взаємодіяти з невизначеністю позиції робота через крос-кореляцію в матриці . На етапі оновлення вимірювань, спостереження орієнтирів, таких як дані з Lidar, камери або GPS, дозволяють уточнювати їхню позицію. При цьому коригується і позиція робота, адже ці величини взаємопов’язані через матрицю коваріації. У результаті утворюється крос-кореляція між роботом і орієнтиром.

Кореляція має вирішальне значення, адже вона дозволяє використовувати орієнтири для уточнення позиції робота. Наприклад, якщо позиція орієнтира стає більш точною, це прямо впливає на покращення оцінки позиції робота. Крім того, кореляція забезпечує узгодженість даних між роботом і орієнтирами, гарантує, що будь-яка зміна позиції робота впливатиме на всі пов'язані орієнтири. У випадку, якщо крос-кореляція залишатиметься нульовою, орієнтири не впливатимуть на оцінку позиції робота, а дані про орієнтири будуть враховуватись ізольовано. Така модель не враховуватиме реальні фізичні взаємозв’язки між роботом і орієнтирами, що суттєво знижує точність SLAM.

Таким чином, кореляція між роботом і орієнтирами, яка виникає в процесі роботи алгоритму, є ключовою для високої точності локалізації та побудови карти середовища.

3.4.3 Етапи роботи алгоритму EKF SLAM

1. Передбачення стану

Передбачення нового стану (μ_t) базується на моделі руху дрона. Для системи з лінійним рухом:

$$x_{t+1} = x_t + vcos(\theta_t)\Delta t;$$
$$y_{t+1} = y_t + vsin(\theta_t)\Delta t;$$
$$\theta_{t+1} = \theta_t + \omega\Delta t.$$

Орієнтири залишаються незмінними на цьому етапі.

Матриця коваріації передбачається так само, як і у фільтрі Калмана:

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q$$

2. Оновлення вимірювань

На цьому етапі використовується інформація від сенсорів, які спостерігають орієнтири. Для кожного орієнтира вимірювання (z_i) включають:

$$r_i = \sqrt{(l_{xi} - x)^2 + \left(l_{yi} - y\right)^2}$$
$$\phi = \arctan 2\left(l_{yi} - y, l_{xi} - x\right) - \theta$$

Оновлення стану та ковариації відбувається у фільтрі EKF Калмана.

3. Ініціалізація нових орієнтирів

Якщо орієнтир спостерігається вперше, його положення обчислюється відносно дрона:

$$l_{xi} = x + r_i cos(\phi_i + \theta)$$
$$l_{yi} = y + r_i sin(\phi_i + \theta)$$

3.4.4 Використання Multi-sensor EKF SLAM

Інтеграція даних з різних сенсорів (IMU, GPS, Lidar) дозволяє:

- Отримувати точні оцінки положення дрона навіть за умов обмеженого доступу до GPS;
- Зменшувати невизначеність позицій орієнтирів за рахунок об'єднання інформації з різних джерел;

- Динамічно оновлювати карту оточення, зокрема для врахування нових або змінених орієнтирів.

Multi-sensor EKF SLAM – це потужний інструмент для автономної навігації, який дозволяє інтегрувати дані з кількох сенсорів для побудови карти та визначення локальної позиції. Цей підхід забезпечує високу точність навіть у складних умовах і є важливою складовою систем управління автономними дронами.

Початкові стани системи:

$$\mu = \begin{bmatrix} 1 \\ 1 \\ \pi \\ \frac{}{2} \end{bmatrix}$$

Переваги EKF SLAM:

1. **Математична строгість:** EKF SLAM базується на перевіреній математичній основі фільтра Калмана, що дозволяє отримувати точні оцінки стану навіть у складних умовах;
2. **Інтеграція багатьох сенсорів:** EKF SLAM дозволяє ефективно об'єднувати дані з різних сенсорів (Lidar, камери, GPS, IMU) для покращення оцінок позиції робота та орієнтирів;
3. **Висока точність:** Завдяки врахуванню кореляції між позицією робота та орієнтирами, алгоритм забезпечує точну локалізацію навіть у складних середовищах;
4. **Реалізація в реальному часі:** EKF SLAM є досить ефективним, щоб бути використаним у додатках реального часу, якщо кількість орієнтирів є помірною;
5. **Узгодженість моделі:** Алгоритм гарантує узгодженість між позицією робота та позицією орієнтирів, що важливо для побудови карти середовища.

Недоліки EKF SLAM:

1. **Обмеження в масштабі:** EKF SLAM є неефективним для великих середовищ із великою кількістю орієнтирів через квадратичне зростання обчислювальної складності (матриця коваріації збільшується у розмірі з кожним новим орієнтиром);
2. **Лінеаризація:** Використання матриці Якобіана для лінеаризації моделі може призводити до похибок, особливо в сильно нелінійних системах;
3. **Чутливість до початкових умов:** Некоректна ініціалізація позиції робота або орієнтирів може суттєво вплинути на результати роботи алгоритму;
4. **Шуми вимірювань:** Алгоритм вразливий до високих рівнів шуму, особливо якщо модель шуму не відповідає реальності;

5. **Обмеження щодо обчислювальних ресурсів:** EKF SLAM вимагає значних обчислювальних ресурсів для підтримки великих матриць стану та коваріації, що може бути проблематичним для мобільних роботів із обмеженими потужностями;

6. **Слабка масштабованість:** Для великих і складних середовищ необхідно використовувати оптимізовані підходи, такі як FastSLAM або Graph-SLAM, оскільки EKF SLAM не забезпечує достатньої ефективності [87].

Зокрема, проблему обмежень, щодо обчислювальних ресурсів можна виключити через специфіку системи дрону, де часто не потрібно запам’ятовувати усі перешкоди на шляху й достатньо орієнтування в просторі, тому деякі точки з плином часу можна виключати з переліку станів для збереження простоти розрахунків й ефективності обчислень.

Іншим вагомим обмеженням EKF SLAM є моделювання складних об'єктів середовища, таких як стіни чи нерегулярні форми. Це зумовлено тим, що алгоритм базується на представленні об'єктів у вигляді окремих точок, координати яких зберігаються у векторі стану. Такий підхід добре працює для простих орієнтирів, але має недоліки при роботі з великими чи складними структурами.

Наприклад, стіна, що складається з багатьох точок, потребує значного збільшення розміру матриці коваріації для врахування кожної з них. Це призводить до ускладнення обчислень і збільшення часу обробки. Якщо ж використовувати лише одну точку для представлення стіни, виникають суттєві похибки у моделюванні реальної геометрії об'єкта. Аналогічно, нерегулярні об'єкти, такі як дерева або транспортні засоби, не можуть бути точно представлені однією точкою, що обмежує деталізацію карти.

Ще однією проблемою є робота з динамічними об'єктами, які рухаються незалежно від робота. EKF SLAM не може чітко розділити рух робота і рух об'єкта, що ускладнює створення точної карти.

Для вирішення цих проблем використовуються декілька підходів. Наприклад, об'єкти середовища можуть бути спрощені до базових моделей, таких як лінії для стін чи контури для площ. Це дозволяє зменшити кількість орієнтирів і зберегти основну структуру середовища без суттєвих втрат точності. Інтеграція EKF SLAM із сучасними алгоритмами, такими як Graph-SLAM чи FastSLAM, також є ефективним підходом. Ці методи підтримують більшу кількість точок і можуть працювати з більш складними моделями середовища.

Поліпшення роботи з динамічними об'єктами та складними формами також можливе за попередньої обробки даних з сенсорів, таких як Lidar чи камери, зокрема моделями комп'ютерного зору, зокрема Yolo, що дозволяэ визначати ключові точки або межі об'єктів. Наприклад, замість представлення всієї стіни, Yolo може виділити її основні кути чи перехрестя та/або визначити тип об’єкту з його межами, що дозволяє оптимізувати модель середовища для SLAM.

Завдяки таким удосконаленням можна значно підвищити точність і продуктивність EKF SLAM. Однак навіть із цими покращеннями використання

EKF SLAM залишається оптимальним лише для середовищ із невеликою кількістю складних об’єктів, тоді як для високодеталізованих чи динамічних середовищ доцільно розглянути альтернативні підходи. Метод EKF SLAM використовується у зв’язку з простотою моделювання, що є цілком доцільним в межах дипломної роботи.

3.5. Використання алгоритму A* для побудови попередньої траєкторії до цілі та забезпечення оптимального шляху

Алгоритм A* (A-star) є одним із найефективніших і широко використовуваних алгоритмів пошуку найкоротшого шляху. Його застосування дозволяє забезпечити оптимальну траєкторію для дрона, враховуючи статичні та динамічні перешкоди на шляху до цільової точки. Використання цього алгоритму є важливим етапом в автономній навігації, оскільки він забезпечує побудову попередньої траєкторії, яка може бути адаптована за допомогою інших алгоритмів, таких як EKF чи SLAM.

3.5.1 Принцип роботи алгоритму A*

Алгоритм A* поєднує два підходи: пошук у ширину та використання евристичної функції, яка дозволяє оцінити відстань до цілі. Його робота базується на мінімізації наступного критерію [88, 89]:

$$f(n) = g(n) + h(n)$$

Де:

– $g(n)$ – вартість шляху від початкової точки до поточної вершини n;

– $h(n)$ – евристична оцінка вартості від поточної вершини n до кінцевої точки;

– $f(n)$ – загальна вартість шляху через вершину n.

3.5.2 Покроковий алгоритм A*

1. Ініціалізація:

- Всі вершини (клітинки) позначаються як неперевірені;
- Стартова вершина додається до відкритого списку $Open$, а її g-вартість дорівнює 0;
- Всі інші вершини отримують початкові значення $g = \infty,\ h = 0$.

2. Вибір вершини:

- Із відкритого списку $Open$ вибирається вершина з мінімальним значенням $f(n)$.

3. Оновлення сусідів:

- Для кожного сусіда поточної вершини розраховується нове значення g та f;
- Якщо новий шлях до сусіда кращий за попередній, оновлюються значення g, h, та додається посилання на поточну вершину як батьківську.

4. Рух до наступної вершини:

- Поточна вершина переноситься з відкритого списку $Open$ до закритого списку $Closed$.

5. Завершення:

- Алгоритм завершується, коли цільова вершина потрапляє до списку Closed, або коли відкритий список порожній (шлях відсутній).

3.5.3 Особливості використання A для дронів*

У випадку дронів алгоритм A* працює на основі карти середовища, яка може бути:

- **Статичною**: карта відома заздалегідь і не змінюється під час руху дрона;
- **Динамічною**: карта змінюється в режимі реального часу завдяки обробці даних з сенсорів, таких як Lidar, стереокамери чи RGBD-камери.

Дрон має використовувати A* для визначення початкової траєкторії, яка буде оновлюватися у реальному часі за допомогою алгоритмів локалізації та уникнення перешкод.

Переваги алгоритму A*:

1. **Оптимальність**: Завжди знаходить найкоротший шлях, якщо евристична функція є допустимою (не перевищує реальну відстань до цілі);
2. **Гнучкість**: Може працювати в середовищах з різною щільністю перешкод;
3. **Швидкість**: Завдяки евристичній функції A* працює значно швидше, ніж традиційний пошук у ширину.

Недоліки алгоритму A*:

1. **Високі обчислювальні витрати**: У складних середовищах потребує значних ресурсів пам'яті;
2. **Чутливість до евристики**: Неправильно вибрана евристична функція може значно знизити ефективність алгоритму.

3.5.4 Реалізація A для дрона*

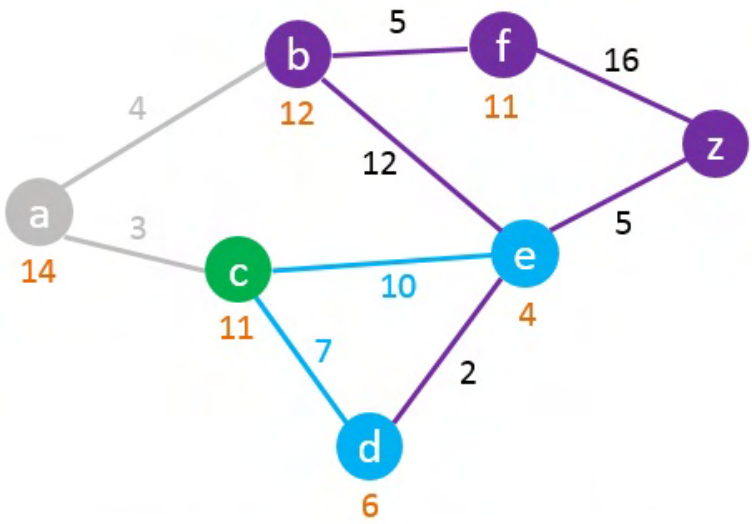
Алгоритм A* у контексті навігації дрона працює наступним чином:

1. На основі початкових даних створюється карта середовища;

2. A* знаходить оптимальний шлях до цільової точки, враховуючи статичні перешкоди;

3. Дрон починає рух за цим шляхом;

4. У разі виникнення динамічних перешкод, та/або статичних невідомих перешкод, маршрут перебудовується в режимі реального часу.



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Visited	0	14	14	
B		4	12	16	A
C	Current	3	11	14	A
D		∞ 3+7=10	6	16	C
E		∞ 3+10=13	4	17	C
F		∞	11		
Z		∞	0		

* Total Distance = Shortest Distance from A + Heuristic Distance to Z

Рисунок 3.10 – Принцип роботи алгоритму [90]

3.6. Інтеграція штучного інтелекту для визначення об’єктів

YOLO (You Only Look Once) – це одна з найпопулярніших моделей для задач комп'ютерного зору, зокрема для виявлення об'єктів у зображеннях у реальному часі. Ця модель використовує підхід, який відрізняється від традиційних методів виявлення об'єктів. Замість поділу задачі на окремі етапи (генерація

кандидатів, класифікація), YOLO виконує виявлення та класифікацію об'єктів за один прохід через нейронну мережу.

Основна ідея YOLO полягає в поділі зображення на сітку, де кожна клітинка відповідає за виявлення об'єкта, якщо його центр потрапляє в цю область. Для кожної клітинки модель передбачає обмежувальні рамки (bounding boxes), ймовірності наявності об'єкта та класи об'єктів. Завдяки цьому підходу YOLO здатна виконувати детекцію об'єктів у реальному часі, що робить її ідеальною для застосувань, де потрібна висока швидкість обробки, наприклад у дронах, системах відеоспостереження чи автономних транспортних засобах [91, 92].

Основні переваги YOLO:

1. **Швидкість**. YOLO здатна обробляти десятки кадрів на секунду завдяки одночасному виконанню виявлення та класифікації;
2. **Універсальність**. Модель може працювати з різними класами об'єктів на одній сцені, зокрема людьми, транспортними засобами, будівлями, тваринами тощо;
3. **Масштабованість**. Існують різні версії YOLO (наприклад, YOLOv4, YOLOv5, YOLOv8), які оптимізовані для різних апаратних платформ: від потужних серверів до компактних пристроїв на кшталт дронів [93].

Як YOLO використовується у SLAM:

YOLO може бути інтегрована у системи SLAM для поліпшення ідентифікації та моделювання середовища. Наприклад:

- **Виявлення динамічних об'єктів**. YOLO дозволяє визначати рухомі об'єкти (автомобілі, люди, тварини) і виділяти їх із моделі статичного середовища, що важливо для коректної локалізації;
- **Ключові точки для орієнтирів**. Замість використання випадкових точок для визначення об’єктів середовища, YOLO може виявляти специфічні об’єкти (наприклад, дорожні знаки чи світлофори), які слугують стабільними орієнтирами;
- **Семантична сегментація**. Завдяки класифікації об’єктів, система може створювати семантично розширені карти, що містять не лише геометричну, а й контекстну інформацію про середовище.

Виклики інтеграції YOLO:

1. **Високі обчислювальні витрати**. Хоча YOLO є швидкою, для її ефективного виконання потрібна потужна апаратна платформа;
2. **Обмеження виявлення**. Модель може мати проблеми з виявленням об’єктів у складних умовах, таких як слабке освітлення, висока швидкість руху чи накладання об’єктів;
3. **Точність локалізації**. Похибки у виявленні обмежувальних рамок можуть впливати на точність SLAM.

Таким чином, YOLO є потужним інструментом для підвищення ефективності SLAM, особливо у випадках, коли важливо враховувати контекст

середовища чи працювати з динамічними об'єктами. Інтеграція таких моделей в алгоритми SLAM дозволяє створювати більш інтелектуальні системи, здатні орієнтуватися в реальному середовищі з урахуванням його складності та багатогранності.

3.6.1 Постановка задачі: Реалізація зчитування даних з камери для ідентифікації об'єктів за допомогою YOLO

Однією з важливих складових автономної навігації безпілотного літального апарата є здатність розпізнавати та ідентифікувати об’єкти в середовищі. Це завдання потребує ефективних алгоритмів комп’ютерного зору, які можуть працювати в реальному часі та забезпечувати високу точність.

Метою цього підрозділу є реалізація системи обробки даних із камери, яка дозволяє розпізнавати об'єкти, класифікувати їх і визначати положення відносно дрона. Для цього буде використано модель YOLO (You Only Look Once) як одну з найбільш сучасних і швидких архітектур нейронних мереж для виявлення об'єктів.

Завдання полягає у наступному:

1. **Обробка відеопотоку з камери.** Реалізувати зчитування даних із камери дрона, забезпечуючи їх подальшу передачу в нейронну мережу для аналізу;
2. **Розпізнавання об’єктів.** Використовувати модель YOLO для ідентифікації об’єктів у кадрі, їх класифікації та визначення меж (bounding boxes);
3. **Використання ідентифікованих об’єктів.** Інтегрувати розпізнані об’єкти в систему навігації для аналізу перешкод і уточнення локалізації дрона.

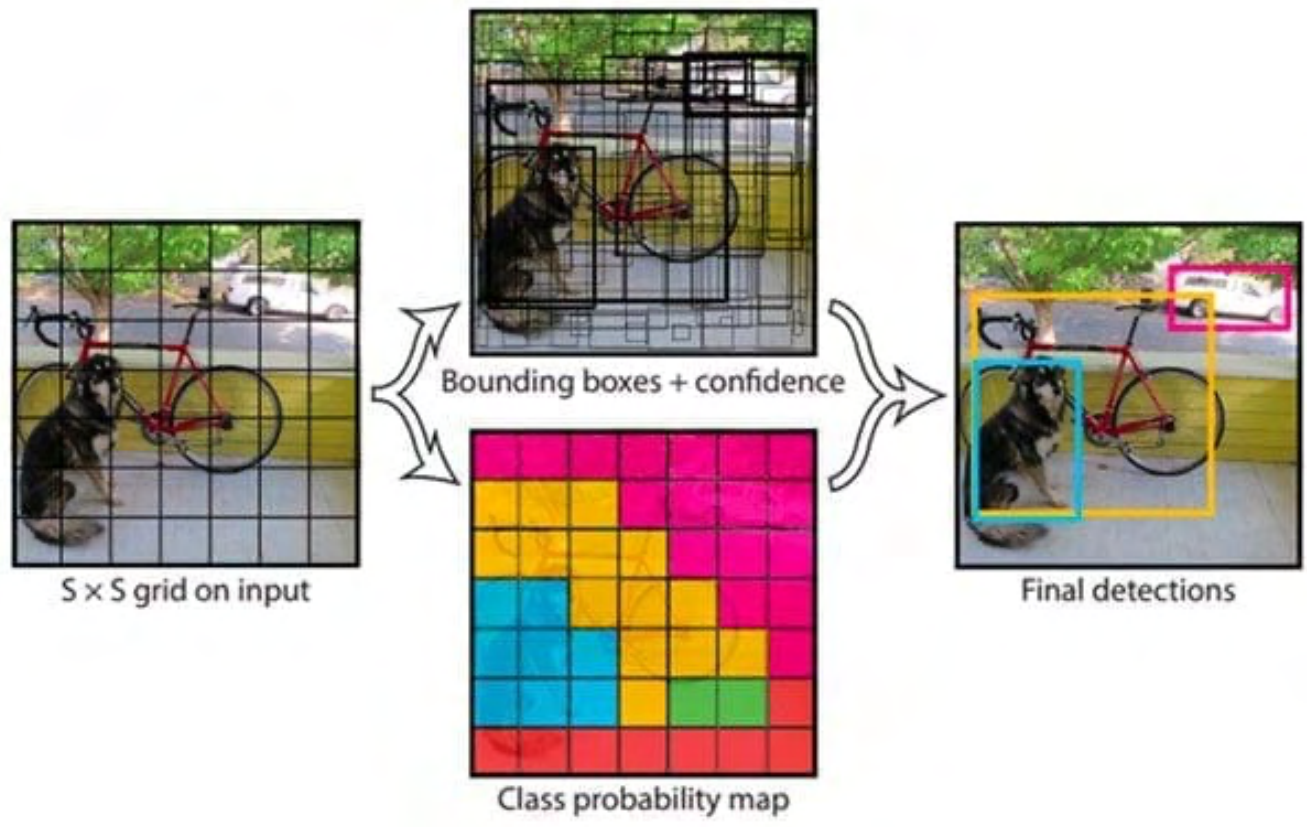


Рисунок 3.11 – Принцип роботу бібліотеки комп’ютерного зору Yolo

Висновки за розділом

У третьому розділі було проведено розробку комплексної системи автономної навігації дрону, що базується на інтеграції сучасних алгоритмів, сенсорних систем та методів обробки даних. Розглянуто архітектуру системи, яка включає апаратні компоненти, такі як IMU, Lidar, камери, та програмне забезпечення для локалізації, побудови карт і маршруту.

Особливу увагу приділено використанню Multi-sensor EKF SLAM, що дозволяє забезпечити точну локалізацію дрону навіть в умовах обмеженого доступу до GPS. Вектор стану та матриця коваріації детально описані, а також пояснено важливість кореляції між роботом та орієнтирами. Завдяки EKF SLAM було вирішено задачі побудови карти середовища й уточнення позиції дрону на основі даних з кількох сенсорів.

Окремо розглянуто алгоритм A*, який забезпечує побудову оптимального маршруту до цілі з урахуванням перешкод. Переваги та обмеження цього алгоритму були детально проаналізовані в контексті його застосування для дронів.

Також у цьому розділі було висвітлено інтеграцію моделі YOLO для ідентифікації об'єктів. Це дозволяє доповнити систему автономної навігації семантичними даними, що важливо для коректного відображення середовища та уникнення перешкод.

Таким чином, третій розділ обґрунтовує необхідність комплексного підходу до автономної навігації дрону, який включає використання різних типів сенсорів, сучасних алгоритмів локалізації та прокладання маршруту, а також інструментів комп’ютерного зору. Такий підхід забезпечує високу точність, адаптивність і стабільність системи в реальних умовах.

4 ЕКСПЕРИМЕНТАЛЬНЕ МОДЕЛЮВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОЇ СИСТЕМИ

4.1. Опис моделювання та тестового середовища для автономної навігації з мінімальним використання GPS-сенсору

Моделювання систем автономної навігації є важливим етапом у розробці, оскільки дозволяє перевірити алгоритми в контрольованих умовах без потреби у фізичних експериментах. У межах цієї роботи для моделювання було обрано Python, який забезпечує гнучкість, багатофункціональність та доступ до численних бібліотек, таких як NumPy, Matplotlib, Scipy та інші. Це рішення дозволяє швидко реалізувати та протестувати алгоритми, зокрема фільтри Калмана, A* та EKF SLAM, інтегруючи дані з різних сенсорів. Основна мета – мінімізувати залежність від GPS, використовуючи альтернативні джерела інформації, такі як IMU, барометри та камери, дослідити складові алгоритми та підходи.

Під час моделювання особлива увага приділялася імітації шумів, які притаманні даним реальних сенсорів. Для цього до сенсорів додавалися гаусівські шуми, що дозволяє моделювати невизначеність вимірювань і оцінювати стабільність алгоритмів. Було створено сценарії для навігації в умовах відсутності або обмеженого використання GPS.

Моделювання було виконане в Python з акцентом на простоту реалізації та можливість швидкого налаштування параметрів середовища. Хоча Gazebo пропонує розширені можливості для створення реалістичних середовищ із фізичними моделями, його використання потребує значних витрат часу й специфічних знань, таких як робота з ROS. У цьому проєкті вибір на користь Python був зумовлений необхідністю зосередитися на швидкому прототипуванні та демонстрації алгоритмів.

Такий підхід забезпечив простоту інтеграції алгоритмів обробки даних, швидкість налаштування експериментів і можливість моделювання складних сценаріїв, що включають визначення позиції дрона та створення карти середовища, знаходження короткого шляху від об’єкту до пункту призначення – дослідження цих алгоритмів по окремості.

4.2. Аналіз точності позиціонування на основі IMU, GPS і фільтра Калмана

За результатами роботи скрипту Python, відображено результати використання фільтра Калмана для об'єднання даних з IMU і GPS з метою покращення точності позиціонування (Рисунок 4.1).

На першому графіку зображено динаміку позиції у часі для кількох джерел даних: реальне положення, GPS-вимірювання, позиція, оцінена через подвійне інтегрування даних IMU, і положення, отримане за допомогою фільтра Калмана. Реальне положення позначено штриховою лінією, а інші джерела показують значні відхилення, спричинені шумами. Дані GPS мають найбільший розкид, що очікувано через високий рівень шуму цього сенсора. Дані IMU, хоч і виглядають більш стабільними, демонструють тенденцію до накопичення помилок через інтегрування шуму прискорення. Натомість фільтр Калмана дозволяє об'єднати ці два джерела, забезпечуючи більш точне відображення положення у часі, яке наближається до реального.

Другий ряд графіків представляє різницю між вимірюваннями і реальними даними. Лівий графік показує розбіжності між GPS-даними та реальним положенням, підкреслюючи високий рівень шуму GPS. Центральний графік демонструє похибки IMU, що збільшуються з часом через накопичення інтеграційних помилок. Правий графік, присвячений фільтру Калмана, свідчить про значне зниження відхилень порівняно з даними окремих сенсорів.

Для кількісної оцінки точності позиціонування було обчислено середньоквадратичні помилки (MSE) для кожного випадку:

– MSE (Mean Squared Error, середньоквадратична похибка) для GPS-вимірювань: **8.34**. Це підкреслює високий рівень шуму в даних GPS.

– MSE для позиції, оціненої через IMU (подвійне інтегрування): **7.32**. Значення ілюструє вплив накопичуваних помилок IMU.

– MSE для фільтра Калмана: **1.66**. Цей результат підтверджує значне зниження похибок після об'єднання даних із сенсорів.

Аналіз різниці між позицією, оціненою через фільтр Калмана, та реальним положенням показує, що алгоритм успішно згладжує шум GPS і коригує накопичувані помилки IMU. Хоча відхилення все ще присутні, їхня амплітуда значно знижена, що забезпечує більшу стабільність і надійність даних.

Таким чином, результати підтверджують ефективність використання фільтра Калмана для об'єднання даних з GPS і IMU. Фільтр дозволяє не лише згладжувати шум, а й компенсувати недоліки кожного окремого сенсора, забезпечуючи високу точність позиціонування у складних умовах [94].

4.3 Використання моделі штучного бачення YOLO для визначення об’єктів та перешкод

Для виявлення об'єктів на зображеннях та у відеопотоці використовується модель YOLO (You Only Look Once), яка є однією з найбільш популярних архітектур для задач розпізнавання об'єктів. Модель YOLOv3 була обрана через її здатність забезпечувати високу точність при достатній швидкості обробки.

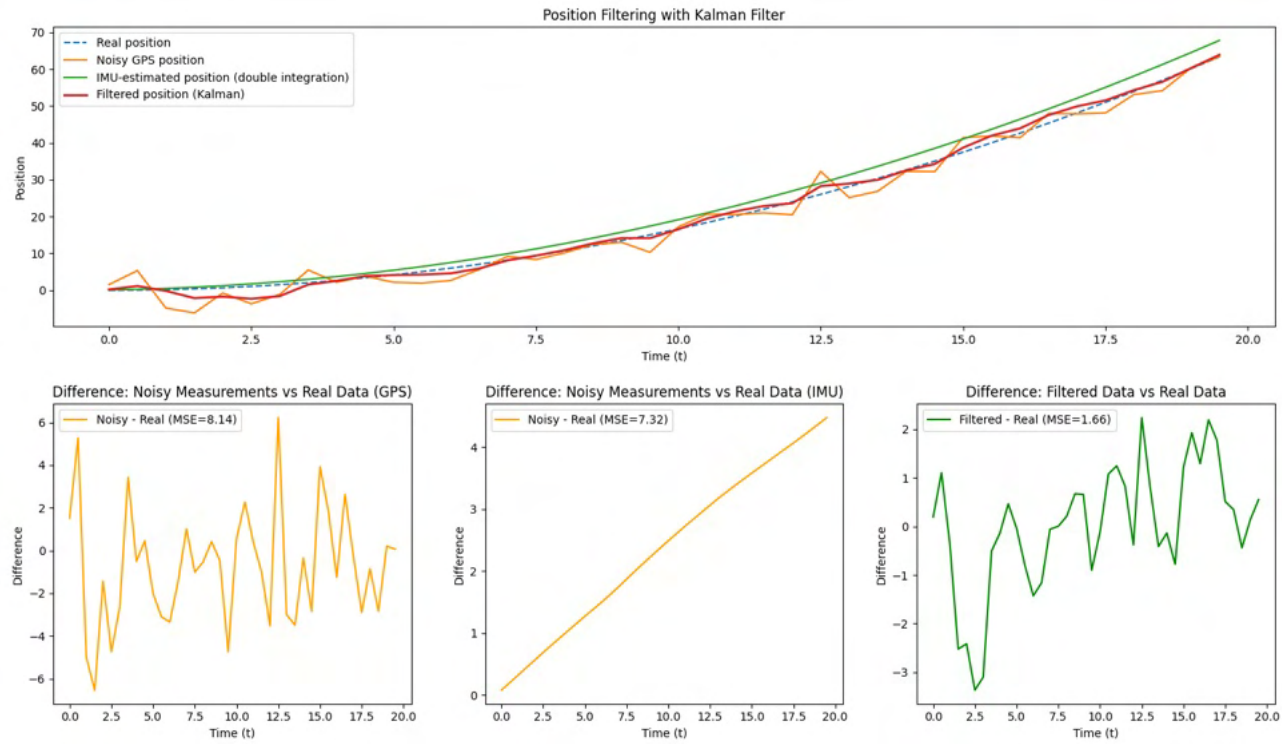


Рисунок 4.1 – Отримані графіки використання [94]

Модель була завантажена з використанням бібліотеки OpenCV, а для підготовки зображень використовувався модуль cv2.dnn. Було враховано специфічні параметри, такі як розмір вхідного зображення (416x416 пікселів) і нормалізація пікселів.

Програма починається з завантаження моделі YOLO та класів об'єктів із набору даних COCO, після чого виконується обробка зображень та відеопотоку з веб-камери.

Проведемо тестування даного коду з бібліотекою YOLO [95].

Зокрема, наведемо декілька прикладів тестування програми:

1. **Тестування на зображеннях із різною кількістю об'єктів** – програма правильно виявляла людей на зображеннях, навіть якщо вони перебували в різних положеннях або частково перекривалися іншими об'єктами.

Для початкового тестування було обрано просте зображення дівчинки, де вона є єдиним об'єктом на фоні (Рисунок 4.2а). Це дозволило перевірити здатність програми розпізнавати об'єкт за відсутності складного фону або інших відволікаючих елементів.

Програму було протестовано на цьому зображенні, щоб оцінити її здатність точно визначати межі та ідентифікувати об'єкт. Результат обробки наведено на Рисунку 4.2б. Як видно, програма коректно обробляє зображення, ідентифікуючи дівчинку з вірогідністю 100 відсотків. Це підтверджує її ефективність у розпізнаванні об'єктів на простих фонах без додаткових відволікаючих факторів;

2. **Тестування на зображеннях з автомобілями на фоні складної сцени** – програма виявила автомобілі, незалежно від кольору чи моделі, демонструючи здатність до розпізнавання об'єктів навіть на фоні інших деталей сцени.

Для подальшого тестування було обрано складніше зображення, на якому у великій кількості присутні як люди, так і автомобілі. Це зображення, представлене на Рисунку 4.3а, має різноманітний фон і багато об'єктів, що дозволяє перевірити здатність програми справлятися зі складними сценами та правильно розпізнавати декілька об'єктів одночасно.

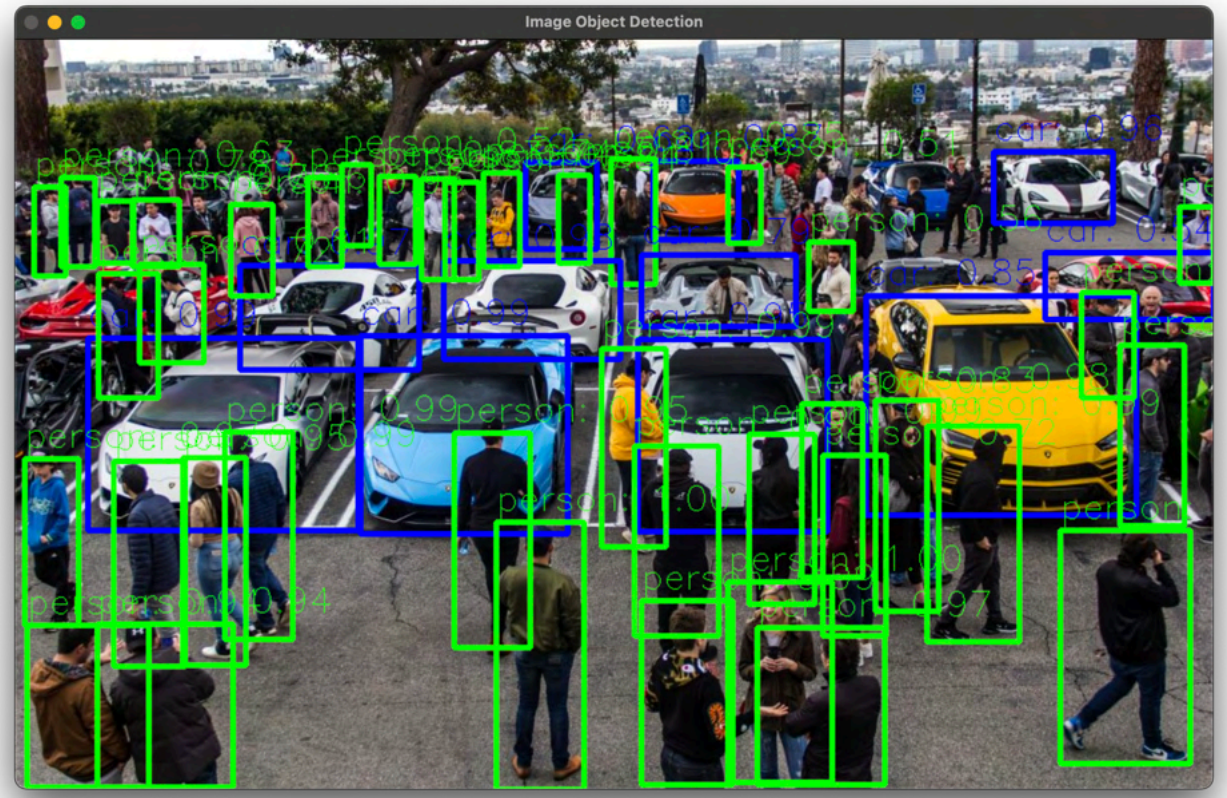
Результат обробки зображення показано на Рисунку 4.3б. Програма успішно ідентифікувала всі об'єкти, навіть коли вони частково перекривалися або перебували на складному фоні, демонструючи свою високу точність і надійність у складних умовах.

а

б

Рисунок 4.2 – Фотографія дівчинки [96]

а



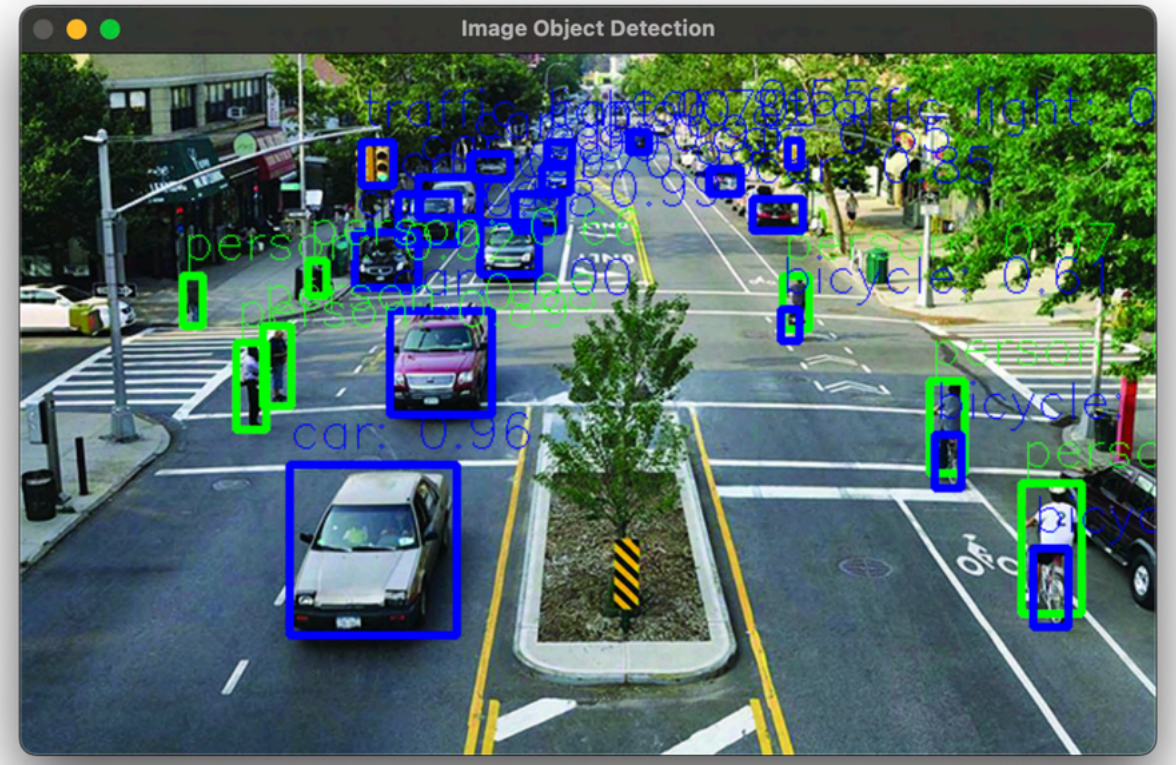
б

Рисунок 4.3 – Складніша фотографія [97]

Далі проведемо тестування програми на ще одному складному зображенні, зробленому за допомогою камери відеоспостереження, що представлено на Рисунку 4.4а. Це зображення є особливо цікавим для перевірки, оскільки воно включає типові умови відеоспостереження – низька роздільна здатність, можливі перекриття об'єктів, а також різні рівні освітлення.

Оброблена версія цього зображення наведена на Рисунку 4.4б. Результати показують, що програма здатна ефективно розпізнавати об'єкти в умовах, характерних для відеоспостереження, підтверджуючи свою здатність працювати з різними джерелами даних та в різних середовищах з можливістю використання на дронах.

а



б

Рисунок 4.4 – Фотографія з камери спостереження [98]

3. **Тестування з використанням веб-камери у режимі реального часу** – система успішно ідентифікувала об'єкти, як-от люди, що рухалися перед камерою, із незначною затримкою в обробці, що доводить ефективність роботи у реальному часі.

Для завершення тестування було проведено випробування програми в режимі реального часу з використанням веб-камери, результати якого зображено на Рисунку 4.5. Цей тест є особливо важливим, оскільки дозволяє оцінити ефективність роботи додатку в умовах реального часу, де необхідна швидка обробка кадрів та миттєва реакція на появу нових об'єктів.

Як видно з результатів на Рисунку 4.5, програма успішно ідентифікує всі об'єкти в кадрі, незалежно від їх кількості та розташування. Додаток коректно працює, виявляючи всі наявні об'єкти в режимі реального часу, що підтверджує його надійність та функціональність у динамічних умовах.

Таким чином, результати тестування підтвердили здатність програми ефективно розпізнавати різні об'єкти як на статичних зображеннях, так і у режимі реального часу, що робить її придатною для широкого спектра практичних застосувань, зокрема у системі ідентифікації об’єктів з подальшим використанням в EKF SLAM.

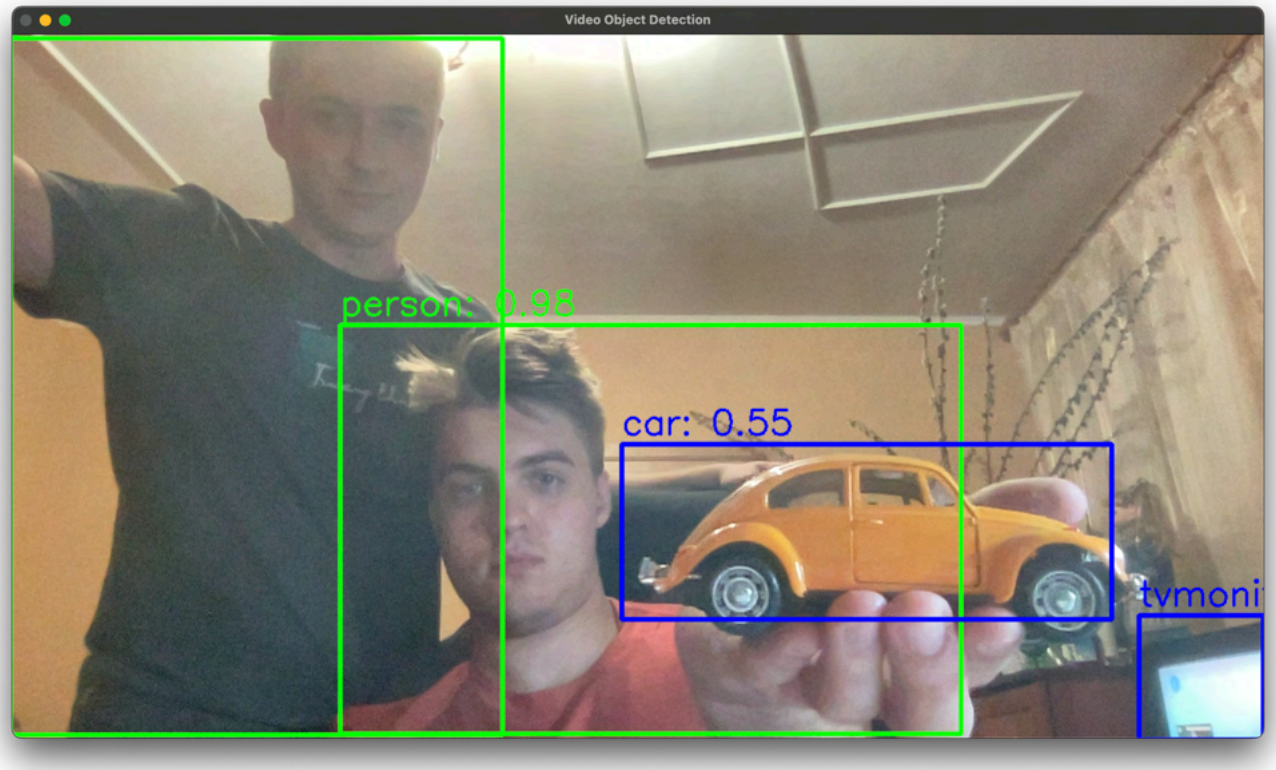


Рисунок 4.5 – Тестування роботи з відеорядом з веб-камери

4.4 Тестування точності та стабільності SLAM у визначенні відносного положення

Протестуємо алгоритм EKF SLAM за допомогою демонстраційного коду EKF SLAM demo [99]. Дана програма моделює роботу алгоритму SLAM, використовуючи імітаційні дані для оцінки точності визначення відносного положення об'єктів у середовищі та локалізації самого робота. Як видно на Рисунку 4.6, робот рухається в середовищі, що містить низку статичних орієнтирів, представлених у вигляді блакитних точок.

На тестуванні робот оснащений сенсорами, які забезпечують вимірювання його руху (наприклад, IMU) та вимірювання положення орієнтирів (наприклад, Lidar або камера). Алгоритм EKF SLAM використовує ці дані для визначення поточної позиції робота та оновлення карти середовища. Робот одночасно оцінює власне положення і положення орієнтирів за допомогою розширеного фільтра Калмана (EKF), що дозволяє зменшити вплив сенсорних шумів і підвищити точність визначення станів.

На графічному інтерфейсі можна побачити кілька важливих аспектів роботи алгоритму:

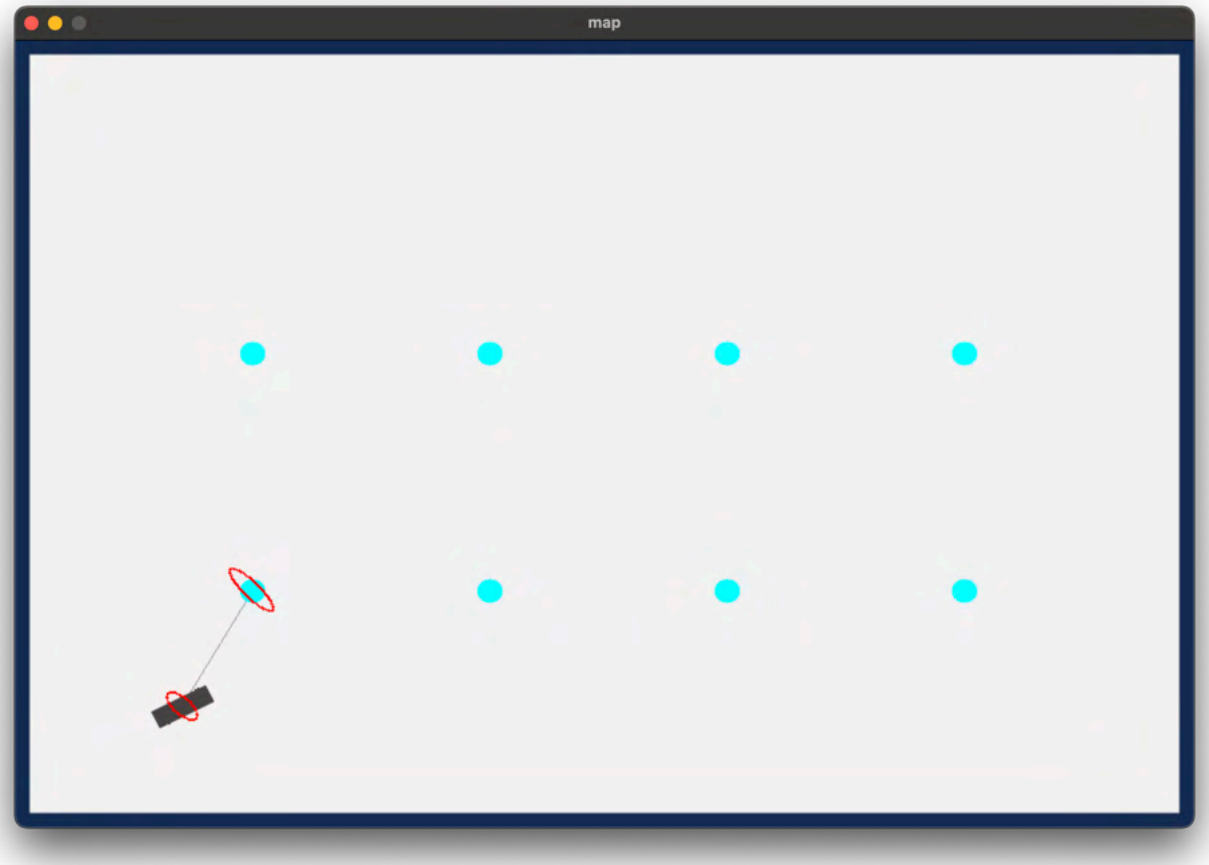
1. Робот (позначений сірим прямокутником) рухається по середовищу, здійснюючи спостереження за орієнтирами;

2. Кожен орієнтир (блакитні точки) визначається з певною невизначеністю, яка поступово зменшується з часом завдяки уточненню даних на основі нових спостережень;

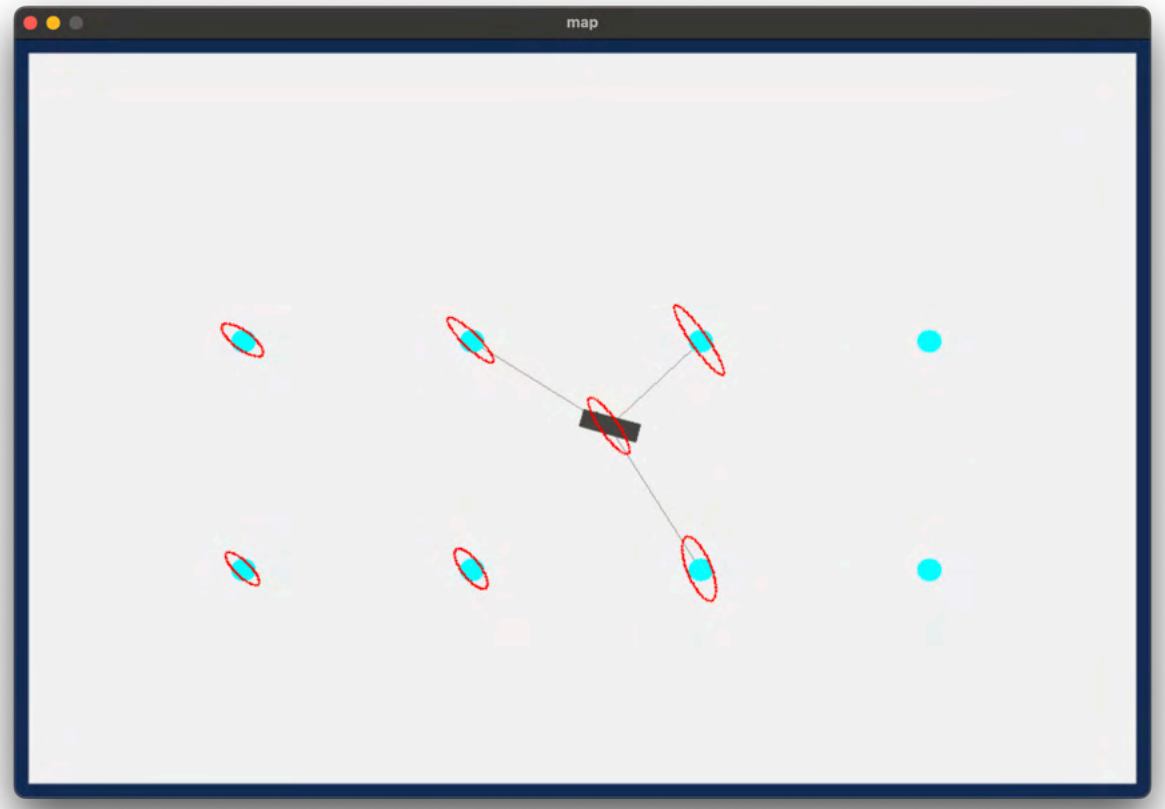
3. Відносне положення робота уточнюється під час кожного циклу спостереження, що забезпечує стабільну локалізацію навіть у складних умовах, червоний еліпс демонструє стандартне відхилення, що отримано з коваріаційної матриці.

Основна перевага EKF SLAM у цьому тесті полягає в його здатності адаптивно зменшувати невизначеність як у позиції робота, так і в позиціях орієнтирів, що видно за Рисунками роботи. Робота алгоритму підтверджує його ефективність для динамічного середовища, де доступ до глобальної GPS-навігації може бути обмеженим або недоступним.

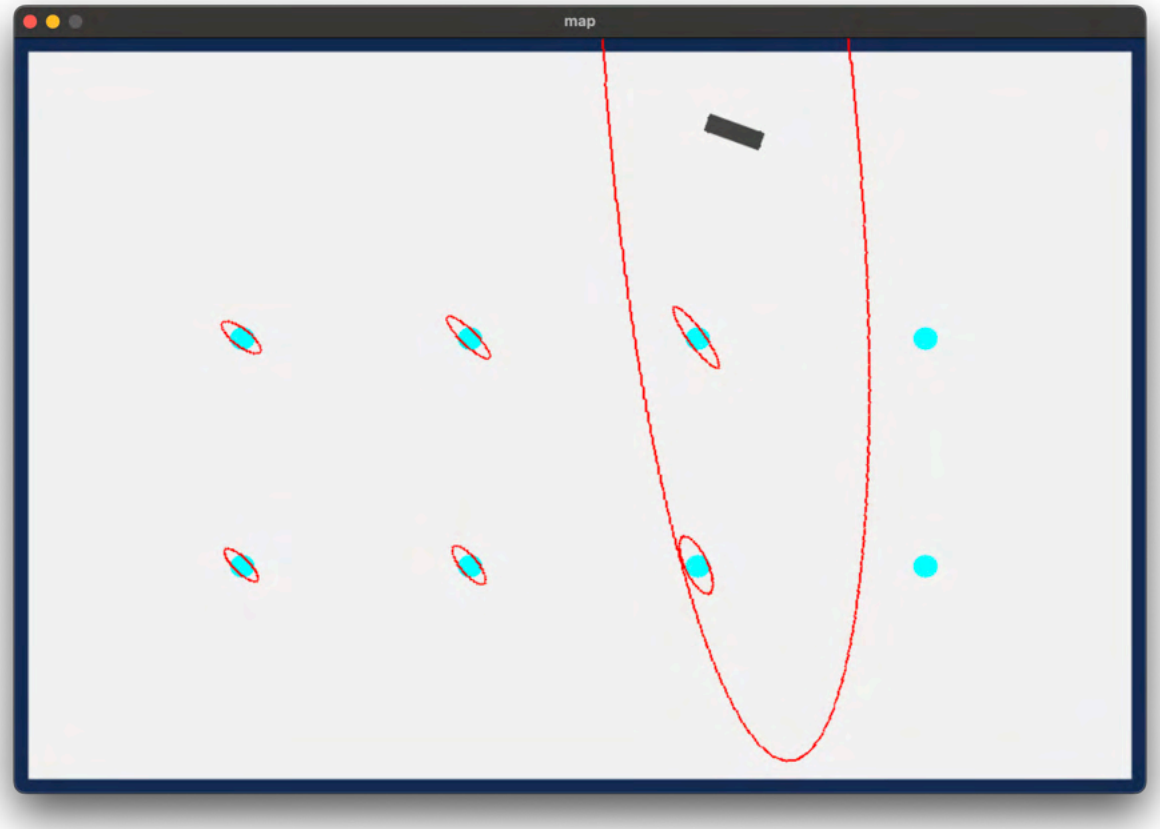
Отже, тестування продемонструвало, що EKF SLAM стабільно працює для систем автономної навігації, забезпечуючи точне визначення відносного положення навіть у присутності шумів і обмежених сенсорних даних. Зазначений підхід може бути використаний у реальних сценаріях, таких як роботи-дослідники або дрони, які функціонують у невідомих середовищах.



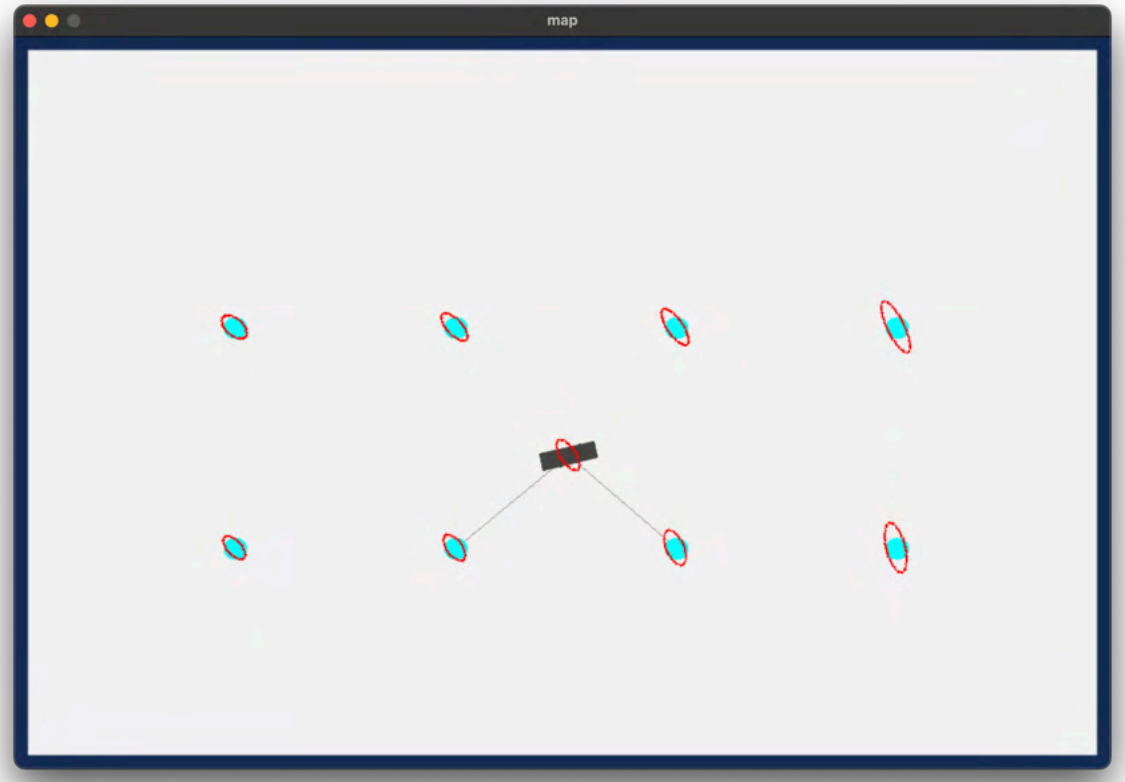
а



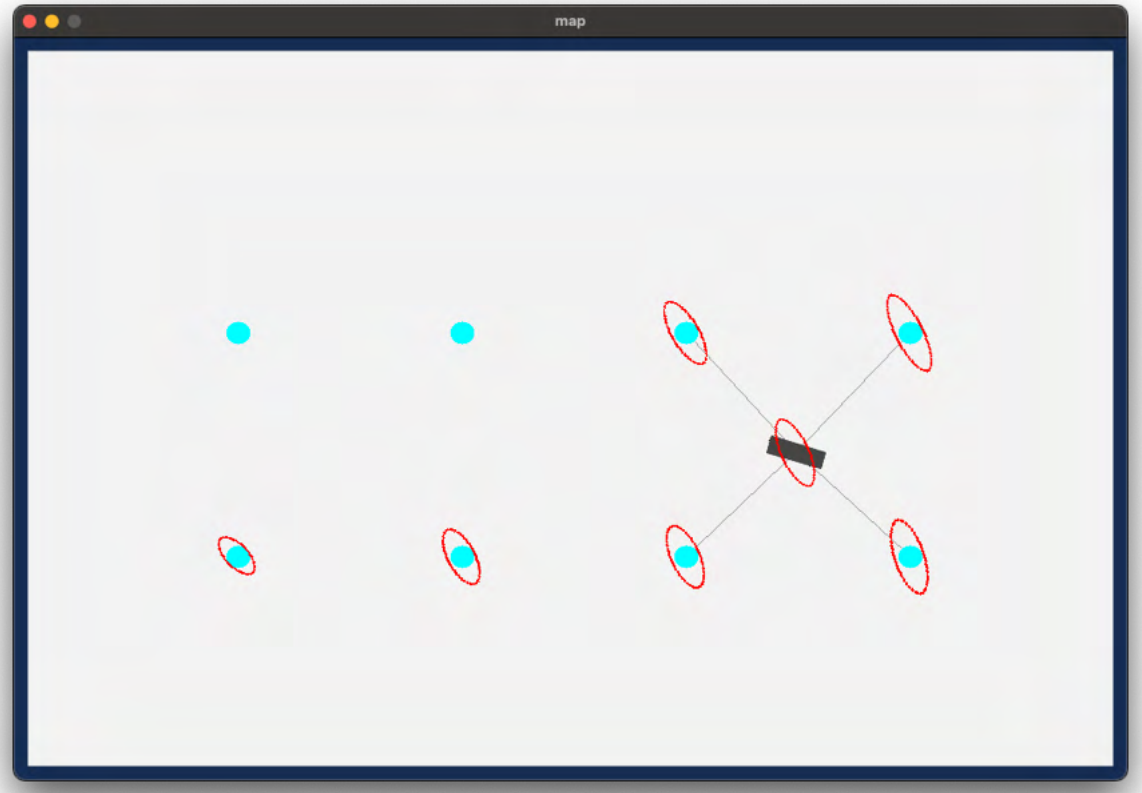
б



в



г



г

Рисунок 4.6 – Демонстрація роботи EKF SLAM

4.5 Результати тестування алгоритмів прокладання маршруту A*

Для оцінки ефективності роботи алгоритму A* було створено тестове середовище у вигляді лабіринту, де завдання полягає у знаходженні найкоротшого шляху від точки старту до точки призначення. На Рисунку 4.7 зображено результати роботи алгоритму, який успішно знаходить оптимальні маршрути для навігації.

Алгоритм A* використовує евристичну функцію для оцінки вартості шляху, яка комбінує вартість пройденого шляху від початкової точки до поточної (g) та оцінку вартості від поточної точки до цілі (h). У тестовому середовищі були застосовані дві основні евристики — евклідову та мангеттенську метрики. Вони дозволяють враховувати характеристики простору та забезпечують швидке знаходження найкращого маршруту.

На рисунку 4.7 видно, що алгоритм A* успішно обходить перешкоди і знаходить оптимальний шлях навіть у складному середовищі. Оптимізація маршруту відбувається за рахунок точного вибору наступної вершини для розгляду на основі мінімізації значення $f = g + h$.

На рисунку 4.7 також видно, що алгоритм знаходить маршрут у менш завантаженому середовищі. Алгоритм ефективно уникнув перешкоди, оптимізувавши шлях таким чином, щоб мінімізувати загальну довжину маршруту.

Під час тестування було відзначено, що A* демонструє високу швидкість та ефективність, але складність алгоритму прямо залежить від розмірності простору та кількості перешкод. У середовищах із великою кількістю перешкод збільшується кількість обчислень, необхідних для пошуку оптимального маршруту. Проте навіть у таких випадках A* забезпечує правильність та оптимальність знайденого маршруту.

4.6. Порівняння комплексного підходу з традиційними методами навігації: обґрунтованість комбінованого підходу

Комбінований підхід до вирішення задач автономної навігації полягає у поєднанні даних від різноманітних сенсорів та алгоритмів, що дозволяє досягти більшої точності та надійності у порівнянні з традиційними методами. Інтеграція різних джерел інформації спрямована на компенсацію обмежень окремих компонентів системи. Наприклад, методи, які покладаються виключно на GPS, забезпечують глобальну локалізацію, проте втрачають ефективність у середовищах із обмеженим доступом до супутникового сигналу, таких як міські каньйони, густі ліси чи підземні приміщення.

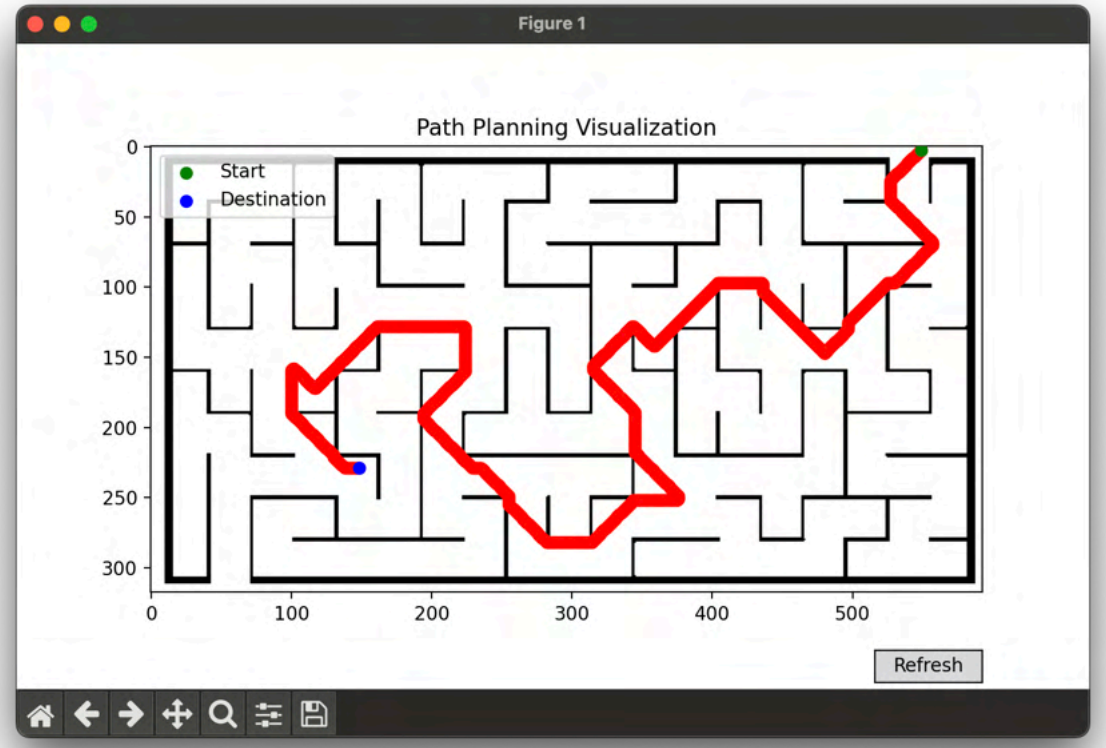


Figure 1
Path Planning Visualization
Start
Destination
0
50
100
150
200
250
300
0
100
200
300
400
500
Refresh

а

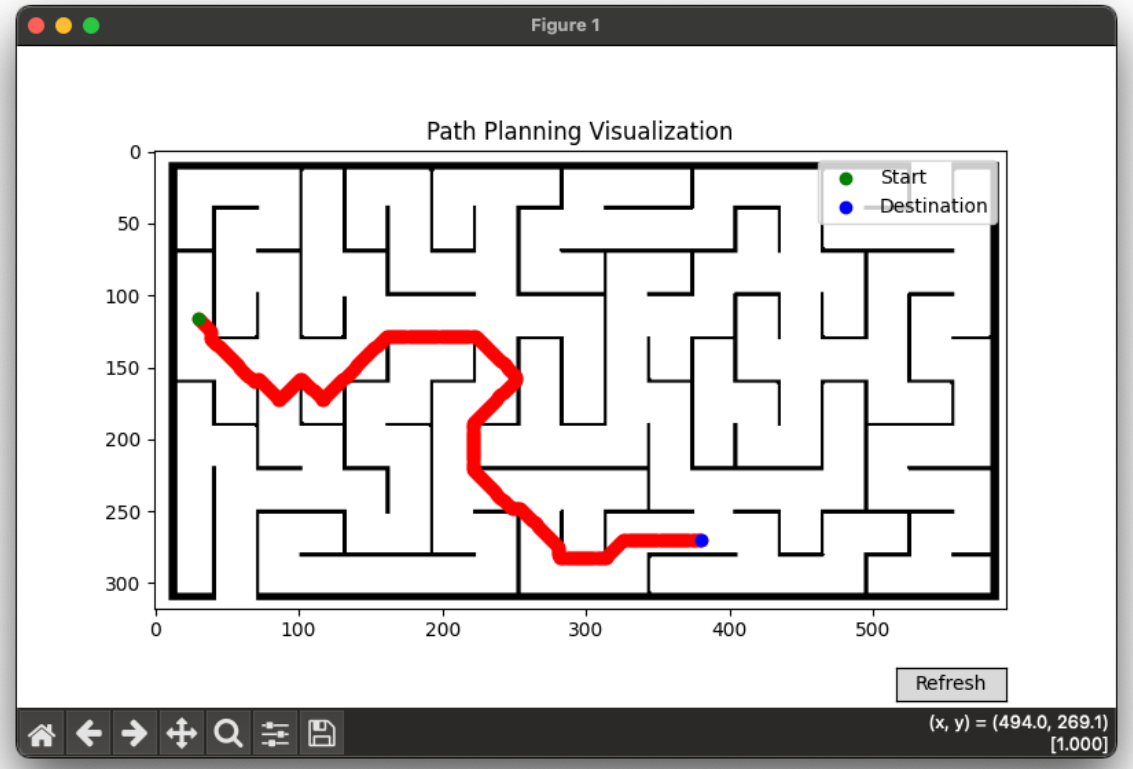
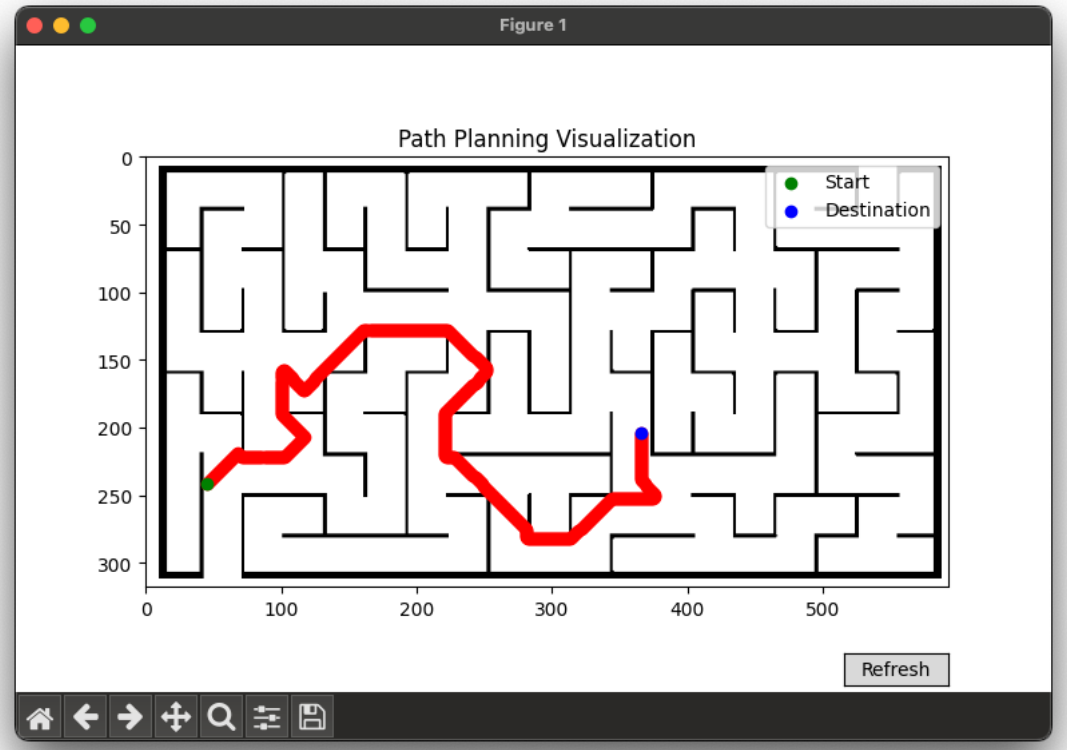
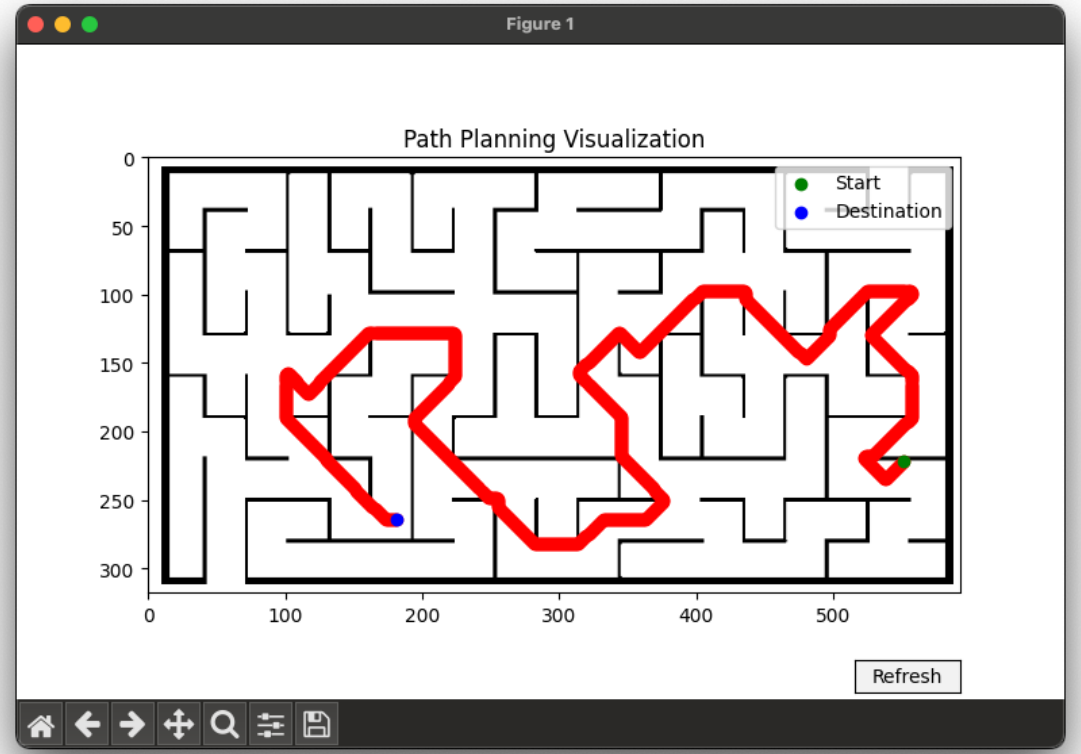


Figure 1
Path Planning Visualization
Start
Destination
0
50
100
150
200
250
300
0
100
200
300
400
500
Refresh
(x, y) = (494.0, 269.1)
[1.000]

б



В



Г

Рисунок 4.7 – Перевірка працездатності алгоритму пошуку шляху

Комплексний підхід включає використання EKF SLAM, алгоритму A*, а також даних із різноманітних сенсорів, зокрема IMU, оптичного потоку, LiDAR, камер та GPS. Це дозволяє значно покращити точність навігації та забезпечити автономну роботу системи навіть за відсутності сигналу GPS. EKF SLAM є потужним інструментом для локалізації, який дозволяє інтегрувати дані з IMU та камер, створюючи карту середовища та визначаючи позицію дрона. Алгоритм A* забезпечує ефективну побудову маршруту в складних середовищах, уникаючи перешкод і зберігаючи оптимальність траєкторії. Разом ці компоненти створюють надійну систему, здатну до автономної роботи в умовах, де традиційні методи не дають очікуваного результату.

Традиційні підходи, які базуються виключно на GNSS/GPS або інерційних системах, мають істотні обмеження. GNSS/GPS надає абсолютну позицію, але є вразливим до сигналів слабкої якості або зовнішніх перешкод. Інерційні системи, такі як IMU, страждають від накопичення похибок, що призводить до значного дрейфу у довгостроковій перспективі. Комбінований підхід дозволяє поєднувати сильні сторони обох технологій: GPS забезпечує абсолютну позицію для корекції, тоді як IMU гарантує високу частоту оновлення даних, необхідну для швидких маневрів.

Використання камер та алгоритмів виявлення об’єктів, таких як YOLO, також є важливою складовою. Завдяки цьому система отримує змогу розпізнавати перешкоди та визначати орієнтири, що покращує якість локалізації та деталізацію карти. У випадку втрати одного з джерел даних, наприклад GPS, система може продовжувати роботу за рахунок IMU, оптичного потоку та SLAM.

Порівняння з традиційними методами навігації демонструє значні переваги комбінованого підходу. Він забезпечує гнучкість та стійкість до відмов, дозволяє адаптувати систему до різноманітних умов роботи та інтегрувати нові сенсори чи алгоритми. Такий підхід має високу практичну цінність для автономної навігації в складних середовищах, де необхідно забезпечити точність, надійність і стабільність.

Висновок за розділом

Четвертий розділ роботи присвячено експериментальному моделюванню, оцінці ефективності та аналізу розроблених алгоритмів для автономної навігації. У цьому розділі детально розглянуто інтеграцію різних сенсорів та алгоритмів з метою підвищення продуктивності автономних систем у реальних умовах експлуатації.

Початково зосереджено увагу на об'єднанні даних із GPS та IMU (інерціальної навігаційної системи) за допомогою фільтра Калмана. Це дозволило значно зменшити вплив випадкових шумів і накопичуваних помилок, що є невід'ємною частиною сенсорних вимірювань. Фільтр Калмана діє як оптимальний спостерігач, комбінуючи дані з різних джерел та надаючи більш точну та стабільну оцінку позиції. Особливо це важливо в умовах, де сигнали GPS можуть бути нестабільними або заблокованими, наприклад, у міських каньйонах чи лісових масивах.

Далі впроваджено модель глибинного навчання YOLO (You Only Look Once) для виявлення об'єктів у реальному часі. Експерименти показали, що ця модель демонструє високу точність розпізнавання навіть у складних умовах, таких як часткове перекриття об'єктів, динамічні сцени чи недостатнє освітлення. Використання YOLO забезпечує швидку обробку зображень, що є критичним для систем автономної навігації, де затримки можуть призвести до небажаних наслідків.

Алгоритм EKF SLAM (Extended Kalman Filter Simultaneous Localization and Mapping) протестовано для одночасної локалізації та побудови карти невідомого середовища. Результати підтвердили стабільність та надійність алгоритму у визначенні відносного положення робота та орієнтирів. Навіть за умов обмежених сенсорних даних EKF SLAM успішно будує карту оточення, що дозволяє роботу орієнтуватися у невідомому середовищі та приймати обґрунтовані рішення щодо подальшого руху.

Окремо проведено детальний аналіз алгоритму A* для планування маршрутів у складних середовищах з численними перешкодами. Алгоритм демонструє здатність швидко знаходити оптимальні шляхи, обминаючи статичні та динамічні перешкоди. Висока швидкість обчислень забезпечує оперативну реакцію системи на зміну умов, що є важливим для підтримки безпечної та ефективної навігації.

Таким чином, результати моделювання підтверджують, що комплексний підхід, заснований на інтеграції різних сенсорів і передових алгоритмів, забезпечує високу точність, надійність та стабільність роботи системи в умовах, де традиційні методи є менш ефективними. Цей підхід дозволяє системі адаптуватися до різноманітних сценаріїв та умов експлуатації, розширюючи її застосування у різних галузях.

Перспективи подальшого вдосконалення включають впровадження додаткових сенсорів, таких як лідари, радари чи ультразвукові датчики, що може підвищити точність та надійність системи, особливо у складних погодних умовах або при відсутності візуальних даних. Адаптація алгоритмів до більш складних сценаріїв, зокрема розвиток методів машинного навчання та штучного інтелекту, покращить здатність системи до самонавчання та адаптації у динамічних середовищах з великою кількістю рухомих об'єктів. Крім того, оптимізація продуктивності шляхом вдосконалення алгоритмів для зменшення

обчислювальних витрат та енергоспоживання є критичною для автономних систем з обмеженими ресурсами.

ВИСНОВОК

У результаті виконання дипломної роботи було реалізовано та протестовано ключові алгоритми, що входять до складу сучасних систем автономної навігації дрону. Зокрема, це фільтр Калмана, EKF SLAM і алгоритм прокладання маршруту A*. Проведені експерименти продемонстрували високу ефективність цих алгоритмів у вирішенні завдань автономного переміщення та локалізації дрону навіть в умовах мінімального покладання на GPS-модуль. Такий підхід є перспективним для використання в середовищах зі слабким або повністю відсутнім сигналом GPS, забезпечуючи високу автономність системи та підвищуючи її функціональні можливості.

Фільтр Калмана був використаний для об’єднання даних із різних сенсорів, таких як IMU і GPS, що дозволило забезпечити більш точну оцінку положення дрону. Він продемонстрував здатність значно знижувати вплив шумів і похибок, характерних для окремих сенсорів, а також створювати стабільну систему позиціонування, здатну ефективно працювати в динамічних умовах. EKF SLAM, у свою чергу, показав свою ефективність у побудові карти оточення та уточненні положення дрону шляхом використання орієнтирів, що дозволило підвищити точність локалізації навіть у складних умовах, таких як урбанізовані або природні середовища з великою кількістю статичних і динамічних перешкод.

Алгоритм A* дозволив знайти оптимальні маршрути до цілі, враховуючи обмеження середовища, такі як наявність перешкод і різних умов місцевості. Це стало важливим елементом автономного руху дрону, забезпечуючи мінімізацію енергозатрат та ефективну навігацію навіть у складних умовах. Крім того, використання алгоритмів штучного інтелекту, таких як YOLO, дозволило ідентифікувати об'єкти оточення, що додатково покращило якість навігації.

Попри досягнуті результати, важливим аспектом є інтеграція розглянутих алгоритмів у єдину систему автоматичної навігації, що залишилося поза межами цієї роботи. Наразі кожен із алгоритмів був протестований окремо, що продемонструвало їхні переваги та недоліки. Подальше дослідження повинно бути спрямоване на розробку цілісної архітектури, яка б об’єднувала всі ці елементи в єдину систему. Це забезпечило б можливість дрону одночасно здійснювати локалізацію, будувати карту оточення, знаходити оптимальний маршрут і уникати перешкод, що значно розширило б його функціональні можливості.

Варто зазначити, що результати тестувань підтвердили обґрунтованість обраного підходу. Алгоритми, розроблені й адаптовані в межах цієї роботи, чудово вписуються в концепцію автономної навігації дрону без широкого покладання на GPS. Це відкриває перспективи використання таких систем у різних сферах, зокрема в пошуково-рятувальних операціях, моніторингу навколишнього середовища, інспекції інфраструктури, а також у військових і логістичних задачах.

Таким чином, виконана робота є міцною основою для подальших досліджень і розробок у галузі автономної навігації дронів. Запропоновані алгоритми та проведені експерименти підтвердили свою ефективність, а їх потенційна інтеграція у комплексну систему навігації відкриває нові горизонти для розробки інноваційних рішень у цій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How does a GPS tracker work? Works in the car and mobile. *Seinxon*. URL: https://www.seinxon.com/de-eu/blogs/blogeintrage/how-does-a-gps-tracker-work (дата звернення: 02.11.2024).

2. What is GPS? | Garmin. *Offizielle Garmin Website & Webshop | Deutschland*. URL: https://www.garmin.com/en-US/aboutgps/ (дата звернення: 01.11.2024).

3. Inertial Measurement Unit (IMU) – An Introduction | Advanced Navigation. *Advanced Navigation*. URL: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/ (дата звернення: 02.11.2024).

4. IBM. What is Computer Vision? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/computer-vision (дата звернення: 02.11.2024).

5. What is Computer Vision? - Image recognition AI/ML Explained - AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/what-is/computer-vision/ (дата звернення: 03.11.2024).

6. Dai, Bo & He, Yu Qing & Yang, Liying & Su, Yun & Yue, Yufeng & Xu, Peter. (2020). SIMSF: A Scale Insensitive Multi-Sensor Fusion Framework for Unmanned Aerial Vehicles Based on Graph Optimization. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3000555.

7. Ito K. An Optimal Optical Flow. *SIAM Journal on Control and Optimization*. 2005. Т. 44, № 2. С. 728–742. URL: https://doi.org/10.1137/s0363012904433444 (дата звернення: 04.12.2024).

8. OpenCV: Optical Flow. *OpenCV documentation index*. URL: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html (дата звернення: 03.11.2024).

9. Estimating optical flow: A comprehensive review of the state of the art / A. Alfarano та ін. *Computer Vision and Image Understanding*. 2024. С. 104160. URL: https://doi.org/10.1016/j.cviu.2024.104160 (дата звернення: 04.12.2024).

10. Tchernykh V., Beck M., Janschek K. OPTICAL FLOW NAVIGATION FOR AN OUTDOOR UAV USING A WIDE ANGLE MONO CAMERA AND DEM MATCHING. *IFAC Proceedings Volumes*. 2006. Т. 39, № 16. С. 590–595. URL: https://doi.org/10.3182/20060912-3-de-2911.00103 (дата звернення: 04.12.2024).

11. Smith G. M. What is Sensor Fusion?. *Data Acquisition | Test and Measurement Solutions*. URL: https://dewesoft.com/blog/what-is-sensor-fusion (дата звернення: 04.11.2024).

12. Recent posts for: “Optical Flow SDK” | NVIDIA Technical Blog. *NVIDIA Technical Blog*. URL: https://developer.nvidia.com/blog/recent-posts/?products=Optical+Flow+SDK (дата звернення: 04.11.2024).

13. Urrea C., Agramonte R. Kalman Filter: Historical Overview and Review of Its Use in Robotics 60 Years after Its Creation. *Journal of Sensors*. 2021. Т. 2021. С. 1–21. URL: https://doi.org/10.1155/2021/9674015 (дата звернення: 04.12.2024).

14. McGee L. A., Schmidt S. F. Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry. Листоп. 1985 р. Меморандум. URL: https://ntrs.nasa.gov/api/citations/19860003843/downloads/19860003843.pdf (дата звернення: 10.05.2024).

15. Rudolf Kalman: The Creator of Kalman Filters and Modern Control Theory - News. *All About Circuits - Electrical Engineering & Electronics Community*. URL: https://www.allaboutcircuits.com/news/rudolf-kalman-the-creator-of-kalman-filters-modern-control-theory/ (дата звернення: 01.06.2024).

16. What Was the Apollo Program? (Grades 5-8) - NASA. *NASA*. URL: https://www.nasa.gov/learning-resources/for-kids-and-students/what-was-the-apollo-program-grades-5-8/ (дата звернення: 20.04.2024).

17. NASA. *NASA*. URL: https://www.nasa.gov/ (дата звернення: 05.11.2024).

18. Kalman Filter Applications. *Home | Department of Computer Science*. URL: https://www.cs.cornell.edu/courses/cs4758/2012sp/materials/MI63slides.pdf (дата звернення: 25.10.2024).

19. Understanding Kalman Filters, Part 3: Optimal State Estimator. *MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink*. URL: https://www.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html (дата звернення: 04.11.2024).

20. Zucconi A. The Mathematics of the Kalman Filter - Alan Zucconi. *Alan Zucconi*. URL: https://www.alanzucconi.com/2022/07/24/kalman-gain/ (дата звернення: 05.11.2024).

21. Online Kalman Filter Tutorial. *Kalman Filter Tutorial*. URL: https://www.kalmanfilter.net/default.aspx (дата звернення: 05.11.2024).

22. Franklin W. Kalman Filter Explained Simply - The Kalman Filter. *The Kalman Filter*. URL: https://thekalmanfilter.com/kalman-filter-explained-simply/ (дата звернення: 05.11.2024).

23. The Extended Kalman Filter - Alan Zucconi. *Alan Zucconi*. URL: https://www.alanzucconi.com/2022/07/24/extended-kalman-filter/ (дата звернення: 06.11.2024).

24. Franklin W. Kalman Filter Explained Simply - The Kalman Filter. *The Kalman Filter*. URL: https://thekalmanfilter.com/kalman-filter-explained-simply/ (дата звернення: 05.11.2024).

25. Satya. Sensor Fusion With Kalman Filter. *Medium*. URL: https://medium.com/@satya15july_11937/sensor-fusion-with-kalman-filter-c648d6ec2ec2 (дата звернення: 08.11.2024).

26. Consistency of the EKF-SLAM Algorithm / T. Bailey та ін. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, м. Beijing, 9–15 жовт. 2006 р. 2006. URL: https://doi.org/10.1109/iros.2006.281644 (дата звернення: 04.12.2024).

27. Example: Estimate 2-D Target States with Angle and Range Measurements Using trackingEKF. *MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink*. URL: https://www.mathworks.com/help/fusion/ug/extended-kalman-filters.html (дата звернення: 06.11.2024).

28. Przybylski S. The math behind Extended Kalman Filtering. *Medium*. URL: https://medium.com/@sasha_przybylski/the-math-behind-extended-kalman-filtering-0df981a87453 (дата звернення: 05.11.2024).

29. Holland M. Introduction to the Extended Kalman Filter. Nova Science Publishers, Incorporated, 2020.

30. Dead-Reckoning | Advanced Navigation. *Advanced Navigation*. URL: https://www.advancednavigation.com/glossary/dead-reckoning/ (дата звернення: 06.11.2024).

31. What is Dead reckoning navigation method ?. *RF Wireless World*. URL: https://www.rfwireless-world.com/Articles/Advantages-and-disadvantages-of-Dead-Reckoning-navigation-method.html (дата звернення: 07.11.2024).

32. What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink. *MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink*. URL: https://www.mathworks.com/discovery/slam.html (дата звернення: 07.11.2024).

33. McMinn E. Understanding SLAM in Robotics and Autonomous Vehicles. *Flyability's Confined Space Drone Makes Inspections Safer & Cheaper*. URL: https://www.flyability.com/blog/simultaneous-localization-and-mapping (дата звернення: 07.11.2024).

34. Bermudez L. Overview of SLAM. *Medium*. URL: https://medium.com/machinevision/overview-of-slam-50b7f49903b7 (дата звернення: 07.11.2024).

35. What is Visual SLAM Technology and What is it Used For?. *A3*. URL: https://www.automate.org/vision/blogs/what-is-visual-slam-technology-and-what-is-it-used-for (дата звернення: 07.11.2024).

36. Malik S. Lidar SLAM: The Ultimate Guide to Simultaneous Localization and Mapping. *Wevolver*. URL: https://www.wevolver.com/article/lidar-slam (дата звернення: 07.11.2024).

37. Monocular Visual-Inertial SLAM. *MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink*. URL: https://www.mathworks.com/help/vision/ug/monocular-visual-inertial-slam.html (дата звернення: 08.11.2024).

38. Zhu J., Li H., Zhang T. Camera, LiDAR, and IMU Based Multi-Sensor Fusion SLAM: A Survey. *Tsinghua Science and Technology*. 2024. Т. 29, № 2. С. 415–429. URL: https://doi.org/10.26599/tst.2023.9010010 (дата звернення: 04.12.2024).

39. Multi-Sensor SLAM for efficient Navigation of a Mobile Robot / M. S. A. Khan та ін. *2021 4th International Conference on Computing & Information Sciences*

(ICCIS), м. Karachi, Pakistan, 29–30 листоп. 2021 р. 2021. URL: https://doi.org/10.1109/iccis54243.2021.9676374 (дата звернення: 04.12.2024).

40. Robotics S. A review on ORB-SLAM-2 paper. *Medium*. URL: https://medium.com/@sallyrobotics.blog/a-review-on-orb-slam-2-paper-3554d4fcaa7c (дата звернення: 08.11.2024).

41. Kohler D., Hess W., Rapp H. Introducing Cartographer. *Google Open Source Blog*. URL: https://opensource.googleblog.com/2016/10/introducing-cartographer.html (дата звернення: 09.11.2024).

42. GitHub - ethz-asl/rovio. *GitHub*. URL: https://github.com/ethz-asl/rovio (дата звернення: 10.11.2024).

43. Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter / I. Ullah та ін. *Wireless Communications and Mobile Computing*. 2020. Т. 2020. С. 1–12. URL: https://doi.org/10.1155/2020/2138643 (дата звернення: 04.12.2024).

44. What Is Artificial Intelligence (AI)? | Google Cloud. *Google Cloud*. URL: https://cloud.google.com/learn/what-is-artificial-intelligence (дата звернення: 11.11.2024).

45. AI and Drones: Advancements and Applications in Unmanned Aerial Vehicles – visionplatform. *visionplatform*. URL: https://visionplatform.ai/artificial-intelligence-drones (дата звернення: 04.12.2024).

46. Bacon D. AI and the Future of Drone Technology | Toll Uncrewed Systems. *Toll Uncrewed Systems*. URL: https://tolluncrewedsystems.com/blog/ai-and-the-future-of-drone-technology/ (дата звернення: 04.12.2024).

47. UncLe-SLAM: Uncertainty Learning for Dense Neural SLAM / E. Sandström та ін. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, м. Paris, France, 2–6 жовт. 2023 р. 2023. URL: https://doi.org/10.1109/iccvw60793.2023.00488 (дата звернення: 04.12.2024).

48. How do AI robots work? Artificial intelligence and mobile robotics | Robotnik ®. *Robotnik*. URL: https://robotnik.eu/how-do-ai-robots-work-artificial-intelligence-and-mobile-robotics/ (дата звернення: 12.11.2024).

49. Integrated Extended Kalman Filter and Deep Learning Platform for Electric Vehicle Battery Health Prediction / D. C. Li та ін. *Applied Sciences*. 2024. Т. 14, № 11. С. 4354. URL: https://doi.org/10.3390/app14114354 (дата звернення: 04.12.2024).

50. Що таке навчання з підкріпленням? Розбираємо теорію і реальні кейси. *Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів*. URL: https://evergreens.com.ua/ua/articles/reinforcement-learning.html (дата звернення: 21.11.2024).

51. Machine Learning, ML. *IT-Enterprise – your one-stop platform for digital transformation | www.it.ua*. URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 29.11.2024).

52. Bootcamps C. C. O. What Is Machine Learning and How Does It Work?. *Medium*. URL: https://medium.com/@caltechbootcamps/https-www-simplilearn-com-

tutorials-machine-learning-tutorial-what-is-machine-learning-df7b91f10f6 (дата звернення: 22.11.2024).

53. IBM. What are Convolutional Neural Networks? | IBM. *IBM - United States*. URL: https://www.ibm.com/topics/convolutional-neural-networks (дата звернення: 22.11.2024).

54. GeeksforGeeks. Introduction to Convolution Neural Network - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/introduction-convolution-neural-network/ (дата звернення: 23.11.2024).

55. How To Perform Object Detection With CNNS. *EasyODM.tech*. URL: https://easyodm.tech/object-detection-with-cnns/ (дата звернення: 26.11.2024).

56. Give your product an Edge. *STMicroelectronics - STM32 AI*. URL: https://stm32ai.st.com/ (дата звернення: 29.11.2024).

57. Nano 33 BLE Sense Rev2. *Arduino Docs*. URL: https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2/ (date of access: 26.11.2024).

58. ESP32-S3 Wi-Fi & BLE 5 SoC | Espressif Systems. *Wireless SoCs, Software, Cloud and AIoT Solutions | Espressif Systems*. URL: https://www.espressif.com/en/products/socs/esp32-s3 (дата звернення: 28.11.2024).

59. How Does a UAV Navigation System Work? | TransiTiva. *TransiTiva*. URL: https://transitiva.com/knowledgebase/how-does-a-uav-navigation-system-work/ (дата звернення: 27.11.2024).

60. Autonomous Navigation. *MathWorks - Maker of MATLAB and Simulink - MATLAB & Simulink*. URL: https://www.mathworks.com/videos/series/autonomous-navigation.html (дата звернення: 28.11.2024).

61. Popowski S., Dąbrowski W. Individual Autonomous Navigation System. *Transactions on Aerospace Research*. 2017. Т. 2017, № 3. С. 84–106. URL: https://doi.org/10.2478/tar-2017-0023 (дата звернення: 04.12.2024).

62. Armah. IMPLEMENTATION OF AUTONOMOUS NAVIGATION ALGORITHMS ON TWO-WHEELED GROUND MOBILE ROBOT. *American Journal of Engineering and Applied Sciences*. 2014. Т. 7, № 1. С. 149–164. URL: https://doi.org/10.3844/ajeassp.2014.149.164 (дата звернення: 04.12.2024).

63. Sriram. Exploring Software Solutions for Training and Deploying Machine Learning Models on Drones. *Medium*. URL: https://medium.com/@cloudgeek/exploring-software-solutions-for-training-and-deploying-machine-learning-models-on-drones-a0b5d95d840d (дата звернення: 29.11.2024).

64. Rocha G. D. #101 Developing Autonomous Drones Using Python and AI. *Medium*. URL: https://medium.com/@genedarocha/101-developing-autonomous-drones-using-python-and-ai-3a85bc79a80b (дата звернення: 28.11.2024).

65. ROS 2 Documentation – ROS 2 Documentation: Foxy documentation. *ROS Documentation*. URL: https://docs.ros.org/en/foxy/index.html (дата звернення: 04.12.2024).

66. Home. *OpenCV*. URL: https://opencv.org/ (дата звернення: 29.11.2024).

67. Introduction to TensorFlow. *TensorFlow*. URL: https://www.tensorflow.org/learn (дата звернення: 29.11.2024).

68. PyTorch. *PyTorch*. URL: https://pytorch.org/ (дата звернення: 29.11.2024).

69. Matplotlib documentation – Matplotlib 3.9.3 documentation. *Matplotlib – Visualization with Python*. URL: https://matplotlib.org/stable/ (дата звернення: 26.11.2024).

70. Top 10 Tools for Developing Autonomous Drones. *Industry Wired*. URL: https://industrywired.com/top-10-tools-for-developing-autonomous-drones/ (дата звернення: 29.11.2024).

71. Best Robot Simulation Software. *Upcoming Engineer*. URL: https://upcomingengineer.com/best-robot-simulation-software/ (дата звернення: 30.11.2024).

72. Understanding the GUI – Gazebo ionic documentation. *Gazebo*. URL: https://gazebosim.org/docs/latest/gui/ (дата звернення: 28.11.2024).

73. Gazebo Transport: Python Support. *Gazebo*. URL: https://gazebosim.org/api/transport/14/python.html (дата звернення: 30.11.2024).

74. ROS 2 Architecture Overview. URL: https://automaticaddison.com/ros-2-architecture-overview/ (дата звернення: 27.11.2024).

75. The PID Controller & Theory Explained. *Mess- und Prüfsysteme, bei Emerson - NI*. URL: https://www.ni.com/en/shop/labview/pid-theory-explained.html (дата звернення: 30.11.2024).

76. RealPars. PID Controller Explained, 2021. *YouTube*. URL: https://www.youtube.com/watch?v=fv6dLTEvl74 (дата звернення: 30.11.2024).

77. Zulu A., John S. A Review of Control Algorithms for Autonomous Quadrotors. *Open Journal of Applied Sciences*. 2014. Т. 04, № 14. С. 547–556. URL: https://doi.org/10.4236/ojapps.2014.414053 (дата звернення: 04.12.2024).

78. GNSS-Denied Navigation Kit. *UAV Navigation*. URL: https://www.uavnavigation.com/products/navigation-systems/gnss-denied-navigation-kit (дата звернення: 23.11.2024).

79. Hansen M. Solutions for GNSS-Denied Simulation and Training. *Shaping the Future of Work | Kongsberg Digital*. URL: https://www.kongsbergdigital.com/resources/solutions-for-gnss-denied-simulation-and-training (дата звернення: 30.11.2024).

80. Inertial Measurement Unit (IMU) Use - ESE205 Wiki. *Home | WashU McKelvey School of Engineering*. URL: https://classes.engineering.wustl.edu/ese205/core/index.php?title=Inertial_Measurement_Unit_(IMU)_Use (дата звернення: 04.12.2024).

81. Blog - Dalvey. *Exceptional Design for Men: Gifts & Personalised Accessories - Dalvey*. URL: https://www.dalvey.com/in/blog/how-does-a-compass-work (дата звернення: 29.11.2024).

82. Event Cameras, Event camera SLAM, Event-based Vision, Event-based Camera, Event SLAM. *Robotics and Perception Group*. URL: https://rpg.ifi.uzh.ch/research_dvs.html (дата звернення: 28.11.2024).

83. What is an event-based camera?. *dioram*. URL: https://dioramslam.com/2021/10/08/event-cameras/ (дата звернення: 30.11.2024).

84. Terejanu G. Extended Kalman Filter Tutorial. 2009. URL: https://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf (дата звернення: 29.11.2024).

85. Libretexts. 3.8: Jacobians. *Mathematics LibreTexts*. URL: https://math.libretexts.org/Bookshelves/Calculus/Supplemental_Modules_(Calculus)/Vector_Calculus/3:_Multiple_Integrals/3.8:_Jacobians (дата звернення: 30.11.2024).

86. Saeed Faizi F., Khorsheed Alsulaifanie A. Visual-Based Simultaneous Localization and Mapping (VSLAM) Techniques for Robots: A Scientific Review. *Academic Journal of Nawroz University*. 2023. Т. 12, № 3. С. 213–229. URL: https://doi.org/10.25007/ajnu.v12n3a1500 (дата звернення: 05.12.2024).

87. A Brief Survey on SLAM Methods in Autonomous Vehicle / T. Takleh Omar Takleh та ін. *International Journal of Engineering & Technology*. 2018. Т. 7, № 4.27. С. 38. URL: https://doi.org/10.14419/ijet.v7i4.27.22477 (дата звернення: 05.12.2024).

88. GeeksforGeeks. A* Search Algorithm - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/a-search-algorithm/ (дата звернення: 29.11.2024).

89. Kumar R. The A* Algorithm: A Complete Guide. *Datacamp*. URL: https://www.datacamp.com/tutorial/a-star-algorithm (дата звернення: 30.11.2024).

90. A* Search Algorithm - 101 Computing. *101 Computing - Boost Your Programming Skills!*. URL: https://www.101computing.net/a-star-search-algorithm/ (дата звернення: 01.12.2024).

91. Medhi D. Real-time Object Detection with YOLO and Webcam: Enhancing Your Computer Vision Skills. *Medium*. URL: https://dipankarmedh1.medium.com/real-time-object-detection-with-yolo-and-webcam-enhancing-your-computer-vision-skills-861b97c78993 (дата звернення: 30.11.2024).

92. Rosebrock A. YOLO object detection with OpenCV - PyImageSearch. *PyImageSearch*. URL: https://pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/ (дата звернення: 30.11.2024).

93. Mirkhan A. YOLO Algorithm: Real-Time Object Detection from A to Z. *kili-website*. URL: https://kili-technology.com/data-labeling/machine-learning/yolo-algorithm-real-time-object-detection-from-a-to-z (дата звернення: 04.12.2024).

94. GitHub - nixonsd/drone-thesis. *GitHub*. URL: https://github.com/nixonsd/drone-thesis (дата звернення: 05.12.2024).

95. GitHub - nixonsd/detect-objects. *GitHub*. URL: https://github.com/nixonsd/detect-objects (дата звернення: 05.12.2024).

96. Jayant Dassz (@jayant_dassz) | Unsplash Photo Community. *Beautiful Free Images & Pictures | Unsplash*. URL: https://unsplash.com/@jayant_dassz (дата звернення: 25.08.2024).

97. I Hate Car People - Page 2 of 3 - MotoIQ. *MotoIQ*. URL: https://motoiq.com/i-hate-car-people/2/ (дата звернення: 26.08.2024).

98. Short-Range Community Improvement Projects. *AARP*. URL: https://www.aarp.org/livable-communities/tool-kits-resources/info-2015/13-short-range-livability-solutions.html (дата звернення: 26.08.2024).

99. GitHub - jacobhiggins/ekf_slam_demo. *GitHub*. URL: https://github.com/jacobhiggins/ekf_slam_demo (дата звернення: 03.12.2024).