НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем» зі спеціальності 121 “Інженерія програмного забезпечення” на тему: «Метод підвищення автономності об’єкту у 3D просторі»

Виконав:
студент ІІ курсу, групи ІМ-22мп
Федоров Даниїл Олександрович _____

Керівник:
проф., д.т.н., проф.
Стіренко Сергій Григорович _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор
Жабін Валерій Іванович _____

Рецензент:
доц. каф. СПіСКС, к.т.н., доц.,
Орлова Марія Миколаївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО
(підпис) (ініціали, прізвище)
« » 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Федорову Даниїлу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Метод підвищення автономності об'єкту у 3D просторі
Науковий керівник дисертації Стіренко Сергій Григорович, проф., д.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » жовтня 2023 р. №

2. Строк подання студентом дисертації листопад 2023

3. Об'єкт дослідження Метод підвищення автономності об'єкту у 3D просторі

4. Предмет дослідження модифікація та об'єднання існуючих моделей та рішень зі штучним інтелектом в одну систему, що підвищує автономність об'єктів у 3D просторі

5. Перелік завдань, які потрібно розробити: дослідити існуючі методи та підходи для підвищення автономності об’єкту у 3D просторі, порівняння виконаної системи з існуючими рішеннями

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Проф., д.т.н. Кулаков Ю.О.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Вивчення літератури	15.09.2023	
2	Написання вступної частини та огляд рішень	25.09.2023	
3	Аналіз	20.10.2023	
4	Моделювання розробленого способу	17.11.2023	
5	Оформлення документації ДП	25.11.2023	
6	Захист диплому		

Студент _____ __Федоров Д. О. _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Стіренко С.Г.
(підпис) (ініціали, прізвище)

Київ – 2023 року

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Метод підвищення автономності об’єкту у 3D просторі

студентом: Федоровим Даниїлом Олександровичем

Актуальність теми

Актуальність теми полягає у вирішенні важливої проблеми контролю та безпеки у використанні БПЛА в цивільних та військових сферах. У контексті швидкого розвитку технологій об’єктів у 3D просторі, робота критично аналізує існуючі методи управління повітряним простором, визначаючи необхідність автономних систем для підвищення безпеки та ефективності.

Мета і завдання дослідження

Мета дослідження – вдосконалення існуючих систем автономного керування та оцінка автономної системи управління об’єктом у 3D просторі з використанням штучного інтелекту. Для досягнення цієї мети передбачається реалізація таких завдань: інтеграція алгоритмів штучного інтелекту для ідентифікації та перехоплення інших об’єктів, адаптація існуючих моделей для конкретної задачі керування об’єктом у 3D просторі та зв’язування усіх моделей в одну систему керування.

Об'єкт дослідження - процес автономного управління об’єктами у 3D просторі з поставленою задачою.

Предмет дослідження - автономна система управління об’єктом, яка включає в себе алгоритми штучного інтелекту та методи їх застосування для ідентифікації та нейтралізації потенційних загроз.

Методи дослідження

Для досягнення поставлених в даній роботі задач використано методи моделювання середовища, методи аналізу великих даних та методи роботи зі штучним інтелектом з підсиленим навчанням.

Наукова новизна.

Дослідження набуло подальшого розвитку у сфері підвищення автономності об’єктів у 3D просторі. Було розроблено метод, що базується на контролі об’єктів з чотирма моторами у три вимірному середовищі. Цей підхід включає інтеграцію штучних інтелектів YOLOv8, MiDaS та контролюючого штучного інтелекту з глибоким підсиленим навчанням, та їх навчання за допомогою симуляції середовища, використовуючи бібліотеки OpenAI Gym та PyBullet. Це дозволяє підвищити продуктивність, швидкість, точність та надійність автономності об’єктів у 3D просторі.

Також розвинуто модель керування об’єктом у 3D просторі, яка використовує штучний інтелект з глибоким підсиленим навчанням. Вона забезпечена великою кількістю вхідних даних, що сприяє спрощенню системи контролю дронів за рахунок збільшення ітерацій навчання.

Практичне значення одержаних результатів

Результати дослідження мають практичне застосування у сфері цивільної та військової авіації, забезпечуючи більш ефективний контроль та безпеку використання БПЛА. Дослідження може бути використане для розробки нових, фізичних систем управління БПЛА.

Основний внесок магістранта

Магістерське дослідження є самостійно виконаною роботою в якій представлена модифікація існуючих систем та доповнення власними системами цільної програми контролю об’єктів у три вимірному просторі. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Практична цінність. Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- вдосконалення методів автономного контролю об’єктів у 3D просторі;
- модифікація методів розпізнавання об’єктів у польоті
- вирішення задачі автономності літаючих об’єктів

Апробація результатів дисертації

Для перевірки та підтвердження запропонованої у дисертації системи було обрано середовище для симуляції дронів gym-pybullet-drones, яке побудовано на базі середовищ Gym від OpenAI, які побудовані для спрощення роботи зі штучним інтелектом з підсиленим навчанням.

Обсяг роботи

Дисертація складається з 4 розділів загальним обсягом 122 сторінки. Містить 28 ілюстрацій, 23 таблиці. Використана інформація з 22 джерел літератури.

Ключові слова

Безпілотні літальні апарати, БПЛА, штучний інтелект, автономні системи управління, безпека повітряного простору, 3D простір, ідентифікація загроз, алгоритми штучного інтелекту, програмне забезпечення.

ABSTRACT

for the Master's Thesis

on the topic: Method for Enhancing Autonomy of an Object in 3D Space

by student: Fedorov Danyil

Relevance of the Topic

The relevance of this topic lies in addressing the significant problem of control and safety in the use of Unmanned Aerial Vehicles (UAVs) in both civilian and military spheres. In the context of rapid technology development in 3D space objects, this work critically analyzes existing airspace management methods, identifying the need for autonomous systems to enhance safety and efficiency.

Purpose and Objectives of the Research

The purpose of the research is to refine existing autonomous control systems and to evaluate an autonomous management system for an object in 3D space using artificial intelligence. To achieve this goal, the following tasks are envisaged: integration of artificial intelligence algorithms for identification and interception of other objects, adaptation of existing models for the specific task of controlling an object in 3D space, and linking all models into a single control system.

Research Object - the autonomous management of objects in 3D space with a set task.

Research Subject - the autonomous control system of an object, which includes artificial intelligence algorithms and their application methods for identifying and neutralizing potential threats.

Research Methods

To achieve the objectives set in this work, methods of environmental modeling, big data analysis techniques, and artificial intelligence methods with reinforced learning were utilized.

The scientific novelty

The research has further advanced in the field of enhancing the autonomy of objects in 3D space. A method based on the control of objects with four motors in a three-dimensional environment has been developed. This approach includes the

integration of YOLOv8, MiDaS, and controlling deep reinforcement learning artificial intelligences and their training through environment simulation using the OpenAI Gym and PyBullet libraries. This allows for improved productivity, speed, accuracy, and reliability of object autonomy in 3D space.

Additionally, a control model for an object in 3D space has been evolved, utilizing artificial intelligence with reinforced learning. It is equipped with a large volume of input data, which facilitates the simplification of drone control systems through an increased number of learning iterations.

Practical Significance of the Obtained Results

The results of the research have practical applications in the field of civilian and military aviation, providing more effective control and safety in the use of UAVs. The study can be used for the development of new physical control systems for UAVs.

Practical value

The obtained results can be used in future research in the following directions:

• Improvement of methods for autonomous control of objects in 3D space.

• Modification of object recognition methods during flight.

• Solving the problem of autonomy for flying objects.

Approval of dissertation results

To verify and confirm the proposed system in the dissertation, the environment for simulating drones gym-pybullet-drones was chosen, which is built on the basis of the Gym environment from OpenAI, designed to simplify working with artificial intelligence with reinforcement learning.

Work Volume

The thesis consists of 4 chapters with a total of 122 pages. It includes 28 illustrations, 23 tables, and utilizes information from 22 literature sources.

Keywords

Unmanned Aerial Vehicles, UAVs, artificial intelligence, autonomous control systems, airspace safety, 3D space, threat identification, artificial intelligence algorithms, software.

Пояснювальна записка

До магістерської дисертації

На тему «Метод підвищення автономності об’єкту у 3D просторі»

Київ - 2023

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

БПЛА (UAV)	(Unmanned Aerial Vehicle) Безпілотний літальний апарат
CNN	(Convolutional Neural Networks) Згорткові нейронні мережі
YOLO	(You Only Look Once) Ти дивишся тільки раз – state-of-the-art модель для розпізнавання об’єктів на зображеннях
ПН (RL)	(Reinforcement Learning) підсилене навчання
PPO	(Proximal Policy Optimization) Оптимізація проксимальної політики
OpenAI Gym	Python бібліотека для розробок у сфері штучного інтелекту з підсиленим навчанням
Gymnasium	Наслідник OpenAI Gym
RPM	(Rotations Per Minute) Обертів за хвилину
Plug-n-Play	«Ввімкни і грай» - готовий до використання продукт
MVP	(Minimum Viable Product) – Мінімально життєздатний продукт

ВСТУП

У сучасному світі спостерігається інтенсивне зростання використання безпілотних літальних апаратів (БПЛА) у цивільній та військовій сферах, розширюючи можливості від аерофотозйомки до екстрених реагувань на кризи та розваг. Однак, це також призводить до нових викликів: збільшення кількості БПЛА у повітряному просторі може призвести до небажаних інцидентів, порушень конфіденційності, а в критичних випадках - до безпекових загроз. Ці проблеми стають все більш актуальними в умовах глобалізації та технологічного прогресу, вимагаючи розробки нових підходів та рішень.

З одного боку, БПЛА надають неймовірні можливості для інновацій у багатьох областях, включаючи сільське господарство, геодезію, безпеку, та навіть доставку товарів. Вони можуть виконувати завдання, що раніше вимагали значних людських та фінансових ресурсів, з високою точністю та ефективністю. З іншого боку, поширення БПЛА викликає питання регулювання, повітряного контролю, конфіденційності та безпеки. Існують ризики, пов'язані з несанкціонованим використанням БПЛА, включаючи можливість використання для незаконного нагляду, контрабанди, а навіть тероризму.

У цьому контексті, розробка автономних систем для БПЛА, які можуть в реальному часі ідентифікувати та нейтралізувати потенційні загрози, стає життєво важливою. Це вимагає інтеграції передових технологій у галузі штучного інтелекту, машинного навчання, робототехніки та авіаційної техніки. Автономність у тривимірному просторі вимагає високоточних алгоритмів обробки даних, здатності до швидкісної адаптації в змінних умовах, а також забезпечення безпеки та надійності системи.

Ця дисертація фокусується на моделюванні та оцінці системи управління для автономних об’єктів у 3D просторі з використанням штучного інтелекту для ідентифікації та нейтралізації інших об’єктів. Це дослідження має на меті створення ефективних та безпечних механізмів захисту повітряного простору. Важливою частиною є використовування надійних алгоритмів, які дозволять

об’єктам самостійно визначати траєкторії, уникати перешкод та виконувати завдання з мінімальним втручанням людини.

Головна мета роботи полягає у вдосконаленні існуючих методів автономного управління БПЛА в тривимірному просторі, особливо зосереджуючись на здатності слідування за об'єктами у реальних умовах. Дослідження передбачає детальний аналіз, модифікацію, тестування та оцінку запропонованих методів, спрямованих на підвищення безпеки та ефективності використання об’єктів у три-вимірному просторі. Це включає розробку комплексних сценаріїв та моделювання, що дозволять глибше зрозуміти та вдосконалити взаємодію БПЛА з оточуючим середовищем, а також впровадження інноваційних рішень для забезпечення безпеки в умовах складних та непередбачуваних ситуацій.

РОЗДІЛ 1

ІСНУЮЧІ РІШЕННЯ ДЛЯ ЗБІЛЬШЕННЯ АВТОНОМНОСТІ ОБ’ЄКТІВ

У зверненні до нагальної необхідності покращення автономних операцій у тривимірних просторах, зокрема у сфері безпілотних літальних апаратів (БПЛА), є імперативом ретельний огляд існуючих методологій та технологічних рішень. Першочергова мета полягає у підсиленні автономії об'єктів, що навігують через такі простори, на прикладі дрону, з задачою автономного нейтралізування інших рухаючихся цілей у повітрі. Цей розділ розпочинається з ретельного аналізу поширених рішень, критично оцінюючи їхні переваги, недоліки та еволюційний шлях, який вони пройшли протягом років.

Обговорення розгортатиметься структуровано, починаючи з проникливої оцінки алгоритму YOLOv8, ключового елементу у сфері виявлення та розпізнавання об'єктів. Це відбуватиметься без вичерпного архітектурного аналізу, з огляду на широке розуміння у науковому співтоваристві, але зосередиться на основних принципах, прогресі з часом та на протиставленні сильних та слабких сторін. Далі буде здійснено аналітичний огляд алгоритму MiDaS, дотримуючись подібної оціночної структури, що дасть змогу глибше зрозуміти його оперативні принципи та місце серед різноманітних технік оцінки глибини трьох кольорового зображення.

Після цього увага переміститься до емпіричного огляду існуючих рішень, що стосуються дронів, керованих штучним інтелектом. Ця частина має на меті надати всебічне уявлення про сучасний стан розвитку, роз'яснюючи механізми, за допомогою яких штучний інтелект використовується для збільшення автономії та оперативної ефективності дронів.

Завершуючи розділ, буде сформульовано синтез отриманих знань, надаючи зрозуміле узагальнення про стан та потенційний шлях технологічних досягнень у посиленні автономії об'єктів у тривимірних просторах. Це структуроване викладення задумане як міцна основа для наступних розділів з формулювання задачі та впровадження її.

1.1 Розпізнавання об’єктів на зображенні з використанням YOLOv8

Серія YOLO (You Only Look Once) зробила значний внесок у сферу виявлення об’єктів у реальному часі [1]. В минулому вчені опублікували декілька YOLO версій, тяких як YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, також існують перероблені спрощені версії, наприклад YOLO-Lite. На рисунку 1.1 приведено графічне порівняння різних моделей YOLO.

YOLOv8 - це найновіша версія YOLO від Ultralytics. Як передова модель, яка є state-of-the-art сучасних технологій, YOLOv8 побудована на успіху попередніх версій, вводячи нові функції та вдосконалення для підвищення продуктивності, гнучкості та ефективності. YOLOv8 підтримує повний спектр завдань штучного інтелекту в галузі зору, включаючи виявлення, сегментацію, оцінку пози, відстеження та класифікацію. Ця універсальність дозволяє користувачам використовувати можливості YOLOv8 у різноманітних застосунках та доменах [2].

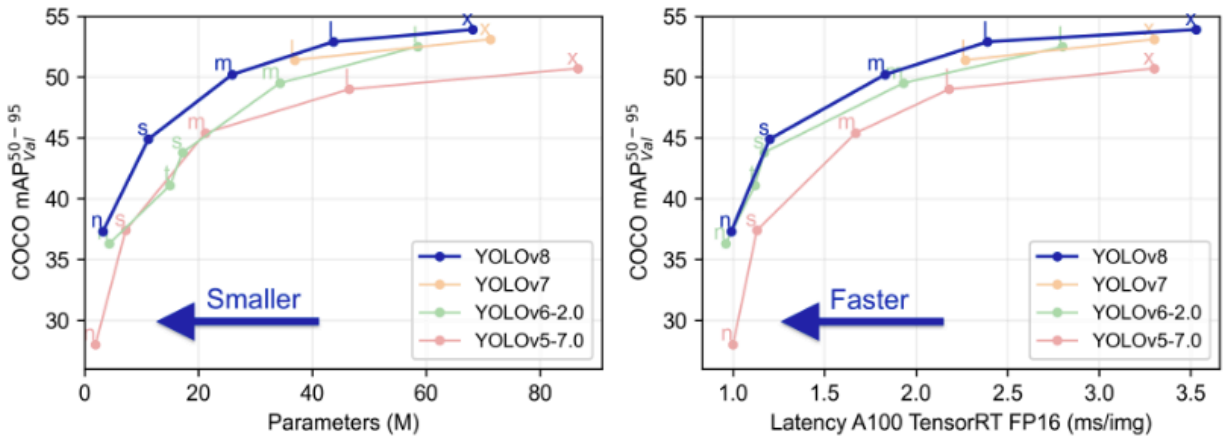

Рисунок. 1.1. Порівняння різних версій алгоритму YOLO [6]

1.1.1 Короткий опис архітектури та алгоритму YOLO, переваги та недоліки

YOLO має 24 згорткові шари (рисунок 1.2.), за якими слідують 2 повнозв'язних шари. Чергуючи 1 × 1 згорткові шари зменшують простір ознак з попередніх шарів. Згорткові шари попередньо треновані на задачі класифікації ImageNet при половині роздільності (вхідне зображення 224 × 224) а потім

подвоює роздільність для виявлення. Принципе роботу зображений на рисунку 1.3.

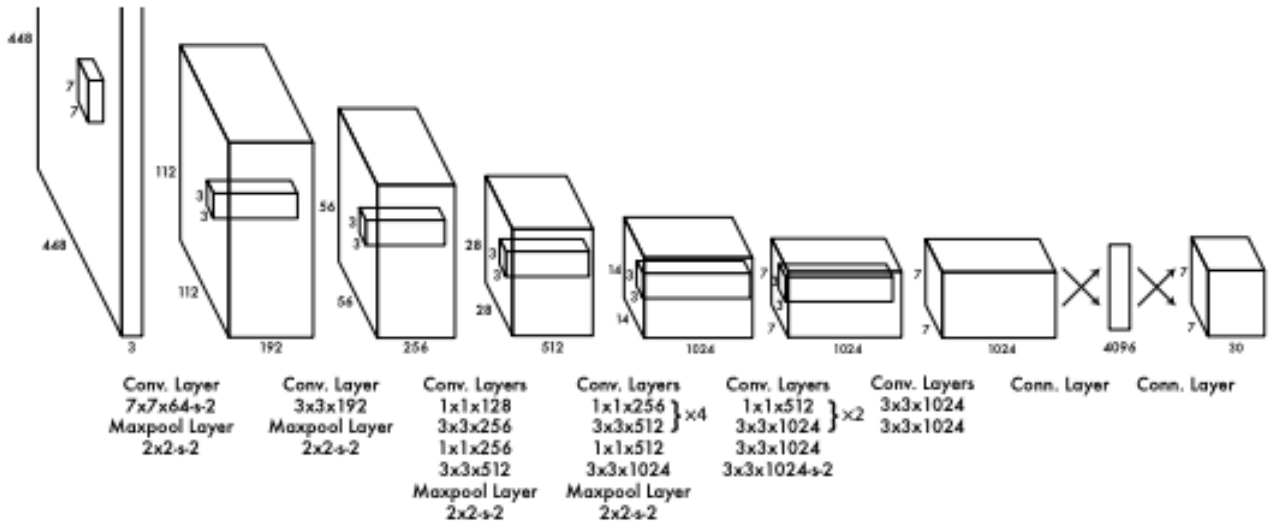

Рисунок. 1.2. Архітектура YOLO [3]

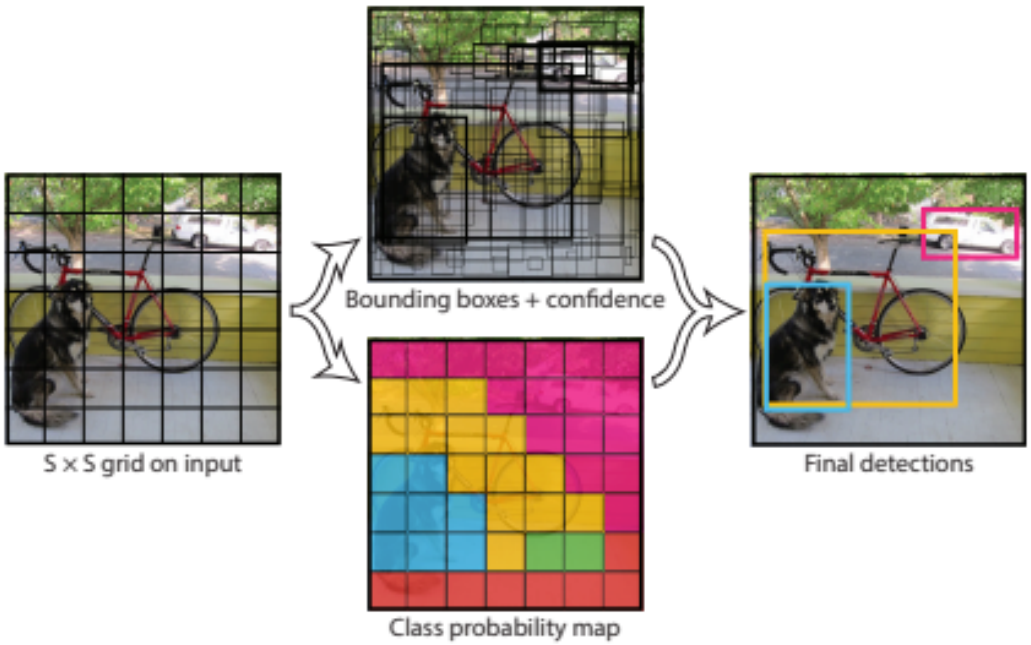

Рисунок. 1.3. Приклад роботи моделі YOLO [5]

Переваги:

- Швидкість виявлення: Модель YOLO v8 забезпечує високу швидкість обробки зображень, що є критично важливим для реального часу автономних БПЛА.
- Точність: Версія v8 продемонструвала покращення у точності в порівнянні з попередніми версіями, що робить її більш надійною для виявлення різноманітних об'єктів.

- Універсальність: YOLO здатний виявляти широкий спектр об'єктів, що дозволяє застосовувати його у різних сценаріях.
- Можливості налаштування: Система YOLO дозволяє налаштувати модель під конкретні задачі, що може підвищити ефективність виявлення у специфічних умовах.

Недоліки:

- Складність тренування: Для досягнення оптимальної точності, YOLO v8 вимагає великих об'ємів даних та значних ресурсів для тренування.
- Обмеження у виявленні дрібних об'єктів: У деяких випадках YOLO може мати труднощі з виявленням дуже маленьких або віддалених об'єктів.
- Вимоги до обчислювальних ресурсів: Висока обчислювальна потужність необхідна для підтримки швидкості та точності YOLO v8, що може бути проблематичним для деяких мобільних систем.
- Чутливість до умов оточення: Точність YOLO v8 може знижуватися у складних або змінних умовах освітлення та погоди.

Високої популярності модель YOLO змогла досягти також через продуману формулу оцінки впевненості [4] (формула 1.1)

$$Pr(Class_i|Object) * Pr(Object) * IOU\frac{truth}{pred} = Pr(Class_i) * IOU\frac{truth}{pred} \quad (1.1)$$

Формат прогнозу YOLO для одного класу:

- x – координата центру оточуючої рамки по осі x
- y – координата центру оточуючої рамки по осі y
- width – ширина оточуючої рамки
- heigh – ширина оточуючої рамки
- confidence score – значення впевненості моделі в розпізнаному об’єкті

1.2 SSD – Single Shot MultiBox Detector

Метод розпізнавання об’єктів на зображеннях використовуючи єдину глибоку нейронну мережу. Автори Вei Лью та ін. даного методу за головну перевагу виділяють логіку систему без повторної дискретизації пікселів або функцій для гіпотез обмежувальної рамки [16]. Архітектура моделі показана на рисунку 1.4.

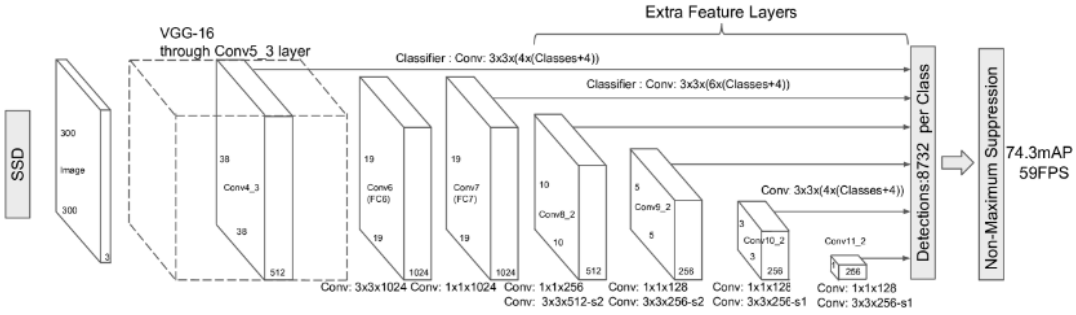

Рисунок 1.4 Архітектура моделі SSD [16]

Переваги SSD в порівнянні з YOLO:

- Краща точність виявлення: SSD зазвичай демонструє кращу точність виявлення малих об'єктів завдяки використанню декількох розмірів фільтрів на різних рівнях.
- Гнучкість у виборі базової мережі: SSD може бути ефективно інтегрований з різними архітектурами сверточних нейронних мереж.

Недоліки SSD в порівнянні з YOLO:

- Складність архітектури: SSD має складнішу архітектуру порівняно з YOLO, що може ускладнити розуміння та реалізацію.
- Швидкість обробки: У деяких випадках YOLO може перевершувати SSD за швидкістю обробки, особливо в моделях нових версій YOLO.
- Складність налаштування: SSD вимагає більш детального налаштування параметрів, таких як розміри та аспектні співвідношення "default boxes", що може бути складним і часомістким.
- Чутливість до розміру об'єктів: Хоча SSD краще виявляє малі об'єкти, він може бути більш чутливим до різниці у розмірах об'єктів, що може вимагати додаткової настройки або попередньої обробки даних.

- Обмеження в швидкості роботи з великими датасетами: У деяких випадках, при роботі з великими датасетами, YOLO може переважати SSD за швидкістю через свою оптимізовану обробку даних.

1.3 Faster R-CNN

Faster R-CNN є моделлю виявлення об'єктів, що базується на глибоких згорткових мережах. Вона включає Мережу Подання Регіонів (Region Proposal Network, RPN) та Мережу Виявлення Об'єктів, які обидві навчаються для спільного використання згорткових шарів для швидкого тестування. RPN - це повністю згорткові мережі, що одночасно прогнозують межі об'єктів та оцінки "об'єктності" на кожній позиції. RPN навчаються від початку до кінця для генерації високоякісних пропозицій регіонів, які використовуються Fast R-CNN для виявлення [17].

Ця робота представляє RPN, яка ділить повноекранні згорткові характеристики з мережею виявлення, тим самим забезпечуючи майже безкоштовні пропозиції регіонів і подальше об'єднання RPN та Fast R-CNN в одну мережу шляхом спільного використання їхніх згорткових характеристик [17].

Основні ідеї Faster R-CNN полягають у запропонуванні RPN для генерації пропозицій регіонів, яка є швидшою і точнішою, ніж Selective Search, та використанні ітеративного спільного методу навчання для навчання RPN та Fast R-CNN [17]. Схема роботи моделі представлена на рисунку 1.5.

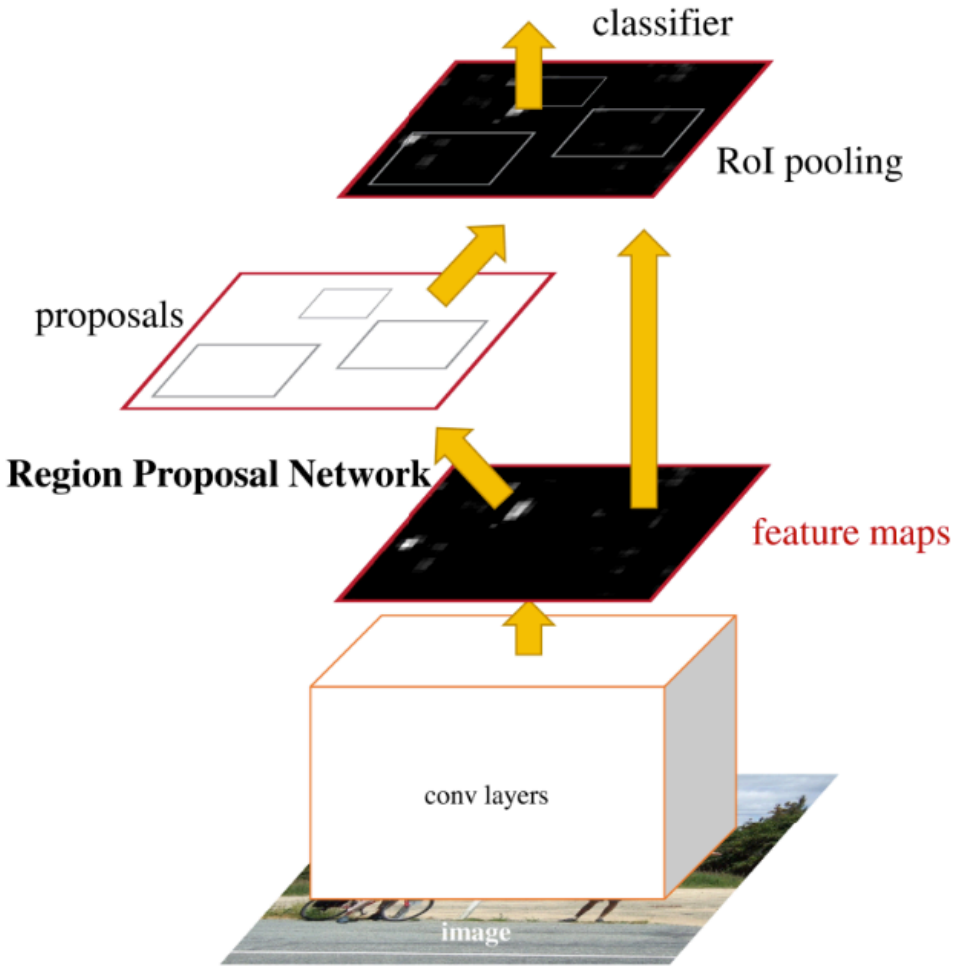

Рисунок 1.5 Схема роботи Faster R-CNN [17]

Переваги Faster R-CNN порівняно з YOLO:

- Вища точність: Faster R-CNN, як правило, демонструє вищу точність у виявленні об'єктів, оскільки він використовує механізм регіональних пропозицій, що дозволяє краще локалізувати об'єкти.
- Більш детальне виявлення: Завдяки механізму регіональних пропозицій Faster R-CNN здатен більш точно визначати розташування та розміри об'єктів, що є важливим у деяких застосуваннях, таких як медичне зображення.
- Гнучкість у роботі з різними масштабами об'єктів: Faster R-CNN ефективніше справляється з об'єктами різних розмірів завдяки використанню анкорів (anchors) різних масштабів та пропорцій.

- Краще навчання на складних датасетах: Faster R-CNN часто показує кращі результати на складних датасетах з великою кількістю класів або детальними анотаціями.

Недоліки Faster R-CNN порівняно з YOLO:

- Нижча швидкість виявлення: Один з ключових недоліків Faster R-CNN - це його порівняно низька швидкість обробки зображень порівняно з YOLO, що обмежує його використання в реальному часі.
- Більш складна архітектура: Архітектура Faster R-CNN є більш складною, що ускладнює її впровадження та оптимізацію.
- Вищі вимоги до обчислювальних ресурсів: Для тренування та використання Faster R-CNN потрібні більш потужні обчислювальні ресурси, що може бути проблематичним для обмежених або мобільних середовищ.
- Складність у тренуванні: Процес тренування моделі Faster R-CNN може бути більш складним та часомістким через необхідність налаштування більшої кількості гіперпараметрів порівняно з YOLO.

1.4 Визначення глибини кольорового зображення з використанням MiDaS

Характеристика глибини зображення є одною з найважливіших репрезентацій для дій у фізичних середовищах [7]. Незважаючи на розвиток технологій – оцінка глибини по одному зображенню все ще є дуже важкою задачою, яка потребує використання численних візуальних підказок, а також контекст зображення на великій відстані та попередньо отримані знання. Це вимагає застосування методів на основі навчання [8]. Як і для будь-яких інших типів штучного інтелекту – для отримання одночасно правильного та ефективного результату для різноманітних сценаріїв, вимагається рівно такий самий різноманітний та великий набір даних для тренування ШІ. Це і є основний виклик – отримання таких даних. Одним з передових методів отримання набору даних для навчання, запропонованим розробниками MiDaS є збір інформації з

3D фільмів (рисунок 1.6.). Незважаючи на те, що такі фільми не дають метричну глибину, є можливість використання стерео-збігу для отримання приблизної глибини. У процесі навчання – на вхід до алгоритму подається зображення з фільму і його зворотню мапу глибини.

Рисунок. 1.6 Приклад використання кадрів з 3D фільмів і їх мап глибини для навчання MiDaS [8]

Загалом, для навчання і тестування MiDaS використовували велику кількість датасетів (таблиця 1.1.), оскільки немає єдиного датасету, який задовольняв би усі потреби і також був представлений великим набором різних сценаріїв.

Dataset	Indoor	Outdoor	Dynamic	Video	Dense	Accuracy	Diversity	Annotation	Depth	# Images
DIML Indoor [31]	✓			✓	✓	Medium	Medium	RGB-D	**Metric**	220K
MegaDepth [11]		✓	(✓)		(✓)	Medium	Medium	SfM	No scale	130K
ReDWeb [32]	✓	✓	✓		✓	Medium	**High**	Stereo	No scale & shift	3600
WSVD [33]	✓	✓	✓	✓	✓	Medium	**High**	Stereo	No scale & shift	1.5M
3D Movies	✓	✓	✓	✓	✓	Medium	**High**	Stereo	No scale & shift	75K
DIW [34]	✓	✓	✓			Low	**High**	User clicks	Ordinal pair	496K
ETH3D [35]	✓	✓			✓	**High**	Low	Laser	**Metric**	454
Sintel [36]	✓	✓	✓	✓	✓	**High**	Medium	Synthetic	(Metric)	1064
KITTI [28], [29]		✓	(✓)	✓	(✓)	Medium	Low	Laser/Stereo	**Metric**	93K
NYUDv2 [30]	✓		(✓)	✓	✓	Medium	Low	RGB-D	**Metric**	407K
TUM-RGBD [37]	✓		(✓)	✓	✓	Medium	Low	RGB-D	**Metric**	80K

Таблиця. 1.1. Датасети, використані в тренуванні та тестуванні MiDaS [11]

Одна з проблем використання різних датасетів – різні показники глибини. Наприклад, у датасеті DIML Indoor – глибина представлена в метрах, коли в інших датасетах – у відносних величинах. Це змусило розробників винайти свою функцію втрат. Після тестувань багатьох формул на різних типах вхідних даних – було винайдено остаточну формулу (формула 1.2), з використанням відповідного градієнтного визначника [11].

$$L_{reg}(\hat{d}, \hat{d}^*) = \frac{1}{M}\sum_{k=1}^{K}\sum_{i=1}^{M}(\left|\Delta_x R_i^k\right| + \left|\Delta_y R_i^k\right|), \quad (1.2)$$

де $R^i = \widehat{d_i} - \hat{d}_i^*$, та R^k позначають різницю невідповідності карти в масштабі k. Використовують K = 4 рівня масштаб, зменшуючи зображення вдвічі на кожному рівні [11]. Остаточна формула приведена у формулі 1.3.

$$L_l = \frac{1}{N_l}\sum_{n=1}^{N_l} L_{ssi}\left(\hat{d}^n, \left(\hat{d}^*\right)^n\right) + \alpha L_{reg}\left(\hat{d}^n, \left(\hat{d}^*\right)^n\right) \quad (1.3)$$

Завдяки усьому об’єму роботи розробників, зараз ми можемо використовувати модель MiDaS, яка в порівнянні з іншими системами виглядає набагато краще і більш детально.

Порівняння MiDaS з аналогами показано на рисунку 1.7 [11].

Рисунок. 1.7 Порівняння MiDaS (останній рядок) з іншими схожими алгоритмами

1.5 ШІ з глибоким підкріпленим навчанням для керування БПЛА

Виникнення автономних безпілотних літальних апаратів (БПЛА) стало значним віхою в області повітряної робототехніки. У сучасному контексті критична важливість збільшення автономності об'єктів у тривимірному просторі підкреслена зростаючою складністю завдань, що відводяться дронам, особливо в умовах, які є небезпечними або недоступними для людей. Ключовим елементом підвищення автономності БПЛА є інтеграція штучного інтелекту (ШІ), з особливим акцентом на глибоке підсилене навчання (ГПН). Ця інтеграція представляє собою парадигмальний зсув від ручного керування до інтелектуальної автоматизації, що вирішує декілька ключових проблем, притаманних ручній експлуатації БПЛА.

Однією з фундаментальних переваг використання ШІ у системах керування БПЛА є суттєве зниження затримок. При ручному керуванні зв'язок

між вхідними командами пілота та реакцією БПЛА може бути згубним, особливо в критичних сценаріях, де важлива кожна мікросекунда. Системи на базі ШІ, підкріплені алгоритмами ГПН, пристосовані до обробки та реагування на динамічні змінні довкілля з швидкістю та точністю, якій не може дорівнятися непідготовлене людське керування. Цей аспект є критичним при розгляді завдань, таких як перехоплення та нейтралізація ворожих об'єктів, де час для ефективної відповіді надзвичайно короткий.

Крім того, ШІ-керовані БПЛА перевершують людські оперативні обмеження, виконуючи складні маневри та приймаючи автономні рішення на основі аналізу даних у реальному часі. Ця здатність є ключовою в військових операціях, де середовище не тільки небезпечне, але й може бути непередбачуваним та повним несподіваних змінних. Використовуючи ГПН, БПЛА можуть вчитися взаємодіяти з навколишнім середовищем, адаптуючи свої стратегії керування для максимізації оперативної ефективності без прямого людського нагляду.

Додатково, розгортання ШІ у БПЛА вирішує проблему людської вразливості в військових контекстах. Дистанційно керовані дрони, хоча і зменшують ризик для пілотів-людей, все ще вимагають операторів, які знаходяться в зоні ризику, чи то на передових базах, чи у радіусі дії контрзасобів. Повністю автономні дрони усувають потребу у втручанні людей на близькій відстані, таким чином знижуючи ризик для персоналу та дозволяючи проводити операції в умовах, що вважаються надто небезпечними для людської присутності.

Більш того, системи на базі ШІ не підпадають під фізіологічні обмеження, з якими стикаються людські пілоти. Проблеми, такі як когнітивна втома, зниження концентрації уваги та вплив екстремального стресу, не перешкоджають системі ШІ, яка працює з непохитною послідовністю. Ця надійність особливо важлива під час тривалих місій, які вимагають постійної уваги, загальнопоширений сценарій у розвідці або в операціях постійної активності.

1.5.1 Автономна навігація БПЛА у середовищі без GPS

Розробка вчених даної технології ставлять за мету вдосконалити автономність БПЛА у приміщеннях, де слабкий або відсутній зв'язок GPS. Архітектура являє собою збірку згорткових нейронних мереж з підсиленим навчанням (рисунок 1.8., рисунок 1.9. та рисунок 1.10.).

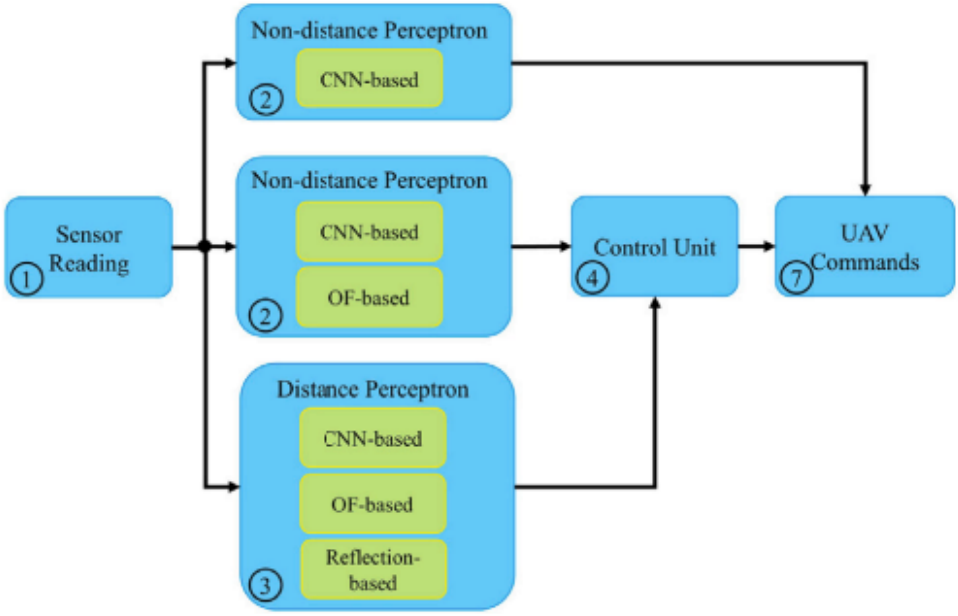

Рисунок. 1.8. Архітектура ШІ для навігації БПЛА без мапи [9]

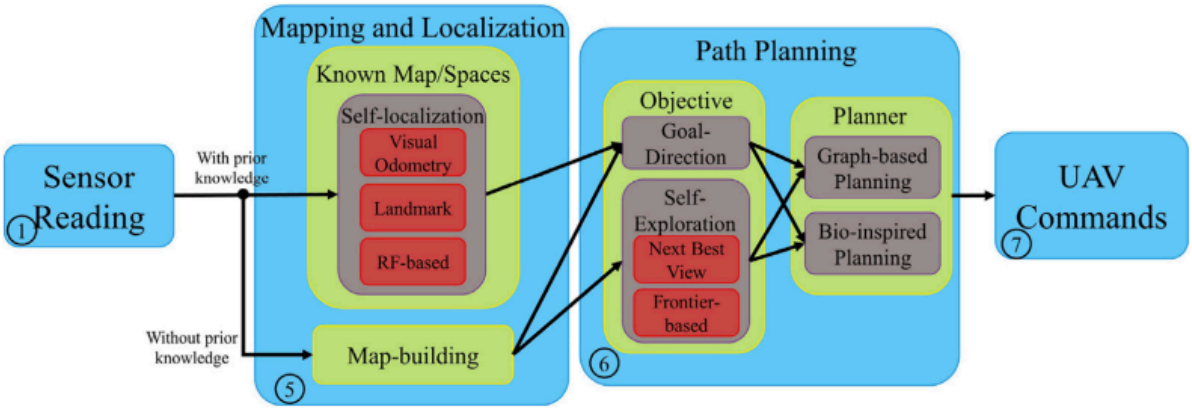

Рисунок. 1.9. Архітектура ШІ для навігації БПЛА з мапою [9]

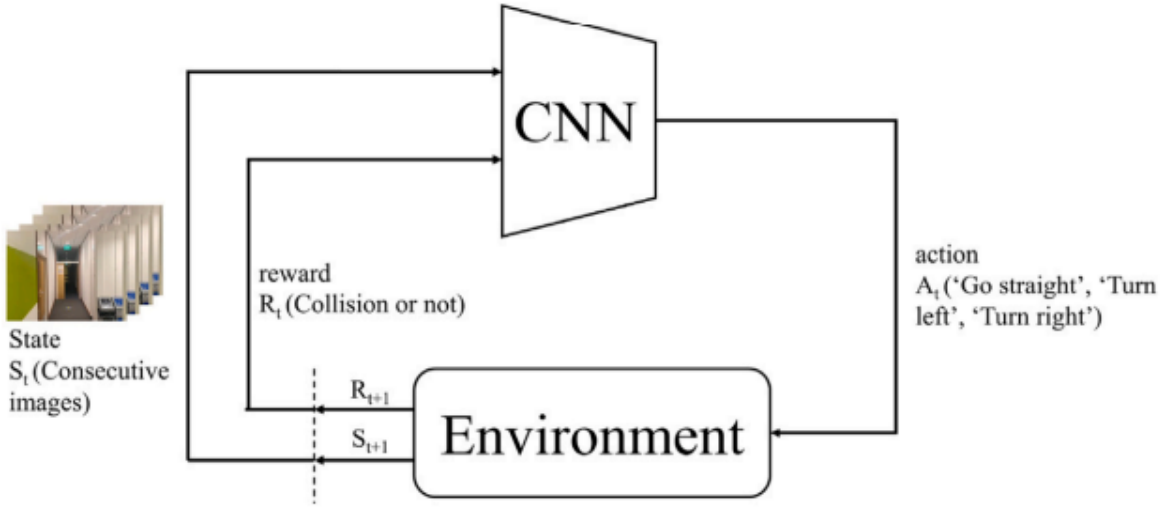

Рисунок. 1.10 Архітектура підсиленого навчання згорткової мережі [9]

Запропонована І. Чангом та ін. система пропонує контроль БПЛА як без мапи так і з мапою. Згідно висновків, у випадку користування ШІ без мапи – рано чи пізно БПЛА стикнеться з перешкодою, оскільки не задано ні початок, ні кінець маршруту, отже керування з мапою є найбільш оптимальне у випадку, описаному в роботі [9].

1.5.2 Використання штучного інтелекту для навігації БПЛА

У своїй роботі С. Режван та В. Шоі представляють дослідження, по використанню різних типів ШІ та різних методів оптимізації.

Метод Оптимізації Роєм Часток (ОРЧ) продемонстрував значний потенціал у сфері автономної навігації, зокрема у трирозмірному оперативному театрі безпілотних літальних апаратів (БПЛА). Парадигма інтелекту рою, на якій ґрунтується ОРЧ, забезпечує міцний каркас для навігації складними середовищами шляхом імітації колективної поведінки, спостережуваної у природних системах. У випадку БПЛА, кожен дрон можна розглядати як автономного агента або "частку", яка ітеративно коригує свій траєкторію на основі спільної навігаційної інформації – оптимізуючи як індивідуальні, так і колективні польотні шляхи до визначеної мети [10].

Застосування ОРЧ у БПЛА перевищує просто шляхопрокладання; воно було адаптоване та удосконалене для врахування динамічних змінних, властивих реальним сценаріям. Модифікації, такі як Модифікований ОРЧ (МОРЧ) та

Дискретний ОРЧ (ДОРЧ), були введені для вирішення конкретних завдань, як-от уникнення перешкод та обчислювальних складнощів планування маршруту у дискретизованому повітряному просторі. Ці зміни уособлюють еволюцію від стандартного ОРЧ, включаючи механізми, які використовують фактори помилки для перетворення непрактичних траєкторій на виконувальні маневри, тим самим забезпечуючи навігаційну безпеку та ефективність.

Ефективність МОРЧ демонструється через його здатність до переорієнтації БПЛА при виявленні потенційних зіткнень, динамічно коригуючи польотні траєкторії для підтримання безперервності місії. ДОРЧ, з іншого боку, застосовує принципи ОРЧ у контексті оптимізації дискретного простору, перекладаючи безперервний простір пошуку у проблему, аналогічну задачі комівояжера (ЗК), та відтак використовуючи встановлені техніки оптимізації для удосконалення вибору шляхів.

Такі прогреси, як ОРЧ на основі стратегії конкуренції (СОРЧ), пропонують подальше ускладнення, інтегруючи процеси конкурентного відбору, які дозволяють ідентифікувати найбільш придатні маршрути з набору можливих альтернатив. Такий підхід є індикатором ширшого напрямку до методів еволюційного обчислення, де відбіркові тиски керують емерджентним інтелектом до оптимальних рішень.

Узагальнені результати цих досліджень вказують, що ОРЧ, особливо у його вдосконалених формах, є високоефективною стратегією для підвищення автономності об'єктів у трирозмірному просторі. Його пристосовність до нюансів навігації БПЛА та здатність вирішувати обчислювальні завдання реального часу роблять його потужним інструментом для сценаріїв, що вимагають високого рівня автономності та точності. Розвиток цих методів демонструє потенціал ОРЧ значно підвищити продуктивність БПЛА, особливо в застосуваннях, де критично важливі автономне перехоплення та нейтралізація повітряних загроз. Отже, алгоритми на основі ОРЧ обіцяють встановити нові стандарти в галузі технології автономних дронів, пропонуючи поєднання

адаптивності, надійності та стратегічної складності, які відповідають вимогам сучасних повітряних операцій.

Застосування *оптимізації мурашиними колоніями (ОМК)* у контексті підвищення автономності об'єктів у тривимірному просторі, зокрема для безпілотних літальних апаратів (БЛА), становить значний прорив на шляху розв'язання складних проблем навігації та залучення цілей. Цей біологічно натхненний алгоритм, що віддзеркалює шляхи здобування їжі мурахами, було вміло реінтерпретовано для підвищення оперативної ефективності дронів та тактичних можливостей прийняття рішень у протистоянні з іншими повітряними загрозами.

Стратегічне розгортання ОМК у системах дронів сприяє створенню динамічної та міцної навігаційної структури, яка імітує децентралізований інтелект мурашиних колоній. Через застосування феромоново-залежного вибору та підсилення шляхів, БЛА наділяються кооперативною стратегією, яка поступово удосконалює їхні льотні маршрути, уникаючи перешкод та водночас точно націлюючись на цілі. Стохастичний, проте структурований характер дослідження, властивий ОМК, забезпечує всебічний пошук у повітряному просторі, що призводить до швидкого виявлення та перехоплення ворожих дронів.

Адаптації, такі як багатоколоніальний та подвійноколоніальний ОМК, були ключовими у подоланні передчасної конвергенції, поширеного обмеження, коли процес пошуку застоюється на локальних оптимумах. Ці розвинені ітерації сприяють міжколоніальному зв'язку та співпраці, тим самим диверсифікуючи шаблони пошуку та покращуючи якість рішень. Інтеграція генетичних алгоритмів (ГА) з ОМК, як це продемонстровано Гуаном та ін., додатково зміцнила механізм феромонів, посилюючи еволюційний аспект оптимізації маршрутів.

Синтез ОМК з штучним потенційним полем (ШПП), запропонований Джином та ін., є свідченням гнучкості методу та його здатності до синергетичного покращення. Використовуючи відштовхувальні сили ШПП для

уникнення перешкод та його притягальні сили для захоплення цілей, результуючий алгоритм потенційного поля ОМК (ПФОМК) не лише обходить ризики передчасної конвергенції, але й забезпечує баланс між швидкісним знаходженням шляху та уникненням зіткнень.

По суті, застосування ОМК до БПЛА відзначає трансформаційний прогрес у пошуку повністю автономних дронів, здатних нейтралізувати повітряні загрози (рисунок 1.11.). Ітеративна природа алгоритму, разом зі стратегічними удосконаленнями, довела свою ефективність у різних симуляційних сценаріях. Емерджентна поведінка, що виникає з дотримання окремими БЛА простих правил для досягнення складних цілей, відповідає глобальній меті підвищеної автономії у тривимірному просторі, що вказує на високообіцяючий напрямок для подальших досліджень та практичного впровадження [10].

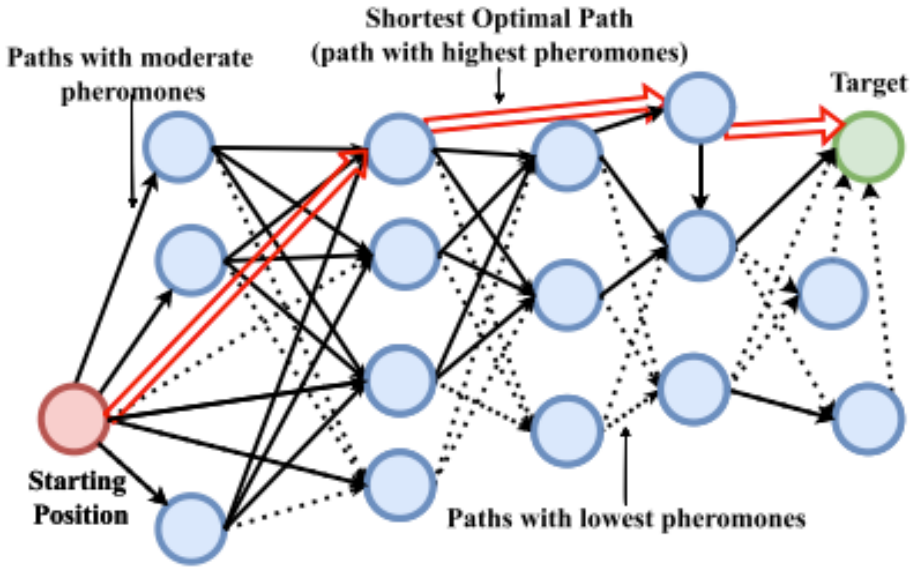

Рисунок. 1.11. Схема оптимізації мурашиними колоніями (ОМК) [10]

Генетичний алгоритм (ГА) виступає як надійний метод для підвищення автономності БПЛА у складних 3D середовищах. Його стохастичний характер оптимізації дозволяє синтезувати ефективні траєкторії польоту шляхом імітації еволюційних процесів. Кодуючи траєкторію у структуру, подібну до хромосоми, ГА ітеративно удосконалює потенційні рішення, використовуючи генетичні операції, такі як кросинговер, мутація та відбір. Ці операції сприяють дослідженню простору рішень та використанню рішень з високою

пристосованістю, збігаючись на траєкторії, яка мінімізує попередньо визначену функцію пристосованості. Ця функція, зазвичай, узагальнює ефективність шляху, відповідність до динаміки БПЛА та обмеження навколишнього середовища, забезпечуючи, щоб результативний шлях був не лише коротким, але й відповідав можливостям польоту та був безпечним від перешкод.

Сила ГА полягає у його адаптивності та різноманітності пошуку, базованого на популяції, який забезпечує стійкість проти локальних оптимумів – критичного атрибуту у складних, багатовимірних просторах (рисунок 1.12.). Література вказує на вдосконалення стандартного ГА, вводячи ієрархічні, рекурсивні та хаотичні елементи пошуку для уточнення якості рішень та прискорення збіжності. Ці модифікації підкреслюють потенціал ГА у вирішенні NP-складних проблем, таких як автономне навігування БПЛА, роблячи його переконливим вибором для застосувань, які вимагають швидкого, надійного та автономного планування польотного шляху дронів.

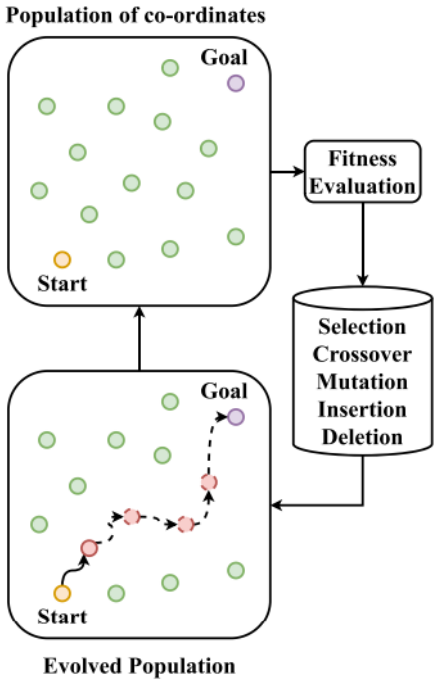

Рисунок. 1.12. Генетичний алгоритм для керування БПЛА [10]

Також, було розглянуто навчання з підкріпленням (НП), яке пропонує динамічний підхід до покращення автономності об'єктів у тривимірному просторі, з особливою придатністю до технології дронів, яка спрямована на автономне нейтралізування ворожих дронів. У рамках НП, дрон, що виступає

агентом, вчиться навігувати та реагувати на стохастичне тривимірне середовище через взаємодії, які приносять скалярні винагороди, тим самим удосконалюючи свою політику до оптимальної поведінки. Адаптивність НП підкреслюється її здатністю не тільки приймати рішення в відомих станах, але й досліджувати та виводити ефективні стратегії в раніше незустрінених сценаріях, що є критичним для реальних застосувань, де непередбачуваність є загальним явищем.

Застосовуючи алгоритми, такі як Q-навчання, БПЛА можуть автономно налаштовувати параметри управління, як-от ті, що в контролері ПІД, або ефективно навігувати через складні середовища. Ця самонавчальна характеристика є ключовою для поліпшення часу реакції та оперативної ефективності дронів, особливо порівняно з традиційними, більш детермінованими автономними системами. Просунуті реалізації також побачили використання багатоагентних систем, де дрони спільно вчаться та адаптуються для максимізації загальної продуктивності системи та досвіду користувача.

Успіх НП у автономності дронів далі посилюється завдяки таким покращенням, як апроксимація функцій для швидшої конвергенції та алгоритмів, розроблених для децентралізованого планування траєкторій в реальному часі. Ці просування дозволяють зменшити складність простору стан-дія і включення прогностичних моделей, тим самим сприяючи створенню більш стійкої та ефективної автономної системи. Включення НП у технологію дронів таким чином представляє обнадійливий шлях для розвитку витончених автономних систем, здатних до інтелектуальної, адаптивної поведінки у тривимірних середовищах.

Висновки до розділу 1

У цьому розділі було проведено детальний аналіз найпопулярніших існуючих рішень у сфері автономних операцій у тривимірних просторах, зокрема у контексті безпілотних літальних апаратів. Значний акцент було зроблено на розгляді та оцінці різних алгоритмів і технологій, які підсилюють автономію БПЛА та їх здатність до автономного нейтралізування інших об'єктів у повітрі.

Аналіз показав, що сучасні алгоритми обробки зображень, такі як YOLOv8, SSD та Faster R-CNN забезпечують високу точність та швидкість у виявленні об'єктів, хоча мають певні обмеження, наприклад, у виявленні дрібних або віддалених об'єктів. Алгоритм MiDaS, що займається оцінкою глибини на одному кольоровому зображенні, демонструє високу ефективність завдяки інноваційним підходам у зборі та обробці даних. Це свідчить про значний прогрес у сфері перцептивних здібностей БПЛА.

Додатково, було розглянуто використання штучного інтелекту з глибоким підкріпленим навчанням для управління БПЛА, що дозволяє дронам виконувати складні маневри та адаптуватися до швидкозмінних умов середовища. Це підсилює їх автономію та ефективність у виконанні завдань, особливо в умовах, недоступних для людей.

Різноманітні стратегії оптимізації, такі як оптимізація роєм часток та оптимізація мурашиними колоніями, демонструють потенціал у підвищенні ефективності та гнучкості автономного навігування БПЛА, водночас вирішуючи складні задачі обхіду перешкод та планування маршрутів.

Загалом, розглянуті технології та методології демонструють значні досягнення у сфері автономії БПЛА. Ці досягнення відкривають нові можливості для використання БПЛА у різноманітних сценаріях, починаючи від військових застосувань до пошуково-рятувальних місій, створюючи суттєву основу для подальших інновацій у цій області.

РОЗДІЛ 2

ФОРМУЛЮВАННЯ ЗАДАЧІ РЕАЛІЗАЦІЇ СИСТЕМИ ПІДВИЩЕННЯ АТОНОМНОСТІ ОБ’ЄКТІВ

Сучасні технологічні досягнення у сфері штучного інтелекту та машинного зору забезпечують значний потенціал для розробки високоавтономних БПЛА, спроможних діяти в динамічних і непередбачуваних умовах. Втілення цього потенціалу вимагає інтеграції та синергії різноманітних систем і алгоритмів.

Перша наукова новизна випливає з синтезу та оптимізації роботи двох передових систем машинного зору — YOLOv8 для розпізнавання об’єктів, MiDaS для оцінки глибини та штучного інтелекту з підкріпленим навчанням для контролю чотиримоторним об’єктом (БПЛА) в єдиному потоці обробки даних. Цей підхід не тільки розширює можливості кожної з систем, але й відкриває шлях для забезпечення високої точності, надійності визначення позиції, відстані до інших об’єктів та прокладання навігації до нього в тривимірному просторі, що є критично важливим для автономної місії об’єкту.

Друга наукова новизна роботи полягає у методі використання складної нейронної мережі з підкріпленим навчанням з можливістю обробки множини параметрів отриманих від бортових сенсорів, яка відповідає за керування чотиримоторним об’єктом (БПЛА) в тривимірному просторі. Унікальність підходу визначається не лише кількістю враховуваних вхідних параметрів, таких як координати цілі, швидкість, орієнтація, кутова швидкість та висота, але й способом їх взаємодії в процесі прийняття рішень нейромережею.

Далі, у розділі буде представлено детальний опис процесів адаптації, налаштування та інтеграції зазначених систем з використанням технології ROS як інтерфейсу для збору даних від сенсорів, а також механізмів підкріпленого навчання для ефективного управління поведінкою БПЛА. Завершальним кроком стане об'єднання усіх компонентів у єдину контролюючу архітектуру, спроможну до самостійного виконання завдань у складних умовах.

2.1 Метод вирішення поставлених задач

2.1.1 Метод роботи з YOLOv8 та MiDaS

Вирішення поставлених задач потребує комплексного та складного підходу. Можливе вирішення задачі тільки за допомогою розбиття її на більш малі задачі.

Першим і основним етапом буде навчання алгоритму YOLOv8 розпізнавати конкретні зображення та налаштувати модель MiDaS для співпраці.

Для більш якісного навчання моделі розпізнавання об’єктів YOLOv8 треба буде підготувати базу зображень з анотаціями до них. Потім цю базу треба буде збільшити за рахунок відомих алгоритмів аугментації. Для отримання задовільних результатів – модель YOLOv8 треба буде навчати на датасеті з мінімумом 5000 зображень мінімум 100 епох.

Правильна робота моделі MiDaS є не менш критичною для всієї системи, отже треба буде розробити формулу оцінки відставні до розпізнаного об’єкту. Важливо розуміти, що розпізнані об’єкти будуть окреслені прямокутною рамкою, отже великою частиною зображення в рамці буде фон, який буде збільшувати похибку оцінки глибини до об’єкту. Отже, для вирішення цієї проблеми буде використано множення глибини кожного пікселя на певну вагу, значення якої буде визначати Гаусівський розподіл. Це дозволить присвоювати більшу вагу пікселям, які знаходяться ближче до центру зображення, ніж тим, що по краях. Адже, більша частина розпізнаного об’єкта знаходиться ближче до центру прямокутника, ніж по краях.

Не менш важливим етапом є поєднання двох моделей – YOLOv8 та MiDaS у правильний «контролер», який буде видавати правильне фінальне значення. Схематичний розбір принципу роботи системи показано на рисунку 2.1. У разі, якщо YOLOv8 не розпізнає ніяких об’єктів на зображені – буде відправлено значення (1, 1, 1) – що буде негативним показником для штучного інтелекту, який контролює дрон.

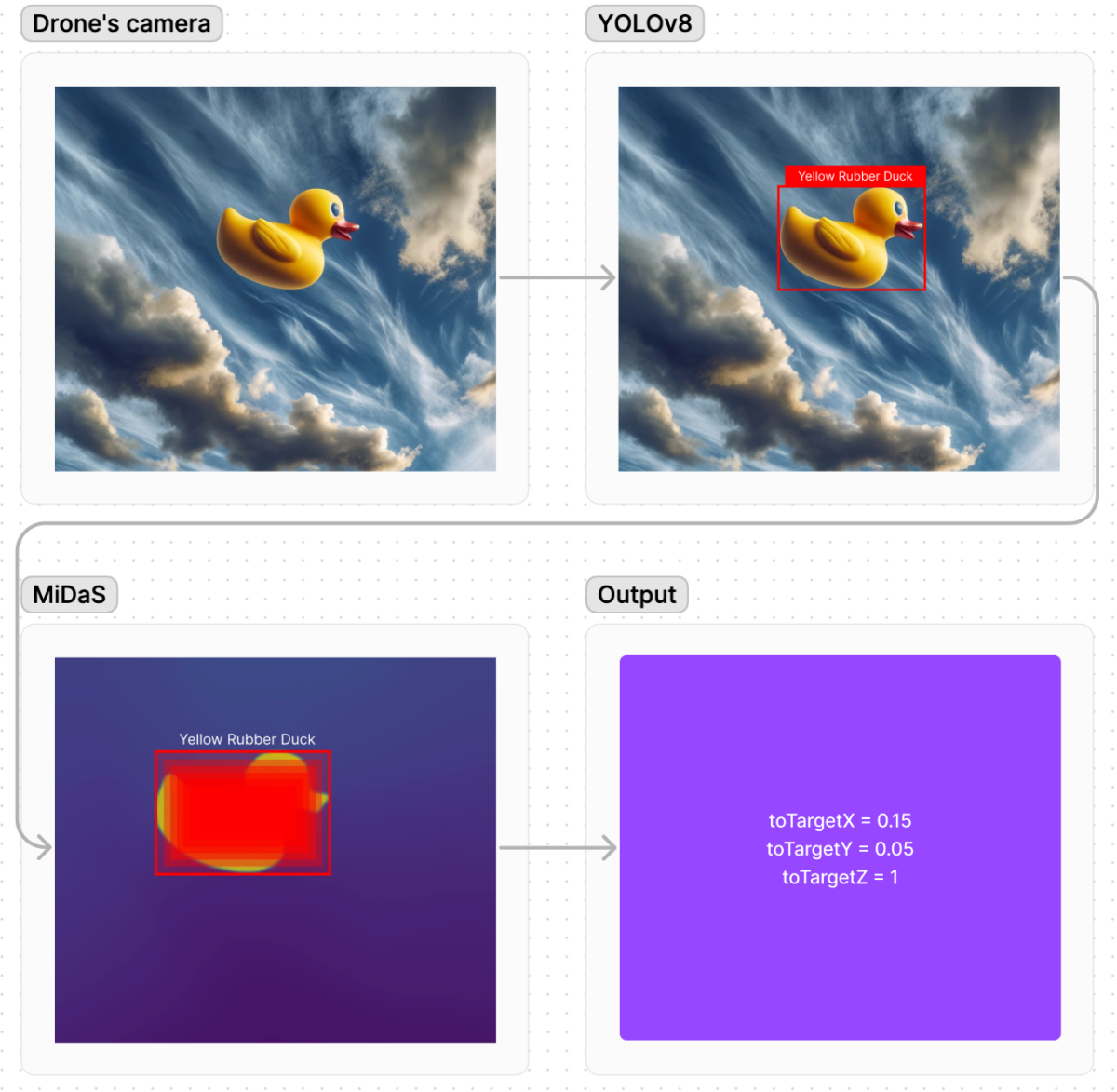

Рисунок. 2.1. Схематичний принцип роботи комплексної системи з YOLOv8 та MiDaS

2.1.2 Метод роботи зі штучним інтелектом з підсиленим навчанням та великою кількістю вхідних даних.

Головним елементом комплексної системи є штучний інтелект з підсиленим навчанням для контролю дрону з чотирьома моторами у три вимірному просторі. Отже, цій частині треба приділити не менш уваги.

Першим і головним є окреслення вхідних параметрів (observations). Для коректного контролю дрону штучному інтелекту, як і людині треба розуміти основні характеристики польоту, а саме: швидкість, кутову швидкість та орієнтацію по трьом осям. Також, для конкретної задачі польоту до рухомої цілі

– ШІ треба розуміти напрям до неї. Так як об’єкт розпізнається з зображення з камери дрону, яка направлена вперед – достатнім має бути вектори по осям X та Y до центру цілі від центру зображення. Ці вектори мають бути нормалізовані відносно розміру зображення. Також, останнім вхідним параметром буде «глибина» до цілі, яка буде сформована алгоритмом MiDaS та фільтрами. Отже, маємо наступний набір вхідних даних: швидкість, кутова швидкість, орієнтація по трьом осям, нормалізовані вектори до цілі та відносна «глибина» до цілі.

Зі структурою штучного інтелекту необхідно буде експериментувати, починаючи з двох прихованих шарів з 128 вузлами у кожному.

Наступним кроком є окреслення вихідних параметрів ШІ (actions). З двох основних варіантів (PID контролер та контроль обертів моторів напряму), в даній задачі буде використано другий варіант для скорочення затримки, оскільки вона присутня через використання YOLOv8 та MiDaS. Отже, в якості вихідних параметрів маємо 4 змінні – значення обертів кожного з моторів.

Через те, що в даній роботі використовується модель штучного інтелекту з підсиленим навчанням – ключовим елементом розробки системи є розробка формули винагороди. Для цього треба чітко розуміти поставлені задачі:

1. Дрон має летіти до розпізнаної цілі
2. Дрон має наздогнати ціль якомога швидше
3. Дрон не має розбитись

Отже, у остаточній формулі має бути враховано усі 3 параметри. Тож, формула має виглядати як приведена нижче формула 2.1.

$$R_{total} = R_{distance} + R_{time} + R_{crash} \quad (2.1)$$

Після об’єднання ШІ з попередньою системою маємо отримати цілісний контрольний апарат (рисунок 2.2.).

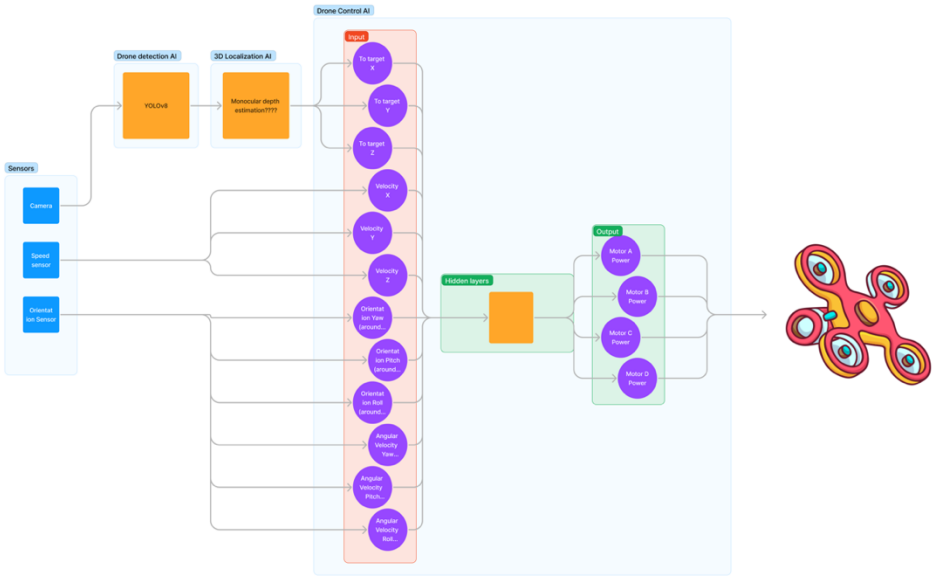

Рисунок. 2.2. Схема комплексної системи

У зв’язку з великою кількістю вхідних параметрів та вихідних параметрів складного типу, очікуємо, що система буде навчатись довше за свої аналоги. Наприклад, через використання PID контролеру в аналогах. Отже, для максимально ефективного навчання буде використовуватись симуляції по 8 секунд (240Гц симуляції, 24Гц контролю). Навчання ШІ буде проходити ітераціями по 54 мільйони кроків. Після кожної ітерації буде проводитись оцінка результатів.

2.2 Адаптація алгоритму YOLOv8 та його навчання для розпізнавання обраних об’єктів під час польоту.

2.2.1 Вибір даних для навчання

Для навчання алгоритму YOLOv8 можуть бути використані наступні датасети:

- "Purdue Aerial Drone Dataset", розроблений групою з Університету Purdue [11];
- "VisDrone Dataset", створений у рамках проєкту VisDrone [12];
- "A Deep Learning Approach to Drone Monitoring", авторство якого належить колективу дослідників під керівництвом Chelicynly [13].

Ці датасети містять велику кількість зображень різних типів БПЛА в різних умовах освітлення та погодніх умов, але, нажаль, цей об’єм зображень все ще є недостатнім для задовільного функціонування алгоритму розпізнавання у поставленій задачі. Для виправлення цього було прийнято рішення використовувати зображення резинової жовтої качки, оскільки ці об’єкти будуть достатньо контрастними порівняно з іншими частинами симуляції, що дозволить використати меншу кількість зображень для навчання YOLOv8 для отримання задовільного результату. Датасет, який буде використано має назву “Rubber Duck Detection Image Dataset” [17].

2.2.2 Доповнення та аугментація даних

Для ефективного навчання нейронних мереж, важливо мати велику та різноманітну навчальну вибірку, що дозволяє моделям краще узагальнювати властивості даних та покращувати їхню продуктивність на нових, раніше невідомих даних. Однак, у реальному світі збір великої кількість високоякісних даних може бути трудомістким та коштовним процесом. Щоб вирішити цю проблему та збільшити розмір та різноманітність наявного набору даних, використовуються техніки доповнення та аугментації даних.

Техніки аугментації даних включають в себе різноманітні просторові та піксельні перетворення зображень, які можуть підвищити здатність моделі узагальнювати навчання на нових даних. До основних технік аугментації зображень належать випадкове масштабування, обертання, зміна освітленості, горизонтальне та вертикальне відображення, та інші.

В контексті даного дослідження, аугментація даних відіграє критичну роль у підготовці навчального датасету для алгоритму YOLOv8, який використовується для розпізнавання об'єктів під час польоту БПЛА. Планується використати наступні техніки аугментації для збільшення та урізноманітнення навчального датасету:

1. Випадкове масштабування: Ця техніка включає зміну розмірів зображень на випадковий множник, що дозволяє моделі краще адаптуватися до об'єктів різних розмірів.

2. Обертання: Обертання зображень на випадкові кути дозволяє моделі краще розуміти орієнтацію об'єктів, що є важливим для ефективного відстеження та ідентифікації об'єктів у 3D просторі.
3. Зміна освітленості: Аугментація за допомогою зміни освітленості допомагає моделі краще адаптуватися до різних умов освітлення, що може бути особливо корисним у реальних умовах польоту, де освітленість може значно варіюватися.

Зазначені техніки аугментації допоможуть створити більш різноманітний навчальний датасет, який в свою чергу допоможе покращити узагальнюючу здатність моделі YOLOv8.

Також розглядається можливість використання додаткових технік аугментації, таких як випадкове змішування каналів кольору, адаптивна гістограмна еквалізація, та інші, для подальшого покращення розміру та різноманітності навчального датасету.

Розробка та використання відповідних стратегій аугментації даних є важливим етапом у підготовці навчального датасету для ефективного навчання моделі YOLOv8, та відіграє ключову роль у досягненні високої точності розпізнавання об'єктів під час польоту БПЛА.

2.2.3 Додавання штучних фільтрів

Для підвищення якості навчання нейронної мережі важливо забезпечити високу здатність моделі адаптуватися до різноманітних умов операційного середовища. Однією з ключових характеристик є здатність алгоритму YOLOv8 коректно розпізнавати об'єкти під час поганих погодних умов, таких як дощ або туман. Для цього, протягом процесу тренування, в датасет буде інтегровано додаткові зображення з штучно симульованими умовами дощу та туману.

Для симуляції умов дощу буде використано бібліотеку OpenCV, яка дозволяє створювати ефект дощу шляхом генерування випадкових ліній на зображенні, які імітують краплі дощу. Процедура включає створення функцій add_artificial_rain() та generate_rain(), яка створює випадкові лінії на зображенні, імітуючи краплі дощу. Оскільки кожна краплина дощу рухається з певною

швидкістю та в певному напрямку, для реалістичної симуляції буде важливо врахувати ці параметри під час генерування випадкових ліній. Таким чином, зображення буде аугментовано штучними краплинами дощу, що дозволило підвищити якість навчання моделі на даній частині датасету [14].

Для симуляції умов туману було обрано бібліотеку Imgaug, яка має вбудований аугментер для додавання туману до зображень. Цей аугментер є обгорткою навколо CloudLayer, який виконує один шар на зображенні з конфігурацією, що веде до досить густого хмарного покриття з низькочастотними шаблонами. Цей аугментер має бути досить стійким щодо розміру зображенн. Додавання штучного туману до зображень дозволить підвищити якість навчання моделі в умовах поганої видимості.

Результати цього етапу будуть вкрай важливими для забезпечення надійності та ефективності роботи алгоритму YOLOv8 в реальних умовах, де погодні умови можуть значно впливати на якість розпізнавання об'єктів. Штучна аугментація даних з використанням фільтрів дощу та туману дозволить підвищити якість навчання моделі та її здатність адаптуватися до різних умов операційного середовища.

2.2.4 Оптимізація гіперпараметрів

Процес тонкого налаштування моделі YOLOv8 починається з визначення оптимальних параметрів для додаткового навчання. Основною метою є покращення точності розпізнавання моделі для специфічного домену, в даному випадку - для розпізнавання об'єктів під час польоту БПЛА камікадзе. Важливим аспектом є вибір цільових датасетів, які відображають реальні сценарії та умови, в яких буде працювати система. Для цього можна використовувати датасети, які відображають рухомі об'єкти, а також специфічні характеристики зображень отриманих під час польоту.

Перший етап тонкого налаштування полягає у визначенні оптимального кроку навчання (learning rate). Зменшення кроку навчання дозволяє моделі уникнути можливих осциляцій навколо оптимального розв'язку та досягнути більш точних результатів. Однак занадто малий крок навчання може призвести

до дуже повільної конвергенції або застрягання в локальних мінімумах, тому важливо знайти золоту середину.

Другий етап включає в себе мінімізацію функції втрат (loss function). Функція втрат відображає різницю між реальними мітками даних та прогнозами моделі. В даному випадку, важливо вибрати таку функцію втрат, яка враховує особливості задачі розпізнавання об'єктів в 3D просторі. Можливо, буде корисним використати комбіновану функцію втрат, яка включає в себе компоненти для класифікації об'єктів та локалізації.

Крім того, може бути корисним провести аналіз впливу різних параметрів та гіперпараметрів на ефективність моделі, включаючи архітектуру мережі, оптимізатори, розмір пакету (batch size) та інше. Аналіз впливу різних факторів на точність розпізнавання може допомогти визначити оптимальні параметри для подальшого навчання та експлуатації моделі.

У підсумку, тонке налаштування моделі YOLOv8 є критично важливим кроком для розробки ефективної системи розпізнавання об'єктів для БПЛА камікадзе. Оптимізація моделі шляхом вибору оптимальних параметрів та гіперпараметрів, мінімізація функції втрат дозволяє забезпечити високу точність розпізнавання та ефективність системи в реальних умовах експлуатації.

2.2.5 Впровадження специфічних метрик оцінювання

У даний час, з урахуванням стрімкого розвитку технологій машинного навчання та комп'ютерного зору, виникає потреба у вдосконаленні методів оцінювання ефективності алгоритмів детекції та локалізації об'єктів. Однією з основних метрик, яка дозволяє оцінити точність локалізації алгоритмів детекції, є Intersection over Union (IoU). Ця метрика вимірює відношення об'єму перетину до об'єму об'єднання двох областей — реального розташування об'єкта та його розташування, визначеного алгоритмом.

Алгоритм YOLOv8, що використовується у даному дослідженні для розпізнавання об'єктів під час польоту БПЛА, вимагає точної оцінки його здатності коректно локалізувати об'єкти в просторі. Враховуючи високу швидкість руху БПЛА та можливі перешкоди на його шляху, важливо

забезпечити високу точність локалізації об'єктів для уникнення колізій та забезпечення ефективного виконання завдань. У цьому контексті, метрика IoU виступає як ключовий інструмент для оцінки та оптимізації роботи алгоритму YOLOv8.

Проте, стандартний підхід до використання метрики IoU може не враховувати специфіку руху БПЛА у 3D просторі. Тому, важливо розробити адаптовані метрики, які б враховували динаміку руху об'єктів, їх відносне розташування та інші фактори, що впливають на ефективність детекції та локалізації. Наприклад, можна розробити варіанти метрики IoU, які б враховували глибину зображення, отриману від моделі MiDaS, або динаміку руху об'єктів.

Більш того, важливо розглянути можливість впровадження додаткових метрик оцінювання, які б дозволили отримати більш повну картину щодо ефективності алгоритму YOLOv8 у реальних умовах. Це може включати метрики, які оцінюють час реакції алгоритму, його здатність адаптуватися до змін умов освітлення, а також робустність до помилок у вихідних даних від сенсорів та інших систем.

Такий підхід до оцінювання та оптимізації роботи алгоритму YOLOv8 дозволить не тільки підвищити його ефективність, але й забезпечити глибше розуміння взаємозв'язку між різними компонентами системи автономного керування БПЛА. Це, в свою чергу, сприятиме подальшому вдосконаленню алгоритмів та методів машинного навчання та комп'ютерного зору, що використовуються для автономного керування об'єктами у 3D просторі.

В результаті проведених оптимізаційних кроків має бути досягнуто високої точності та швидкості розпізнавання БПЛА в реальному часі, що є критично важливим для задачі наведення та знищення повітряних цілей. Таким чином, можливо буде практично оцінити алгоритм YOLOv8 на ефективність у вирішенні поставлених завдань.

2.3 Налаштування моделі MiDaS для оцінки глибини по зображенню, з даними від YOLOv8

Для запланованої системи – велика точність оцінки глибини не треба, оскільки вихідні дані будуть використовуватись як відносні, а не точні.

Отже, для реалізації плану використання моделі MiDaS у запропонованій системі, треба буде виконати такі етапи:

1. Вибір версії MiDaS

Необхідно провести аналіз останніх стабільних версій MiDaS та знайти оптимальну модель по залежності до обчислювальних ресурсів, оскільки зазвичай на БПЛА об’єктах обчислювальні ресурси є обмеженими.

2. Конфігурація MiDaS

Параметри MiDaS налаштовуються для оптимізації точності глибини. Важливо врахувати такі параметри як розмір вхідного зображення та масштаб глибини, щоб забезпечити оптимальну сумісність з даними, отриманими від YOLOv8.

3. Функція "Фільтрації" Даних

Важливою частиною майбутньої інтеграції MiDaS з YOLOv8 є розробка функції "фільтрації" даних. Ця функція забезпечує аналіз глибини тільки у межах областей, ідентифікованих YOLOv8. Це означає, що MiDaS буде оцінювати глибину тільки для визначених об'єктів, ігноруючи інші елементи зображення. Такий підхід дозволяє знизити обчислювальну вимогливість та збільшити точність вимірювань, оскільки аналіз зосереджується лише на релевантних об'єктах.

4. Обробка результатів оцінки глибини зображення

Як вказано в попередньому пункті, на виході з моделі MiDaS будемо отримувати тільки для визначеної YOLO частини зображення. Для отримання фінального результату, який можна було би відправити у штучний інтелект для керування об’єктом у 3D просторі треба обробити матрицю глибини і отримати скалярну величину. Так як у процесі використання модель MiDaS у процесі роботи буде отримувати зображення з бортової камери БПЛА і зображення

будуть інших об’єктів у 3D просторі – у більшості випадків на фоні визначеного YOLO об’єкта буде небо, отже якщо просто обраховувати середнє арифметичне значення даних глибини зі всієї рамки – дані будуть дуже розмитими та неточними. Зрозуміло, що треба відфільтрувати значення, які проходять певний поріг D_{min}. Найпростіше це зробити за допомогою присвоєння індексу ваги W_{xy} зі значенням 0 для певного пікселю.

Задля покращення вихідного результату (скаляру) – для кожного пікселя у рамці буде присвоюватись певний індекс ваги, який буде відображати важливість цього пікселя у подальшій формулі вираховування скаляру. Для визначення ваги для кожного пікселя буде використовуватись Гаусівський розподіл, який призначає більше значення ваги ближче до центру рамки та менший – до країв (формула 2.1).

$$W_{xy} = e - \frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}, \quad (2.1)$$

Де x_c, y_c – координати центру рамки, визначеної YOLO, σ – параметр, який контролює розподіл ваги в рамці.

Після визначення вагових коефіцієнтів та фільтрації глибини, розраховується оптимізоване середнє значення глибини D_{opt} в рамці за допомогою наступної формули (формула 2.2).

$$D_{opt} = \frac{\sum_{i=1}^{N}(D_{xy}*W_{xy})}{\sum_{i=1}^{N} W_{xy}}, \quad (2.2)$$

де N – кількість пікселів у рамці, D_{xy} – піксель з координатами *(x, y)*, W_{xy} – коефіцієнт ваги для пікселю з координатами *(x, y)*

5. Інтеграція MiDaS в загальну архітектуру

Модуль MiDaS розміщується в окремій функції для забезпечення легкої інтеграції з YOLOv8. Ця структура гарантує, що обидві системи можуть ефективно взаємодіяти, обмінюючись даними без конфліктів або перевантаження системи.

2.4 Реалізація алгоритму з глибоким підкріпленим навчанням для контролю об’єкту у 3D просторі.

Впровадження алгоритму глибокого підсиленого навчання (ГПН) в системи автономного керування безпілотними літальними апаратами (БПЛА) представляє собою проривний крок у галузі повітряної робототехніки. Основна ідея алгоритму полягає в його здатності приймати рішення в реальному часі та здійснювати керуючі дії на основі всебічного масиву вхідних даних від датчиків, що включають просторові координати, вектори швидкості, кути орієнтації, кутові швидкості та висоту.

1. Визначення вхідних та вихідних даних

- Представлення простору станів

$$S = \{x_{target}, y_{target}, z_{target}, V_x, V_y, V_z, \theta_{yaw}, \theta_{pitch}, \theta_{roll}, \omega_{yaw}, \omega_{pitch}, \omega_{roll}, A\} \quad (2.3)$$

де $x_{target}, y_{target}, z_{target}$ – координати до ціні по відповідним осям, V_x, V_y, V_z – швидкість об’єкту по відповідним осям, $\theta_{yaw}, \theta_{pitch}, \theta_{roll}$ – орієнтація об’єкту по відповідним осям, $\omega_{yaw}, \omega_{pitch}, \omega_{roll}$ – кутова швидкість об’єкту по відповідним осям, A – висота об’єкту

- Представлення простору дій (формула 2.4)

$$A = \{a_1, a_2, a_3, a_4\} \quad (2.4)$$

де a_1, a_2, a_3, a_4 – задання швидкості обертання моторів 1, 2, 3, 4 об’єкту (RPM).

2. Вибір алгоритму

Для завдання контролювання об’єкту у три вимірному просторі за допомогою нейронної мережі з глибоким підсиленням навчання, було обрано Proximal Policy Optimization (PPO). Ось переваги та недоліки PPO у контексті даного застосування:

Переваги PPO:

- Ефективність використання даних: PPO є більш ефективним з точки зору використання даних порівняно з багатьма іншими алгоритмами RL. Це критично важливо в реальних застосуваннях, як контроль об’єктів, де збір даних може бути дорогим і займати багато часу.

- Стабільність та надійність: PPO забезпечує більш стабільне та надійне навчання. Він обмежує ступінь, в якому політика може змінюватися при кожному оновленні, зменшуючи шанси на дестабілізацію навчання через великі раптові оновлення політики.
- Збалансованість дослідження та використання: PPO знаходить хороший баланс між дослідженням нових дій та використанням відомих ефективних дій, що є важливим у складних середовищах, де агент повинен постійно адаптуватися до нових ситуацій.
- Простота у впровадженні та налаштуванні: PPO простіше реалізувати та налаштувати порівняно з DDPG. У нього менше гіперпараметрів, що робить його легшим для конфігурації, особливо у складних сценаріях, як-от контроль дронів.
- Ефективність у середовищах з безперервними просторами дій: PPO довів свою ефективність у середовищах з безперервними просторами дій, що є актуальним для керування моторами об’єкту.

Недоліки PPO:

- Висока інтенсивність обчислювальних ресурсів: PPO може бути більш обчислювально інтенсивним, ніж деякі інші методи, через необхідність кількох епох оптимізації на кожен зразок даних.
- Потенційно повільніше збігання: У деяких випадках PPO може збігатися повільніше, ніж DDPG, оскільки робить менші, обережніші оновлення політики.
- Субоптимальність у вкрай складних середовищах: Хоча PPO загалом є стійким, у надзвичайно складних середовищах або завданнях з високою складністю динаміки він може мати труднощі зі знаходженням найбільш оптимальної політики.
- Чутливість до гіперпараметрів: Незважаючи на меншу кількість гіперпараметрів, PPO все ще може бути чутливим до їх налаштувань. Неправильні налаштування можуть призвести до розвитку субоптимальної політики.

- Ризик потрапляння в локальні оптимуми: Як і багато алгоритмів RL, PPO може застрягти в локальних оптимумах, особливо в середовищах з рідкісними або обманливими структурами винагород.

Враховуючи характер завдання — керування об’єктом у 3D-просторі з безперервними діями виводу — сильні сторони PPO у роботі з безперервними просторами дій, його баланс дослідження та використання, а також його стабільність у навчанні роблять його сильним кандидатом. Складність завдання та необхідність надійного та ефективного навчання ще більше нахиляють шальки на користь PPO. Незважаючи на виклики, такі як обчислювальні вимоги та потенційна чутливість до налаштувань гіперпараметрів, ці проблеми можуть бути управліні за допомогою ретельної реалізації та експериментів.

3. Визначення кількості шарів

Вибір кількості прихованих шарів та кількості нейронів у кожному шарі є ключовою частиною проектування нейронної мережі. Для даного завдання керування дроном вибір кількості прихованих шарів та кількості нейронів базується на кількох міркуваннях:

- Складність завдання: Завдання керування об’єктом у 3D-просторі з різноманітними входами та виходами є складним. Ця складність вимагає моделі, здатної вчитися широкому спектру нелінійних відносин. Технічно один прихований шар може апроксимувати будь-яку функцію, але додаткові шари можуть більш ефективно та ефективно вчитися складним візерункам.
- Два приховані шари: Два приховані шари часто є хорошою відправною точкою для багатьох проблем. Це баланс між збереженням простоти мережі, щоб не перенавчитися на даних, і при цьому бути достатньо складними для відображення необхідних відносин. Для багатьох практичних проблем, включаючи завдання, як розпізнавання зображень та системи керування, було виявлено, що два приховані шари добре працюють.

- 256-128 нейронів: Вибір кількості нейронів у кожному шарі є компромісом між ємністю моделі та ризиком перенавчання. Занадто мала кількість нейронів може призвести до недонавчання, коли модель не вдається відобразити складність даних. З іншого боку, занадто велика кількість нейронів може призвести до перенавчання, коли модель вчиться шуму в тренувальних даних, а не справжнім основним відносинам. 256-256 нейронів пропонують золоту середину, забезпечуючи достатню ємність для навчання без надмірного збільшення ризику перенавчання. Цей діапазон часто є хорошою відправною точкою, але оптимальна кількість може варіюватися на основі конкретної природи завдання та даних. Для початку буде використовуватись 256 нейронів і якщо відбудеться перенавчання моделі – буде пониження кількості.
- Обчислювальна ефективність: Кількість нейронів та шарів також впливає на обчислювальну ефективність навчання мережі. Більше нейронів та шарів означає більше параметрів для тренування, що вимагає більше обчислювальних ресурсів та часу. Запропонований діапазон намагається збалансувати обчислювальну ефективність з продуктивністю моделі.

На практиці оптимальна конфігурація нейронної мережі часто знаходиться шляхом експериментування та налаштування. Запропонована конфігурація є відправною точкою, і може знадобитися налаштувати її на основі продуктивності моделі під час навчання та перевірки.

4. Нормалізація вхідних даних

Нормалізація даних є ключовим кроком у підготовці вхідних даних для нейронної мережі, особливо у складних завданнях керування, як навігація об’єктом. Процес передбачає масштабування вхідних характеристик так, щоб вони були у подібному діапазоні. Це важливо, оскільки характеристики на більших масштабах можуть непропорційно впливати на процес навчання, ведучи до менш ефективного тренування.

Нормалізація кожного вхідного параметру:

- *To target (x, y):* Нормалізувати ці координати на основі максимально очікуваного розміру фото (вхідного параметру у YOLO). Тобто, кожну координату треба буде поділити на максимальний розмір отриманого зображення по відповідній осі.
- *To target (z):* В даному випадку нормалізація не потрібна, оскільки вихідні параметри з моделі оцінки глибини MiDaS будуть в діапазоні (0, 1)
- *Velocity (x, y, z):* Нормалізувти на основі максимально очікуваної швидкості. Якщо максимальна швидкість становить 10 м/с, поділити кожну компоненту швидкості на 10.
- *Orientation (yaw, pitch, roll):* Для нормалізації – поділити на максимальний кут – 360.
- *Angular Velocity (yaw, pitch, roll):* Подібно до лінійної швидкості, нормалізувати їх на основі їх максимально очікуваних значень.
- *Altitude*: Нормалізувати на основі максимальної експлуатаційної висоти об’єкту.

Чому потрібна нормалізація:

- Однорідність масштабу: Різні входи мають різні діапазони. Наприклад, швидкість може бути у діапазоні метрів за секунду, тоді як кути орієнтації вимірюються у градусах або радіанах. Нормалізація приводить всі входи до подібного масштабу, запобігаючи домінуванню характеристик з більшими діапазонами у процесі навчання.
- Швидше збігання: Коли характеристики мають подібні масштаби, алгоритм градієнтного спуску, який використовується для тренування мережі, збігається швидше. Це тому, що ландшафт оптимізації стає більш керованим, з градієнтами, які не надмірно впливаються певними характеристиками.
- Числова стабільність: Нейронні мережі, особливо глибокі, схильні до проблем, як-от зникнення або вибухання градієнтів. Нормалізація допомагає пом'якшити ці проблеми, зберігаючи вхідні значення у діапазоні, який менш схильний викликати числову нестабільність.

5. Вибір функції активації

Вибір ReLU (Rectified Linear Unit) як функції активації для прихованих шарів у глибокій нейронній мережі (DNN) базується на кількох переконливих причинах, особливо актуальних для завдання керування об’єктом у три вимірному просторі за допомогою навчання з підсиленням:

- Ефективність навчання глибоких мереж: ReLU допомагає ефективно тренувати глибокі мережі. Вона є обчислювально ефективною, оскільки включає простіші математичні операції (лише порогове обмеження на нулі).
- Зменшення проблеми зникнення градієнту: У глибоких мережах проблема зникнення градієнту є поширеною, коли градієнти стають дуже малими, ефективно зупиняючи навчання мережі. Лінійний характер ReLU для позитивних значень означає, що вона не насичується у позитивному діапазоні і тому не страждає від зникнення градієнтів так само, як сигмоїдні або tanh функції.
- Розріджена активація: У мережі ReLU в будь-який момент активна лише частина нейронів. Це призводить до розрідженої активації, роблячи мережу більш ефективною та легшою для тренування. Розріджені представлення, здається, є більш корисними для навчання і є біологічно правдоподібними (імітуючи, як активуються нейрони в мозку).
- Покращена узагальненість: Мережі з ReLU, як правило, краще узагальнюють невидимі дані. Це можна пояснити його лінійною, ненасиченою формою, яка дозволяє моделі вчитися швидше та більш ефективно.
- Біологічна правдоподібність: ReLU вважається більш біологічно правдоподібною, ніж інші традиційні функції активації. Вона імітує спосіб роботи біологічних нейронів, де нейрон або активується, або ні.

Однак важливо зауважити, що ReLU не позбавлена недоліків. Найважливішим з них є проблема "вмирання ReLU", коли нейрони іноді стають неактивними і видають лише нуль, по суті, "вбиваючи" нейрон. Це можна

пом'якшити, використовуючи варіанти ReLU, такі як Leaky ReLU або Parametric ReLU.

У контексті керування об’єктом переваги ReLU, особливо її ефективність та ефективність у вирішенні проблеми зникнення градієнтів у глибоких мережах, роблять її підходящим вибором для архітектури даної нейронної мережі.

6. Формулювання функції винагороди

Повна функція винагороди має включати в себе 3 елементи:

- Винагорода за відстань до цілі ($R_{distance}$):

Ця частина має забезпечувати вищу винагороду за близькість до цілі. Вона може базуватися на евклідовій відстані між поточним положенням дрона (x, y, z) та положенням цілі $(x_{target}, y_{target}, z_{target})$ (формула 2.5).

$$R_{distance} = -\sqrt{(x - x_{target})^2 + (y - y_{target})^2 + (z - z_{target})^2}, \quad (2.5)$$

Це рівняння забезпечує вищу винагороду (менш негативну), коли дрон наближається до цілі.

- Штраф за час (R_{time}) (формула 2.6):

Щоб стимулювати ефективність, ви можете штрафувати дрон за кожен крок часу. Це може бути постійне негативне значення.

$$R_{time} = -C_{time}, \quad (2.6)$$

де C_{time} - мале постійне значення.

- Штраф за аварію (R_{crash}) (формула 2.7):

Якщо дрон зазнає аварії або порушує експлуатаційні обмеження, він повинен отримати значний негативний штраф.

$$R_{crash} = -C_{crash}, \quad (2.7)$$

де C_{crash} - це велике негативне постійне значення.

7. Загальна винагорода: Загальна винагорода на кожному кроці часу є сумою цих компонентів (формула 2.8).

$$R_{total} = R_{distance} + R_{time} * n_{steps} + R_{crash} \quad (2.8)$$

Значення для C_{time} та C_{crash} потрібно вибирати на основі масштабу інших винагород та специфіки симуляційного середовища. Вони повинні бути збалансовані так, щоб дрон не надмірно пріоритетизував одну ціль над іншими. Наприклад, C_{time} повинен бути достатньо малим, щоб не затьмарювати винагороду за відстань, але достатньо значущим, щоб стимулювати швидкі дії. Аналогічно, C_{crash} повинен бути достатньо великим, щоб уникнення аварій стало високим пріоритетом.

2.5 Комбінація усіх частин контролюючої архітектури в одну систему

Основна мета - створити узгоджену систему, яка використовує алгоритм YOLOv8 для розпізнавання об'єктів, MiDaS для оцінки глибини та глибоку модель навчання з підсиленням (RL) для динамічного керування в 3D просторі.

Інтегрована система функціонує послідовно, але взаємопов'язано:

1. Захоплення зображення: об’єкт, обладнаний камерою, у реальному часі захоплює зображення оточення. Цей крок є критичним, оскільки він надає основне джерело даних для розпізнавання об'єктів та оцінки глибини.
2. Виявлення об'єктів за допомогою YOLOv8: Кожне захоплене зображення обробляється алгоритмом YOLOv8. Ця модель спеціально навчена ідентифікувати та локалізувати об'єкти інтересу в оперативному середовищі об'єкту. YOLOv8 видає координати (toTargetX, toTargetY) виявлених об'єктів разом з рейтингом впевненості для кожного.
3. Вибір об'єкта та оцінка глибини:
    - Якщо виявлено кілька об'єктів, система вибирає той, що має найвищий рейтинг впевненості. Цей вибір важливий для зосередження уваги на найбільш ймовірній меті.
    - Сегмент зображення обраного об'єкта подається на модель MiDaS, яка обчислює глибину відносно об'єкту.
4. Передача даних та генерація сигналу керування:

- Координати з YOLOv8 та інформація про глибину з MiDaS об'єднуються для формування всебічного просторового розуміння (toTargetX, toTargetY, toTargetZ).
- Ці передані дані подаються до моделі штучного інтелекту. Ця модель, навчена приймати рішення на основі датчиків, обчислює оптимальні сигнали керування для системи об'єкту (керування 4 моторами) для ефективного навігації в 3D просторі.

5. Управління відсутністю виявлених об'єктів:
   - У сценаріях, коли не виявлено об'єктів, система передає стандартний сигнал (нульові значення), який об'єкт інтерпретує як інструкцію утримувати свою поточну позицію та висоту, фактично переходячи в стан очікування на місці. Цей стан важливий для сценаріїв, коли об'єкту потрібно зупинитися та переоцінити своє оточення.

Однак, інтеграція цих компонентів представляє кілька викликів:

1. Обробка в реальному часі: Система повинна обробляти та реагувати в реальному часі. Оптимізації ефективності алгоритмів та паралельна обробка можуть вирішити цей виклик.
2. Стійкість: Система повинна справлятися з різними умовами навколишнього середовища та типами об'єктів. Розширене навчання моделей YOLOv8 та штучного інтелекту з різноманітними наборами даних може підвищити стійкість.

Висновки до розділу 2

У другому розділі було визначено методологію побудови інтегрованої системи управління об’єктом у три вимірному просторі, використовуючи передові алгоритми штучного інтелекту та комп'ютерного зору. Основними елементами цієї системи є алгоритм YOLOv8 для розпізнавання об'єктів, модель MiDaS для оцінки глибини зображень та глибоке підсилене навчання для управління БПЛА в тривимірному просторі.

Було описано процес адаптації алгоритму YOLOv8 для розпізнавання об'єктів під час польоту та налаштування моделі MiDaS для оцінювання глибини на основі зображень. Це важливо для точного визначення положення об'єктів у просторі.

Розглянуто поєднання YOLOv8 і MiDaS для отримання повних даних про місцезнаходження та відстань до об'єктів. Це дозволяє системі точно оцінювати позиції цільових об'єктів.

Визначено процес імплементації алгоритму глибокого підсиленого навчання для управління об'єктом у тривимірному просторі. Цей алгоритм є ключовим для ефективного маневрування об’єктом.

Останнім етапом було поєднання всіх частин системи управління в єдине ціле. Це включає синхронізацію даних від різних джерел, оптимізацію часу обробки даних та забезпечення їх точності та надійності.

У підсумку, розроблено комплексний підхід до створення інтелектуальної системи управління об’єктом, яка інтегрує передові технології обробки зображень та штучного інтелекту. Цей підхід дозволяє досягти високої точності та ефективності у визначенні положення та управлінні об’єктом в динамічному тривимірному середовищі.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ МЕТОДУ ЗБІЛЬШЕННЯ АВТОНОМНОСТІ ОБ'ЄКТУ У 3D ПРОСТОРІ

3.1 Огляд засобів реалізації

Для розробки та реалізації моделі з використанням методів підсиленого навчання (reinforcement learning) для керування дроном у тривимірному просторі, який слідує за рухомою ціллю, було вибрано мову програмування Python версії 3.9. Вибір обумовлений зручністю та гнучкістю Python у наукових дослідженнях, а також широким спектром доступних бібліотек. Для ефективної реалізації проекту були використані наступні бібліотеки:

- Numpy, math, random для математичних обчислень та алгоритмічних операцій;
- os, glob для роботи з файловою системою;
- cv2 (OpenCV) для обробки та аналізу зображень;
- imgaug для збільшення даних за допомогою аугментації зображень;
- gym, gymnasium, gym-pybullet-drones для моделювання середовища дрона;
- time, datetime для роботи з часовими функціями;
- stable_baselines3 для реалізації алгоритмів підсиленого навчання;
- collections для зручної роботи з колекціями даних;
- pybullet для фізичного моделювання в середовищі 3D.

Розробка проводилася на ноутбуці Apple MacBook Air M2, що забезпечує достатню обчислювальну потужність та зручність у використанні.

Програмування та тестування коду виконувалось за допомогою інтегрованого середовища розробки PyCharm від компанії JetBrains та середовища Colab від Google, яке забезпечує можливості хмарних обчислень.

Основні технології, що були використані у дослідженні, включають:

- Розпізнавання об'єктів для ідентифікації та відстеження цілей;

- Оцінка монокулярної глибини для визначення відстані до об'єктів у просторі;
- Підсилене навчання (reinforcement learning), зокрема алгоритм PPO (Proximal Policy Optimization), для розробки та тренування моделі керування дроном.

При роботі з програмою рекомендуються такі мінімальні технічні характеристики комп’ютерної системи:

- Процесор з тактовою частотою не менше 3 ГГц;
- Об’єм оперативної пам’яті не менше 8 ГБ;
- Вільний об’єм накопичувача не менше 16 ГБ;

3.2 Структура комплексної моделі

Для реалізації моделювання була використана система, що включає симуляцію gym-pybullet-drones. Ця система забезпечує отримання зображень з камери дрону, а також інформації про швидкість, кутову швидкість та орієнтацію дрону.

Процес обробки зображень реалізований за допомогою класу DroneController. Цей клас відправляє зображення до YOLOv8ObjectDetector який здійснює пошук об'єкта на зображенні 640x640 пікселів та відправляє назад векторні координати до об'єкта та координати описуючої рамки цього об’єкта. Після цього DroneController передає зображення та координати рамок об'єкта до DepthEstimator який визначає глибину знаходження об'єкта та відправляє цю інформацію назад.

На основі отриманих координат (toTargetX, toTargetY з YOLO та toTargetZ з MiDaS) та інших даних з симуляції (швидкість, кутова швидкість, орієнтація), та буферу попередніх 10 ітерацій - модель штучного інтелекту виконує прогнозування значень обертів на хвилину (RPM) для кожного з чотирьох моторів дрону. Ці дані потім передаються назад у симуляцію для керування дроном.

Завершуючи, розроблена система забезпечує комплексний підхід до керування дроном у тривимірному просторі, використовуючи передові алгоритми машинного навчання та комп'ютерного зору. Вся архітектура системи розроблена з урахуванням гнучкості та масштабованості, що дозволяє ефективно адаптувати її до різних сценаріїв застосування.

Повний лістинг коду представлений у Додатку В.

3.3 Налаштування і навчання моделі YOLOv8

3.3.1 Формування та аугментація датасету для навчання моделі YOLOv8

Завантажений датасет [17] до будь-яких маніпуляцій з зображеннями має вигляд, як на рисунку 3.1:

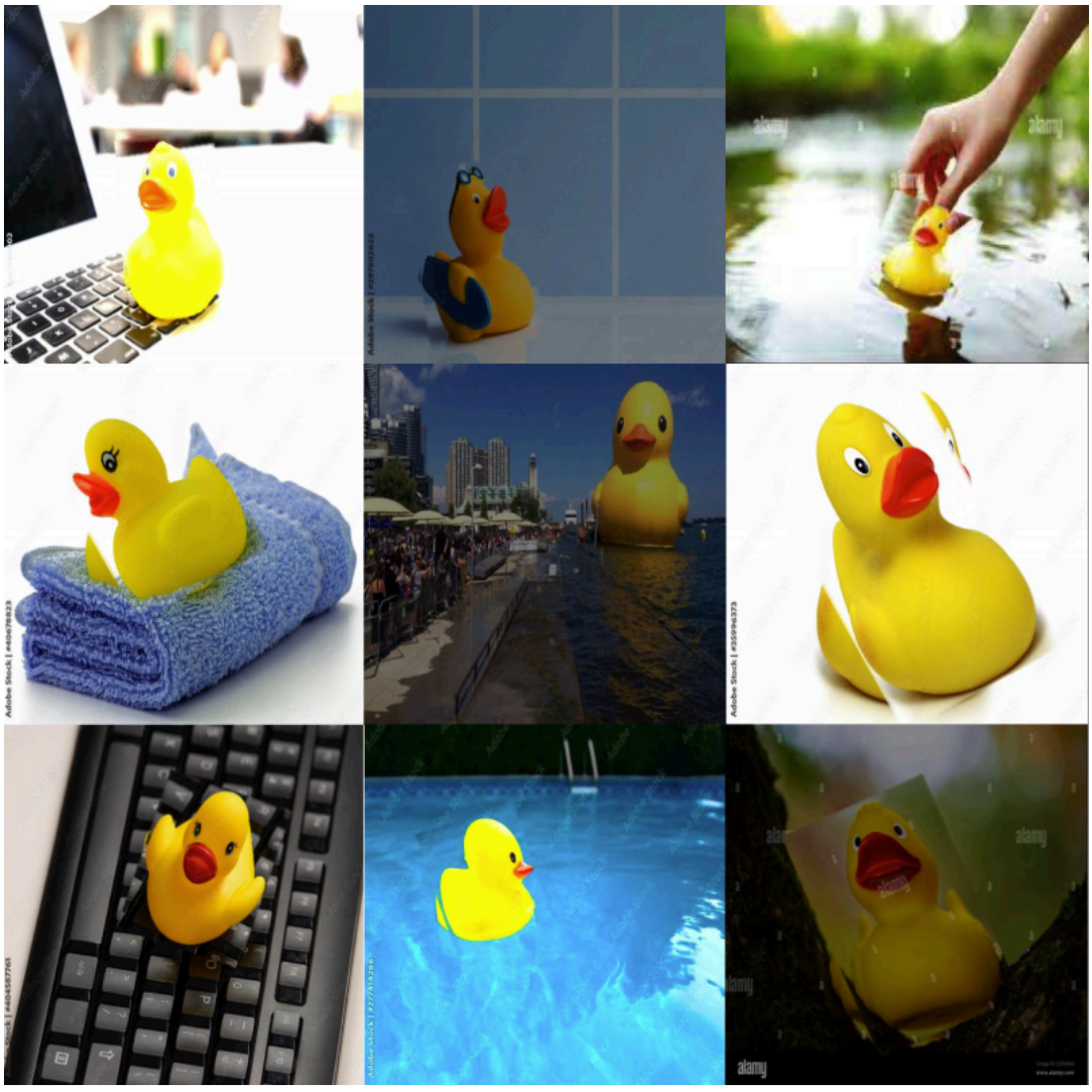

Рис. 3.1 Вибірка з датасету резинових жовтих качок

Датасет містить в собі 254 зображення, що є скоріш недостатнім для коректного навчання моделі YOLOv8, отже для збільшення об’єму зображень було написано скрипт, якій містить в собі 11 функцій:

- run() – основна функція, яка запускає увесь скрипт
- load_dataset(image_folder, annotation_folder) – завантажує датасет у програму
- save_dataset(image_folder, annotation_folder) – зберігає новий датасет з програми у папку
- random_non_symmetrical_scaling(dataset, percentage) – масштабує випадкові зображення
- random_rotation(dataset, percentage) – обертає випадково обрані зображення
- random_lighting_change(dataset, percentage) – додає зміну освітленості на випадково обрані зображення
- random_color_channel_mix(dataset, percentage) – випадково змішує канали для випадково обраних зображень
- apply_histogram_equalization(dataset, percentage) – додає випадкову еквалізацію гістограми випадково обраних зображень
- add_artificial_rain(dataset, percentage) – додає ефект дощу для випадково обраних зображень
- add_artificial_fog(dataset, percentage) – додає ефект туману для випадково обраних зображень

Для повного лістингу див. Додаток А

Після обробки масиву зображень представленим скриптом – отримано 5232 зображення (рисунок 3.2.).

Рисунок. 3.2. Вибірка з датасету резинових жовтих качок після аугментації

3.3.2 Створення окремого класу для майбутньої інтеграції моделі YOLOv8 в комплексну систему

Після навчання моделі YOLOv8 на розширеному пулі зображень було створено клас YOLOv8ObjectDetector з основним методом detect_objects для розпізнавання зображення. В якості вихідного параметру методу detect_objects маємо список з об’єктів, які містять в собі наступні параметри:

- bbox – кортеж з інформацією про описуючий прямокутник (x, y, w, h)
- confidence – відсоток впевненості моделі в розпізнаному об’єкті
- distance_vector – вектор дистанції від центру розпізнаного об’єкту до центру зображення

3.3.3 Перевірка роботи моделі YOLOv8 в тестовому середовищі.

Для затвердження правильності моделі YOLOv8 було проведено перевірку її здатності. Для цього з середовища контролю об’єктів у три вимірному просторі

було завантажено зображення з камери дрону розміром 128x128 пікселів (рисунок 3.3.).

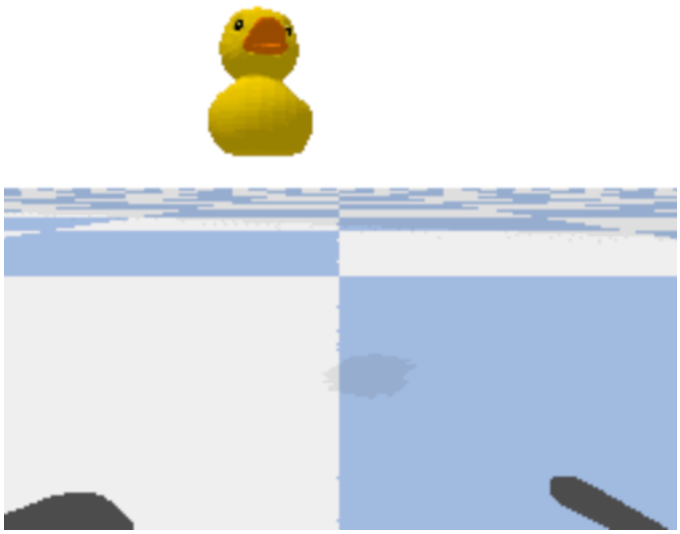

Рисунок. 3.4. Зображення з камери дрону у тестовому середовищі

Після цього зображення було проведено крізь модель YOLOv8 налаштовану на розпізнавання резинових качок. Для візуалізації роботи моделі в режимі тестування було вивантажено результуюче зображення для візуальної оцінки (рисунок 3.4.).

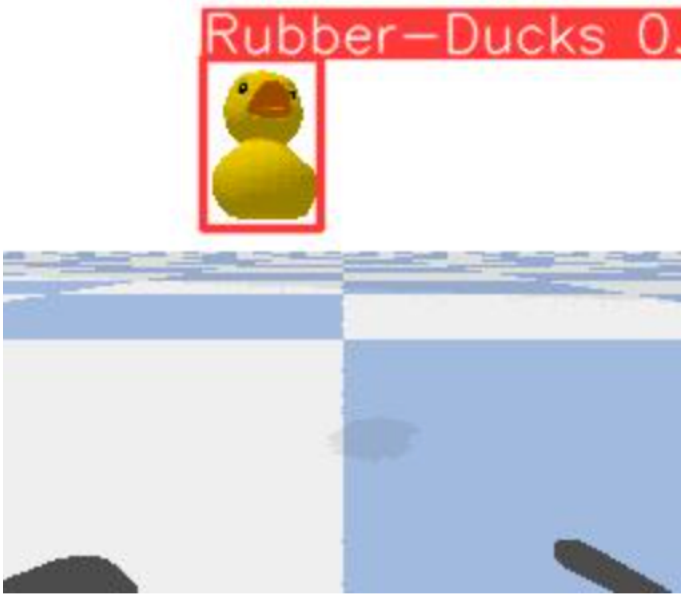

Рисунок. 3.4. Результат роботи моделі YOLOv8 у тестовому середовищі

Основними вихідними даними з моделі були дво-вимірні вектори нормалізованої відстані від центру розпізнаваного об’єкту до центру зображення з камери дрону – (0.0056797266, -0.1995095610).

3.4 Налаштування моделі MiDaS для оцінки глибини зображення

Створений клас призначений в інтеграції з моделлю MiDaS для оцінки глибини об'єктів у зображеннях. Клас приймає зображення у форматі RGB, яке було перетворене як `np.reshape(rgb, (h, w, 4))` з розміром 640x640, та оцінює глибину об'єкта всередині вказаного обмежувального прямокутника (bounding box). Оцінка глибини здійснюється не простим усередненням по обмежувальному прямокутнику, а використовуючи зважене середнє, засноване на гауссовому розподілі, центрованому навколо обмежувального прямокутника.

Клас `DepthEstimator` містить головний метод - estimate_depth, який:

- Передбачає глибину зображення.
- Обраховує зважену глибину за допомогою гауссового розподілу.

Вхідні параметри методу:

- image – зображення з розпізнаним об’єктом за допомогою YOLO v8
- bbox – координати описуючого прямокутника (x, y, w, h)
- sigma - параметр, який контролює розподіл ваги
- D_min - мінімальний поріг глибини для фільтрації

3.5 Реалізація алгоритму з глибоким підкріпленим навчанням для контролю об’єкту у 3D просторі.

У цьому дослідженні було обрано фреймворк gym-pybullet-drones [18] для симуляції та тестування. Цей вибір зумовлений гнучкістю та широкими можливостями фреймворка у створенні реалістичних 3D симуляцій дронів. Для реалізації моделі підкріпленого навчання (RL) було використано бібліотеку stable-baselines з Proximal Policy Optimization (PPO) алгоритмом. Особливість конфігурації моделі PPO полягає у використанні двох прихованих шарів, кожен з яких має 128 нейронів. Ця структура була обрана з міркувань балансу між ефективністю обчислень та потужністю моделі.

Для правильного налаштування та тестування середовища дослідження було розділено на два етапи, перший з яких включає отримання вхідних даних напряму з простору симуляції, без моделей YOLO та MiDaS.

3.5.1 Налаштування середовища та тестування з кінематичними спостереженнями

Перший крок полягав у створенні нового простору TargetAviary, що дозволило додати динамічний компонент (ціль) до симуляції. Це було необхідно для точного моделювання руху об'єктів у 3D просторі, а також для більш якісної візуалізації тесту. Додавання TargetAviary забезпечило більш високу точність та реалізм у моделюванні.

Для коректного навчання моделі треба симулювати складний шлях цілі у тому ж просторі, де і буде об’єкт. Також, цей шлях має автоматично оновлюватись з кожною новою ітерацією. Для цього було створено функцію _generateTargetPath(). Логіка роботи методу:

1. Генерує стартові координати (start) та кінцеві (end): Функція спочатку генерує випадкові стартові та кінцеві координати за допомогою методу self.generate_random_position, обмежуючи їх зазначеними мінімальними та максимальними значеннями.
2. Перевірка на мінімальну відстань: Використовуючи метод _distance_between_points, функція розраховує Евклідову відстань між стартовими та кінцевими координатами. Якщо відстань менше 4 одиниць, кінцеві координати генеруються знову, доки відстань між стартом і кінцем не буде мінімум 4 одиниці.
3. Створення шляху: Функція ініціалізує порожній словник path, який буде містити визначену траєкторію руху.
4. Обчислення амплітуди та довжини хвилі: Для кривизни шляху використовується синусоїда. Амплітуда синусоїди визначається як одна третина різниці між максимальним і мінімальним значенням координати y, а довжина хвилі відповідає кількості кроків.
5. Генерація траєкторії: Для кожного кроку в межах заданої кількості кроків функція обчислює проміжні координати x та z, лінійно інтерполюючи між стартовими та кінцевими координатами. Координата y обчислюється за допомогою синусоїдальної функції, що додає кривизну траєкторії.

6. Коригування координати y: Координата y коригується так, щоб вона не виходила за межі заданих мінімальних та максимальних значень.
7. Збереження траєкторії: Кожен крок з його координатами зберігається у словнику path.
8. Оновлення цільового шляху дрону: Завершуючи, функція оновлює атрибут self.TARGET_PATH цією новою траєкторією.

Наступним кроком було налаштування простору нагород та зміна розрахунку нагород. Це включало модифікацію функції нагород, щоб краще відображати реальні умови польоту та управління об’єктом у поставленій задачі. Зміни в розрахунку нагород допомогли підвищити ефективність навчання моделі, забезпечуючи більш точне та швидке адаптування до змін у середовищі.

Останнім етапом було налаштування моделі PPO від бібліотеки stable_baselines3. Зокрема, було проведено калібрування параметрів навчання, що дозволило моделі краще адаптуватися до особливостей симуляційного середовища. Це включало зміни в швидкості навчання, розмірі пакетів даних та стратегії оптимізації.

Увесь лістинг коду представлений у Додатку Б.

3.6 Об’єднання алгоритму з підкріпленим навчанням з моделями YOLOv8 та MiDaS

Створений Python клас призначений для обробки зображень з метою визначення векторних відстаней до цілі. Основний метод класу - *process_image* зображення як вхід і виконує наступні дії:

- Виявлення об'єктів за допомогою YOLO: Спочатку він використовує *yolo_detector* для виявлення об'єктів на зображенні. Якщо об'єкти не виявлені, метод повертає стандартні значення (0, 0, 0), що інтерпретовано як команда об’єкту підтримувати плаваючу висоту.
- Обробка першого виявленого об'єкту: Якщо об'єкти виявлені, метод бере перший з виявлених об'єктів. Він визначає його положення (координати *toTargetX і toTargetY)* та прямокутник об'єкта *bbo).*

- Оцінка глибини за допомогою MiDaS: Далі використовує *depth_estimator* для оцінки глибини об'єкта *(toTargetZ),* використовуючи положення та прямокутник об'єкта.
- Останнім кроком буде повернення кортежу з трьома векторними дистанціям до цілі *(toTargetX, toTargetY, toTargetZ)*

  Увесь лістинг коду представлений у Додатку В.

3.7 Аналіз навчання та отриманих результатів

Обов’язковим етапом навчання штучного інтелекту є постійний аналіз результатів його роботи. В даному дослідженні основними метриками були кількість значення нагороди відносно кількості кроків навчання. В конфігурації середовища маємо частоту – 240 Гц, що відповідає 240 крокам на секунду. Тобто, для симуляції у, приблизно, 8 секунд маємо 240 кроків. Показники нагороди зберігались кожні 2000 кроків.

3.7.1 Аналіз навчання штучного інтелекту з даними з симуляції

Перший етап навчання був налаштований на $5.4x10^7$ кроків, що дорівнює приблизно 223м тисячам епізодів (повторень). На рисунках 3.5 та 3.6 дуже чітко видно, що оптимальне значення нагороди було отримано вже після приблизно $2.45x10^7$ кроків. Після цього штучний інтелект припинив свій розвиток, а моментами почав деградувати.

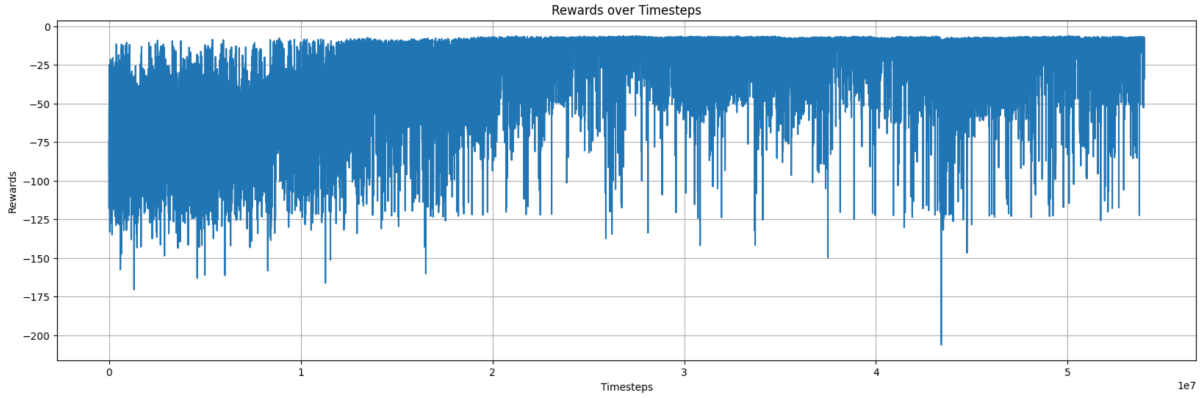

Рисунок. 3.5. Результати навчання в $5.4x10^7$ кроків

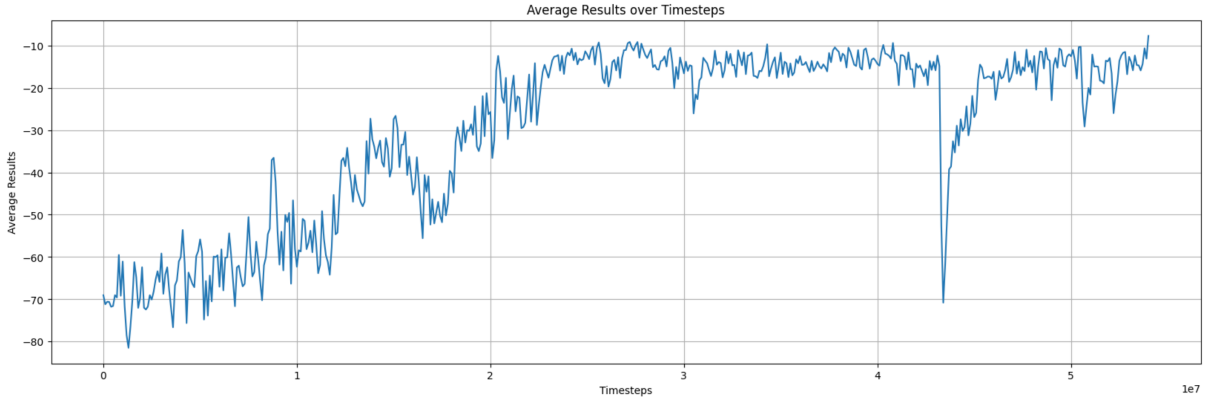

Рисунок. 3.6. Результати навчання в 5.4x10^7 кроків (усереднений результат)

Для закріплення даних результатів, та ж сама модель була запущена на ще 2.7x10^7 кроків навчання. Результати можна побачити на рисунку 3.7.

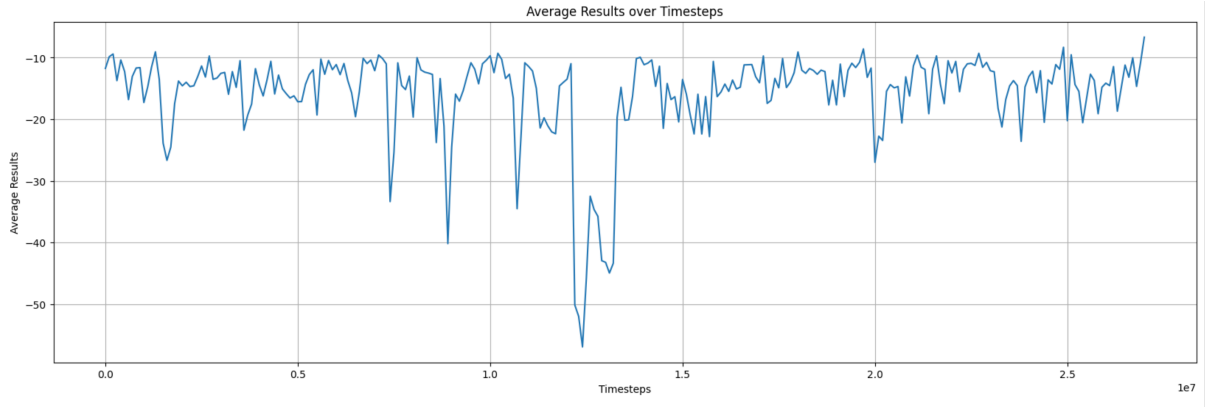

Рисунок. 3.7. Результати навчання в додаткових 2.7x10^7 кроків (усереднений результат)

Отже, як видно за на графіку – можна чітко визначити, що в даній конфігурації моделі та середовища – система досягає оптимального значення вже за 2.5x10^7 кроків.

Також важливим порівнянням буде графік відношення етапів навчання до числа успішних епізодів з 10 тестових (рисунок 3.8.). Успішним епізод вважається, якщо дрон наблизився до цілі хоча б на 2 одиниці (метра).

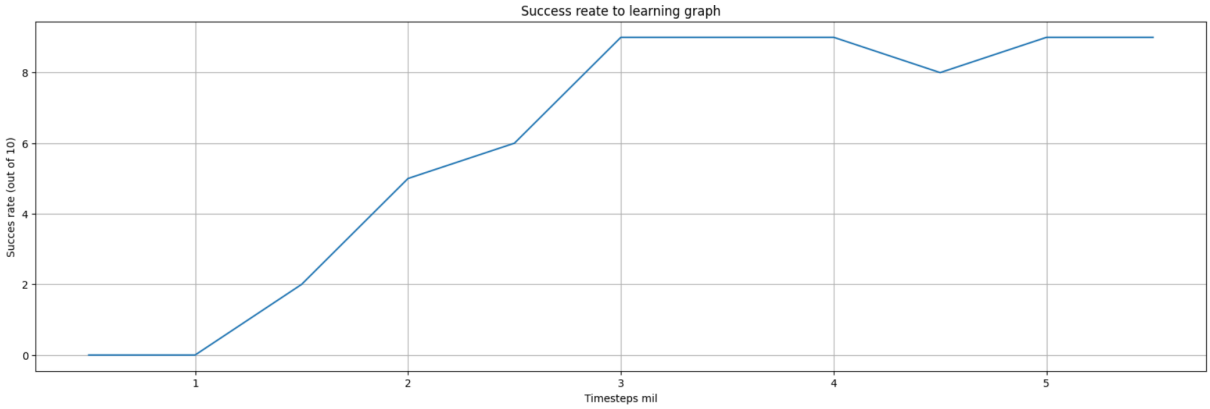

Рисунок. 3.8. Графік вдалих кількості вдалих епізодів з 10 тестових

Як видно, графік досить точно повторює графік навчання до нагороди (рисунок 3.5. та 3.6.). Отже для подальших порівнянь буде вважатись достатнім приведення графіку нагороди до кількості епізодів навчання.

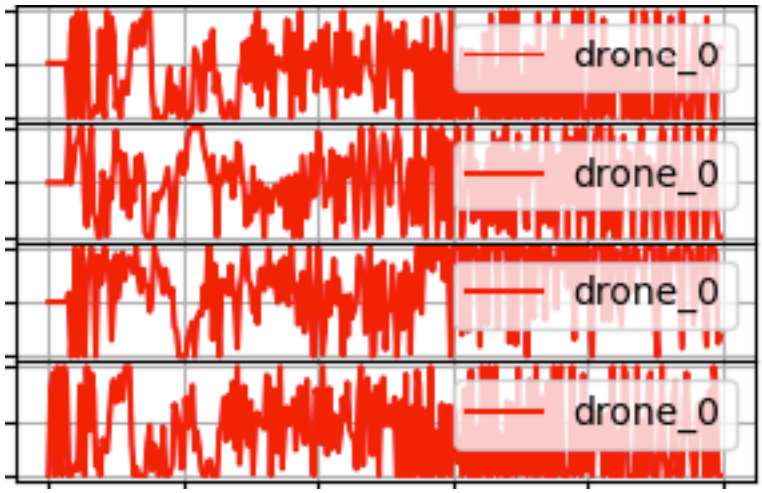

Рисунок. 3.9. Графік вихідних параметрів для контролю дрону.

На рисунку 3.9 представлені графікі контрою дрону під час польоту у середовищі після тренування. Особливо за графіками контролю обертів двигунів чітко видно, що оберти напряму контролюються штучним інтелектом без використання PID контролеру. Видно, що це не є оптимальним у контексті плавності, але є достатнім у виконанні конкретної задачі.

Для перевірки правильності використаної формули нагороди (формула 2.8) – було проведено тестове навчання моделі з модифікованою формулою (формули 3.1 – 3.4) (рисунок 3.10.)

$$R_{crash} = -200, \tag{3.1}$$

$$R_{time} = -0.005, \tag{3.2}$$

$$R_{distance>6} = -\sqrt{(x_{drone} - x_{target})^2 + (y_{drone} - y_{target})^2 + (z_{drone} - z_{target})^2}, \quad (3.3)$$

$$R_{distance<6} = \frac{10}{\sqrt{(x_{drone} - x_{target})^2 + (y_{drone} - y_{target})^2 + (z_{drone} - z_{target})^2} + 0.01}, \quad (3.4)$$

$$R_{total} = R_{distance} + R_{time} * n_{steps} + R_{crash}$$

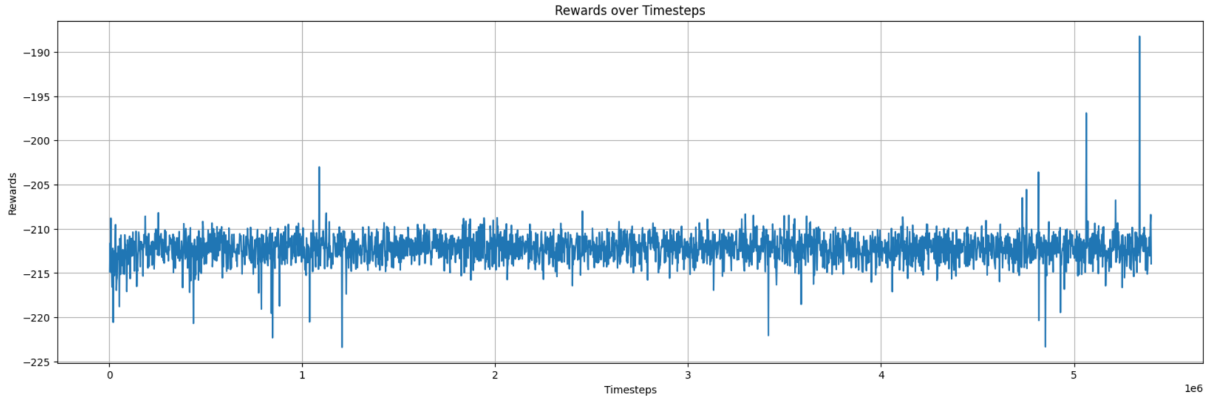

Рисунок. 3.10. Результати навчання в 5.4x10^6 кроків

За графіком видно, що формула винагороди зі змінною формулою винагороди за дистанцію, в залежності від наближення дрону до цілі не є хорошою альтернативою. Отже, усі наступні тести проводились з оригінальною формулою винагороди.

3.8 Порівняння результатів навчання з аналогічними роботами

Порівняння з аналогічними роботами є однією з найважливіших частин роботи, адже неможливо зрозуміти ефективність роботи та навчання. Однак, треба зауважити, що порівняння швидкості навчання не завжди є ефективним, адже різниця розміру штучного інтелекту, вхідних та вихідних параметрів грає дуже важливу, навіть ключову роль.

3.8.1 Порівняння з системою контролю БПЛА на основі візуальних даних для уникнення нерухомих та мобільних перешкод

Структура роботи Абудіні П. Калідас і інших [19] побудована на базі Unreal Engine 4 та AirSim.

Структура основного (контролюючого) штучного інтелекту з підсиленим навчанням включає в себе два вхідних зображення – звичайне кольорове та зображення глибини. Інші параметри дрона (кут нахилу, швидкості та ін.) не

вказані, як вхідні дані. В якості вихідних даних – кортеж з трьох елементів (x, y, z). Принцип керування зображений на рисунку 3.11.

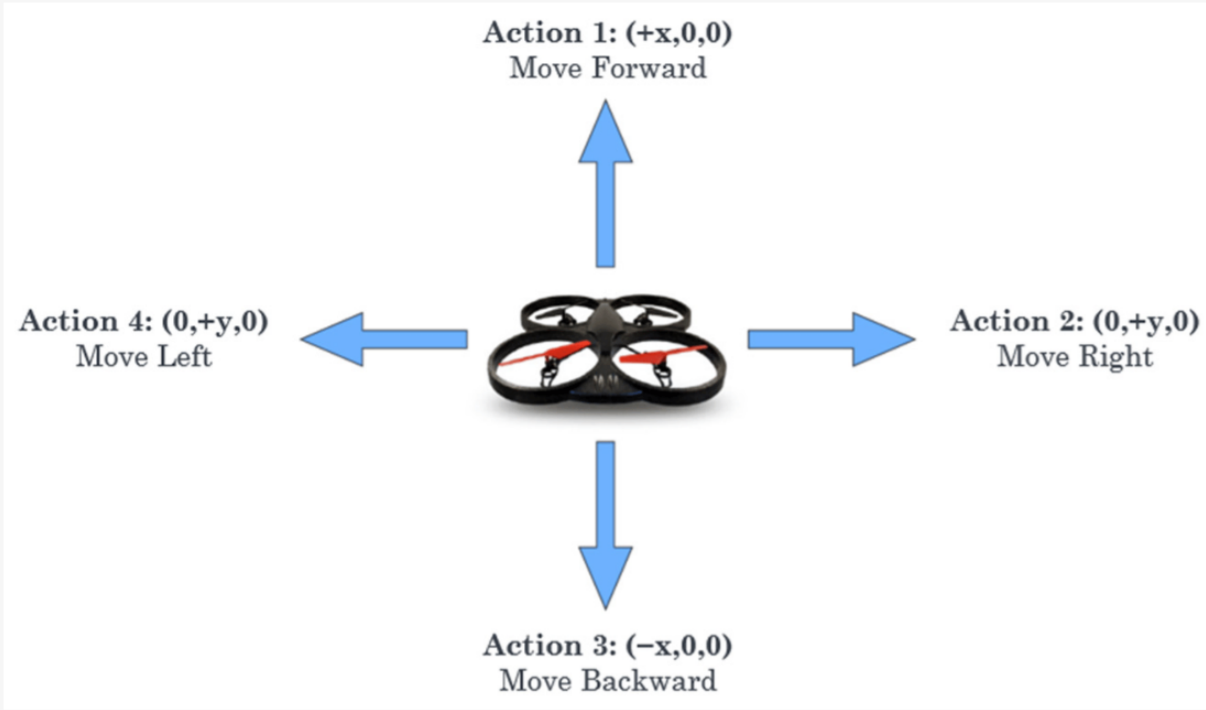

Рисунок. 3.11. Принцип керування дроном в роботі Абудіні П. Калідас та ін. [19]

Отже, комплексна система представленої роботи є трохи простішою, оскільки фізика керування дроном є обмеженою. Однак, проведення порівняння з моєю системо зайвим не буде.

Графік кроків навчання до винагороди представлений на рисунку 3.12.

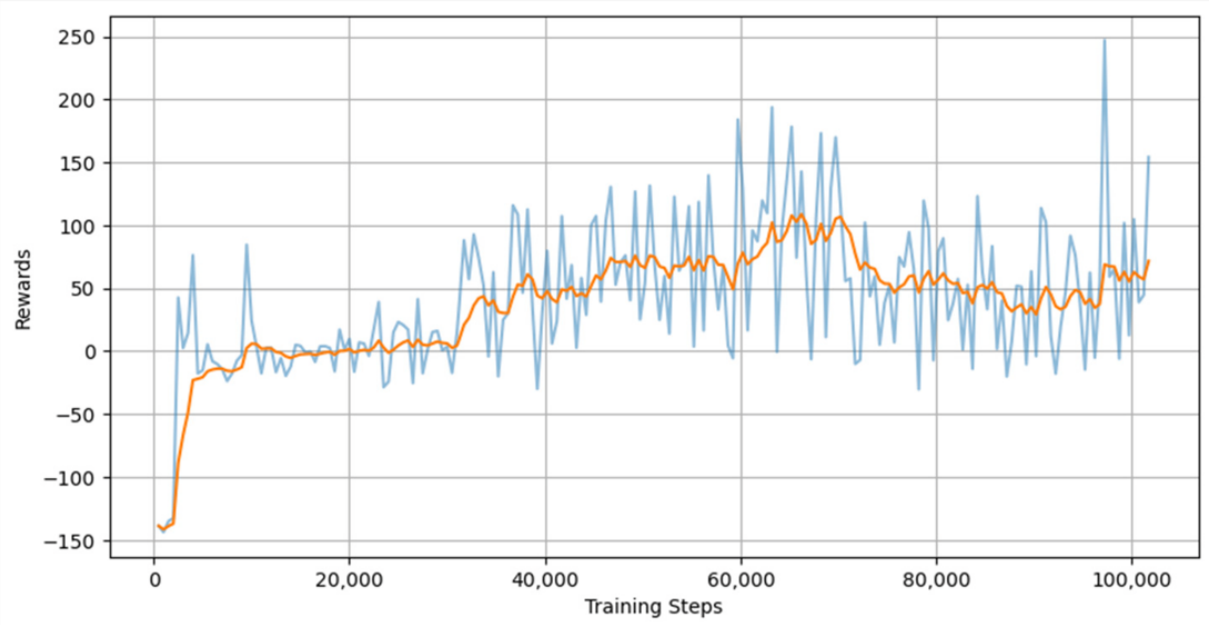

Рисунок. 3.12. Результати навчання в роботі Абудіні П. Калідас та ін. [19]

За графіком видно, що задовільних результатів моделі було досягнено вже після 68000 контролюючих кроків. В моєму випадку, задовільних результатів було досягнуто після (25000000 / 24 (частота контролю, Гц)) = 1.04 млн кроків, що є очікуваним через складність фізики на вхідних, вихідних параметрів. Цікавим є факт, що після досягнення задовільного результату (у випадку цієї роботи – винагорода близько 100) – модель почала деградувати, що корелюється з моделлю представленою в моїй роботі.

3.8.2 Порівняння з системою контролю навігації дрону за допомогою глибокого підсиленого навчання з використанням даних з датчиків

У роботі Ходж В.Д. та ін. представлена система контролю об’єкту за допомогою моделі з підсиленим навчанням. В якості вхідних даних система має зображення з камери дрону та значення компасу, яке вказує на напрям, куди треба дрону летіти. В якості вихідних – параметри для контролю дрону у середовищі. Важливо зазначити, що автори даної роботи контролюють дрон у 3D середовищі без контролю висоти (вона є сталою). Отже, фактично, дрон контролюється у 2D просторі. Це треба мати на увазі при порівнянні швидкості навчання.

Також, в представленій роботі використовується модель навчання «Curriculum Lessons», що представляє собою використання вже попередньо натренованої моделі під частину задачі. В цьому контексті модель вже вміла частково розпізнавати перешкоди. В цьому наші системи є схожими через використання YOLOv8 та MiDaS, які також є попередньо тренованими під частину моїх задач.

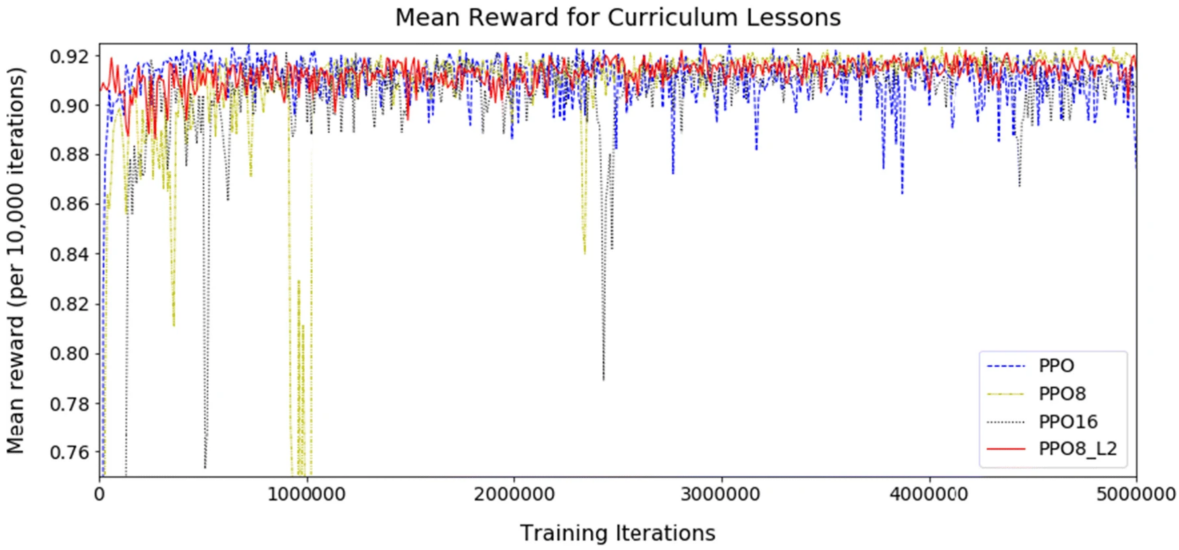

Рисунок. 3.13. Результати навчання в роботі Ходж В.Д. та ін. [20]

На рисунку 3.13. Представлені середні нагороди (на 10000 ітерацій). Через різницю контролю дронів (2D та 3D) – є доцільним порівняння партерну навчання, не зважаючи на швидкість (кількість ітерацій). Отже після аналізу представленого графіку видно, що після того, як система досягла свого локального піку - система перестала вдосконалюватись, що повністю повторює патерн моєї системи на графіку 3.6. Отже, це ще одне підтвердження, що представлена мною система навчається коректно.

3.8.3 Пряме порівняння можливості зависання дрону на певній висоті.

Для більш точного порівняння представленої мною системи з аналогами – була обрана задача підйому та зависання дрону на певній висоті, адже саме цей тип задачі часто зустрічається в аналогічних системах, що дозволяє робити пряме порівняння технологій.

Для даного аналізу було обрано дві роботи:

- Глибоке підсилене навчання для автономного контролю дрону у симуляції [21].
- Глибоке підсилене навчання для дрону. Огляд [22].

У першій роботі в якості контролюючих параметрів (вихідних даних) у цьому тесті є значення обертів усіх чотирьох двигунів (одне значення для 4х двигунів).

У другій – значення було індивідуальне для кожного з двигунів. Для більш коректного та обширного порівняння мною було проведено 2 експерименти:

- Вихідним параметром є значення обертів усіх чотирьох двигунів.
- Вихідними параметрами є значення обертів для кожного двигуна окремо.

Результати аналізу представлені на рисунку 3.14.

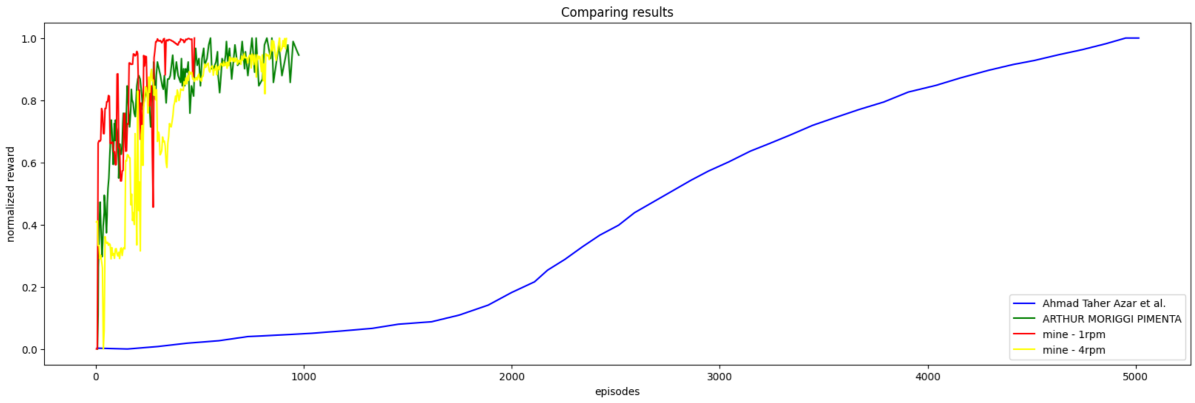

Рисунок. 3.14. Пряме порівняння з аналогами

Задля отримання більш наглядних результатів – значення винагороди було нормалізоване для кожної з кривих за формулою:

$$r_{x,normalized} = (r_x - r_{min})/(r_{max} - r_{min}), \quad (3.4)$$

де r_{min} – значення мінімальної винагороди протягом експерименту, r_{max} – значення максимальної винагороди протягом експерименту, r_x – значення винагороди для епізоду x, $r_{x,normalized}$ – значення нормалізованої винагороди.

Отже, з графіку на рисунку 3.14. Чітко видно, що система представлена мною у даній роботі навчається швидше у обох варіантах – контролю двигунів єдиним значенням, чи індивідуальними. Не дивлячись на те, що різниця з роботою Артура М. П. є досить невеликою у конкретній задачі (підйому та контролю висоти), це все ж таки дає підґрунтя для вдосконалення автономності об’єктів у три-вимірному просторі.

Висновки до розділу 3

У розділі 3 представлено деталі проекту, який зосереджується на розробці та впровадженні моделі керування об’єктом за допомогою методів підсиленого навчання. Описано використання мови програмування Python та різноманітних бібліотек для обробки даних, зображень та моделювання середовища об’єкта. Також згадується про використання спеціалізованого середовища для розробки та тестування моделі. Детально описано структуру комплексної моделі, включаючи обробку зображень з об’єкта, визначення параметрів та прогнозування дій об’єкту. Подано інформацію про налаштування та навчання моделі YOLOv8 для розпізнавання об'єктів, а також про використання моделі MiDaS для оцінки глибини зображень. В розділі також розглядаються аспекти об'єднання різних алгоритмів та моделей для ефективного керування об’єктом у тривимірному просторі. Вказано на важливість комплексного підходу в розробці системи та її масштабованість для різних сценаріїв застосування.

Також, важливою частиною розділу є аналіз представленої системи та порівняння її з найближчими аналогами. Представлена оцінка доводить, що використані модель, тип навчання та формула винагороди вибрані оптимально та демонструють задовільни результати.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП-ПРОЕКТУ

Стартап-проект, який описується в цьому розділі, представляє собою модифікація існуючих систем для автономного керування дроном, здатного слідкувати за рухомою ціллю у тривимірному просторі. Цей проект є відповіддю на зростаючу потребу у використанні безпілотних літальних апаратів для різноманітних цілей - від моніторингу та спостереження до виконання складних військових завдань, або таких як доставка вантажів або допомога у рятувальних операціях.

4.1 Маркетинговий аналіз стартап-проекту

Головна ідея проекту полягає у вдосконалені існуючого програмного забезпечення та алгоритмів для керування дронами, для виконання задач ефективного та автономного слідкування за рухомими об'єктами у тривимірному просторі. Центральним елементом системи є інтеграція алгоритмів машинного зору (YOLO, MiDaS) та підходів машинного навчання (PPO RL) для аналізу зображень, визначення координат цілі та адаптивного керування дроном.

Таблиця 4.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
вдосконалені існуючого програмного забезпечення та алгоритмів для керування дронами, для виконання задач ефективного та автономного слідкування за	Військова справа	Підвищення безпеки, посилення ППО, додатковий захист від атакуючих автономних БПЛА
	Пошуково-рятувальні операції, аграрний сектор, безпека та нагляд	Підвищення гнучкості та адаптивності дрона до різних ситуацій, оптимізація використання ресурсів

рухомими об'єктами у тривимірному просторі	Розвідувальна діяльність, моніторинг та спостереження, доставка вантажів	Забезпечення високоточного та автономного слідкування за об'єктами, підвищення ефективності та безпеки операцій
	Індустрія розваг, кіно- та відеозйомка, геодезичні роботи	Відкриття нових можливостей для використання дронів, підвищення якості та точності зйомки

Для проведення глибокого аналізу потенційних техніко-економічних переваг ідеї, планується:

- Визначення переліку техніко-економічних властивостей та характеристик ідеї.
- Визначення попередніх кола проектів-конкурентів, що вже існують на ринку та проведення збору інформації щодо їхніх техніко-економічних показників.
- Порівняльний аналіз показників: ідентифікація сильних (S), слабких (W) та нейтральних (N) сторін ідеї порівняно з конкурентами.

Таблиця 4.2.

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Проекти				W	N	S
		Мій	1	2	3			
1.	Дешевизна проекту	+	-	-	-			+
2.	Розпізнавання в складних умовах	+	+	+	-		+	

3.	Універсальність – застосування на різних платформах	+	-	-	-			+
4.	Масштабованість	+	+	+	+			+
5.	Повна автономність з моменту зльоту	+	+	-	-			+
6.	Plug-n-Play повністю готовий продукт для використання	-	+	+	+		+	
7.	Нічний зір	-	+	+	-			+

Встановлення переліку ключових сильних, слабких та нейтральних аспектів ідеї потенційного продукту є основою для його конкурентної позиції. Як видно з таблиці 4.2, основними перевагами розроблюваного продукту є його здатність до повної автономії від моменту зльоту та універсальність застосування на різних платформах, що є важливими критеріями для виділення. В порівнянні з іншими існуючими рішеннями, слабкі сторони системи включають її неповну готовність, а саме, відсутність можливості відразу ж використовувати систему (не plug-n-play), що слід врахувати при подальшому розвитку.

4.2 Технологічний аудит ідеї проекту

Таблиця 4.3.

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Комплексна система	Мова програмування Python	Є в наявності	Доступна безкоштовно

2.	Система розпізнавання об’єктів	Модель YOLOv8	Є в наявності	Доступна платно для комерційного використання
3.	Система монокулярної оцінки глибини	Модель MiDaS	Є в наявності	Доступна безкоштовно
4.	Система штучного інтелекту	Бібліотека stable-baselines3	Є в наявності	Доступна безкоштовно

Засновано на доступних технологіях, необхідних для втілення цього проекту, розробка системи автономного керування дроном для слідкування за рухомою ціллю у 3D просторі є цілком досяжною. Всі технології, які використовуються у проекті, включно з MiDaS та PPO RL, є вільно доступними та безкоштовними. Однак, YOLOv8, який є ключовою складовою системи для визначення об'єктів на зображеннях, доступний, але вимагає оплати.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 4.4.

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців	>3
2.	Загальний обсяг продаж, ум.од.	засекречено
3.	Динаміка ринку (якісна оцінка)	Зростає 60% (до 2028 р.)
4.	Наявність обмежень для входу (характер обмежень)	Фінансові затрати, наявність кваліфікованих спеціалістів, Наявність достатнього об’єму реальної інформації для навчання ШІ

5.	Специфічні вимоги до стандартизації та сертифікації	Стандартизація MIL-STD-810 та MIL-STD-461
6.	Середня норма рентабельності в галузі або по ринку, %	Від 200

На основі проведеного аналізу, ринок автономних дронів для слідкування у 3D просторі виглядає перспективним для входження з огляду на високу потенційну рентабельність, обмежену кількість конкурентів та очікуване зростання ринку до 2028 року. Для глибшого вивчення ринкових можливостей необхідно визначити потенційні групи клієнтів та їхні вимоги до продукту. Детальна характеристика цих груп та перелік вимог до стартап-проекту з модифікації автономності об’єкту у три вимірному просторі представлені в таблиці 4.5.

Таблиця 4.5.

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги до споживачів продукту
1.	Стрімке посилення військової напруги	Військові державні установи	Різний рівень військового технологічного розвитку	До продукції: - Надійність; - Безпека; - Засекреченість До компанії-постачальника: - Надійність - Репутація - Постійна підтримка у
2.	Підвищення надійності ППО			
3.	Можливість контролювати повітряний простій			

4.	Стійкість до великої кількості відносно дешевих ворожих БПЛА			користуванні продукту - Постійне вдосконалення
7.	Індустрія розваг	Івент агенції	Розмір компанії	

Наступним кроком необхідно провести аналіз потенційного ринку. Для цього створимо таблиці факторів загроз (таблиця 4.6) та стимулюючих факторів (таблиця 4.7).

Таблиця 4.6.

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Недостатня репутація нової компанії	Потенційні конкуренти – компанії, що вже мають ім’я.	Кооперування з іншими компаніями, які вже мають доступ до ринку.
2.	Незацікавленість потенційних клієнтів у продукті	Недовіра до технології. Часто щоб зацікавитись – треба відчути потребу в реальних ситуаціях.	Демонстрації, надання тестових екземплярів. Якісний пітчинг системи.
3.	Недостатній рівень конкуренто-спроможності	Рішення існуючих конкурентів можуть задовольняти базові потреби клієнтів	Вдосконалення цінової політики, кооперація з компаніями, працюючими з державою
4.	Недостатні випробування	Важкість реалізації реальних іспитів	Пошук додаткових

	продукту в реальних умовах		шляхів іспитів, кооперація з компаніями, які працюють в даній сфері
5.	Складність отримання сертифікатів	Дуже складна і довго тривала процедура сертифікації	Використання послуг спеціальних компаній

Таблиця 4.7.

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1.	Залучення інвестицій	Наявність стабільної MVP версії, чіткого бізнес плану та бачення майбутнього технології	Збільшення компанії у всіх аспектах, вихід на реальний ринок
2.	Кооперування з іншими компаніями	Більше можливостей для тестування та можливість легшого виходу на ринок завдяки імені компанії - партнера	Облегшений вихід на реальний ринок
3.	Проведення широких реальних іспитів	Можливість легшого представлення технології потенційним клієнтам.	Зміцнення позиції компанії на ринку.

Також необхідно проаналізувати риси конкуренції (таблиця 4.8.).

Таблиця 4.8.

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: *олігополія*	На ринку присутні декілька великих гравців, які пропонують рішення для автономного керування дронами	Розробка інноваційного та високотехнологічного продукту, що включає унікальні функції та переваги над конкурентами, активна маркетингова стратегія
За рівнем конкурентної боротьби: *глобальний*	Рішення для автономних дронів затребувані на світовому ринку	Розробка глобальної стратегії виходу на ринки різних країн, адаптація продукту до міжнародних стандартів
За галузевою ознакою: *внутрішньо-галузева*	Конкуренція в середині галузі автономних дронів	Спеціалізація на певних аспектах керування дронами, які є ключовими для певних ринкових сегментів
Товарно-родова конкуренція	Наявність продуктів-замінників, що можуть виконувати схожі функції	Наголос на унікальних рисах та перевагах продукту, які відсутні у замінників
Нецінова конкуренція	Ринкова боротьба здійснюється через якість та характеристики продукту, а не ціну	Постійний розвиток та удосконалення продукту, включаючи його функціональність та надійність
Не марочна інтенсивність	Споживачів більше цікавлять технічні	Зосередитися на якості продукту та його технічних

	характеристики та функціональність продукту, а не бренд	характеристиках, щоб вирізнятися серед конкурентів

Здійснимо ретельний аналіз конкурентного середовища у секторі, використовуючи модель п'яти конкурентних сил за Майклом Портером. Майкл Портер виділяє п'ять ключових елементів, які впливають на привабливість ринку з огляду на конкурентні умови, включаючи:

• існуючих конкурентів у галузі;

• потенційних конкурентів;

• наявність альтернативних товарів;

• постачальників, які змагаються за вплив на ринку;

• споживачів (аналогічно).

Міцні позиції продукту у відповідності до цих факторів вказують на його здатність генерувати необхідний обіг капіталу та впливати на інших учасників ринку, встановлюючи умови для співпраці. Особливості цих факторів варіюються залежно від специфіки галузі та з часом можуть змінюватися.

Таблиця 4.9.

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Постачальники	Клієнти	Продукти-замінники
	Switchblade, UVision Hero, Iron Dome	Baykar, IAI	IAI, AeroVironment, UVision	Військові структури держав	Surface-to-air defence, air-to-air defence
Висновки	Присутня конкуренція на міжнародному	Компанії займаються активною	Постачальниками є компанії з	Клієнтами ж державні структури,	Є не повні аналоги системи, які

	ринку. Конкуренція сильна, кожна з компаній має хорошу репутацію	розробкою та вдосконаленням технології	міжнародним ім’ям	які є повільними та високовим огливими	можуть замінити представлені технології вищою ціною

На підставі даних таблиці приходимо до висновку, що участь у ринку є доцільною з огляду на сучасну ринкову ситуацію. На цьому етапі конкуренція недостатньо інтенсивна, постачальники мало зацікавлені у пропонуванні подібних продуктів, а існуючі рішення відтворюють лише частково функціонал, запропонований нашим проектом.

За результатами аналізу конкурентного середовища (таблиця 4.9), враховуючи особливості нашої проектної ідеї (таблиця 4.2), потреби споживачів (таблиця 4.5), а також фактори маркетингового середовища (таблиці 4.6-4.7), визначаємо та обґрунтовуємо ключові чинники конкурентоспроможності (таблиця 4.10).

Таблиця 4.10.

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Дешевизна проекту	Проект на основному етапі створення MVP не потребує великого штату працівників, отже собівартість MVP може бути нижчою в порівнянні

2.	Універсальність – застосування на різних платформах	Проект з самого початку розроблявся як універсальний, який дозволить встановлення технології на різні платформи, особливо за допомогою ROS
3.	Повна автономність з моменту зльоту	В технології передбачена повна автономність з моменту зльоту, що унеможливлює потенційне подавлення контролюючого сигналу дрону.

Наступним кроком буде аналіз слабких та сильних сторін технології (таблиця 4.11.).

Таблиця 4.11.

Порівняльний аналіз сильних та слабких сторін стартап-проекту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з продуктом						
			-3	-2	-1	0	+1	+2	+3
1.	Дешевизна проекту	16		✓					
2.	Універсальність застосування на різні платформах	16		✓					
3.	Повна автономність з моменту зльоту	10						✓	

Необхідним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 4.12.) на основі виділених ринкових загроз та можливостей, а також сильних та слабких сторін.

Таблиця 4.12.

SWOT- аналіз стартап-проекту

Сильні сторони: 1. Дешевизна проекту 2. Розпізнавання в складних умовах 3. Універсальність застосування на різних платформах 4. Масштабованість 5. Повна автономність з моменту зльоту	**Слабкі сторони:** 1. Наявність вже існуючих конкуруючих рішень 2. Затрата великого об'єму часу для виходу на ринок 3. Відсутність нічного зору 4. Повністю готовий продукт 5. «Заточеність» під вузьке коло цілей
Можливості: 1. Партнерство з компаніями, які вже на ринку 2. Отримання інвестицій 3. Вдосконалення технології з моменту початку використання у реальних умовах	**Загрози:** 1. Занадто повільний вихід на ринок 2. Критична недостатність реальних тестувань

На основі SWOT-аналізу приводяться альтернативи ринкової поведінки для виведення стартап проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (таблиця 4.13.).

Таблиця 4.13.

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Терміни реалізації
1.	Відкриття проекту та розповсюдження в рамках opensource	Низька. Є ризик загублення технології через її універсальність	Одразу після розробки

	ліцензії		
2.	Розповсюдження проекту за допомогою спеціалізованих виставок	Висока, адже майже усі популярні на даний момент компанії починали з таких демонстрацій.	2 роки
3.	Участь у тендерах	Середня, так як компаніям без гучного імені важко пробитись у державний сектор	2 роки

4.4 Розробка ринкової стратегії стартап-проекту

Перш за все треба визначати стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14.

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
1.	Військові сектори держав	Готові сприйняти продукт	Середній	Висока	Складна
2.	Приватні військові компанії	Готові сприйняти продукт	Середній	Висока	Складна
3.	Івент агенції	Не готові сприйняти продукт	Низький	Середній	Складна

4.	Аграрні компанії	Готові сприйняти продукт	Високий	Високий	Проста
Обрана цільова група: військові сектори держав.					

Згідно з аналізом потенційних комерційних груп споживачів на запропонований продукт, було обрано стратегію точкового маркетингу, оскільки компанія може отримати дуже велику вигоду від цього пулу клієнтів.

Таблиця 4.15.

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Стратегія вдосконалення	Точкова стратегія	Вдосконалення технології, реалізація польових іспитів, набуття репутації	Стратегія точкового маркетингу

Таблиця 4.16.

Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні	Компанія буде намагатись притягнути увагу	Ні, у компанії є бачення свого	Стратегія виклику лідера

		існуючих користувачів, оскільки пул користувачів є обмежений і повільно зростаючим	власного рішення існуючої проблеми	

Виходячи з вимог клієнтів із вибраних ринкових сегментів до постачальника (у нашому випадку, стартапу) та вимог до самого продукту, відображених у таблиці 4.5, а також з огляду на базову стратегію розвитку компанії (таблиця 4.15) та обрану стратегію взаємодії з конкурентами (таблиця 4.16), буде сформовано стратегію позиціонування на ринку (див. таблицю 4.17). Ця стратегія передбачає створення унікального комплексу асоціацій з брендом або проектом, які допоможуть споживачам впізнавати та відрізняти нашу торговельну марку.

Таблиця 4.17.

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1.	Вирішення проблеми захисту неба від ворожих автономних БПЛА	Стратегія точкового маркетингу	Дешевизна, універсальність	Дешевизна Надійність Автономність Клієнто-орієнтованість

В результаті аналізу була розроблена узгоджена стратегія ринкової діяльності для стартап-компанії, яка вказує основні напрямки її діяльності на ринку.

4.5 Розробка маркетингової програми стартап-проекту

Спочатку треба сформувати маркетингову концепцію продукту, який отримає потенційний клієнт (таблиця 4.18.).

Таблиця 4.18.

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Вирішення проблеми захисту від ворожих БПЛА	Альтернативну технологію для захисту	Невелика, автономна технологія
2.	Здешевлення захисту неба	Універсальну технологію, яка є кросс-платформною та недорогою	Дешева технологія, яка може бути встановлена на різні платформи
3.	Активна підтримка та зворотній зв’язок	Швидка та активна підтримка з боку розробника	Активна допомога у використані

Далі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його

надання.

Таблиця 4.19.

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="2">Кросс - платформна технологія для підвищення мобільності та автономності об’єктів у три вимірному просторі, у тому числі дронів</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Розмір</td></tr>
<tr><td>1. Комплексна система контролю об’єктів</td><td>250МБ</td></tr>
<tr><td colspan="2">Якість: обробники виключень, логування</td></tr>
<tr><td colspan="2">Пакування: комплексний скрипт</td></tr>
<tr><td colspan="2">Марка: назва організації-розробника + назва товару</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу: ліцензія на використання</td></tr>
<tr><td colspan="2">Після продажу: надання підтримки</td></tr>
<tr><td colspan="3">За рахунок чого потенційний товар буде захищено від копіювання:
За рахунок постійного розвитку технології, також за рахунок патентів</td></tr>
</table>

Таблиця 4.20.

Визначення меж встановлення ціни

№	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1.	Від 60000$ шт	Від 80000$ шт	-	3000-16000/міс.

Таблиця 4.21.

Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Покупка ліцензії на технологію або договір про надання послуг по використанню технології без надання ПЗ	Підтримка то допомога в налаштуванні, використанні	Канал нульового рівня	Тендери, прямі контракти з клієнтами

Таблиця 4.22.

Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікації, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Купують	Виставки,	Представлення	Підготовка	Демонстраці

	товар для постійної наявності	тендери, закриті кола комунікації	на виставках	якісної презентації	я функціоналу, обґрунтування необхідності наявності даної технології.

Висновки до розділу 4

Розділ четвертий присвячений розгортанню стартапу, заснованого на розробленій системі для створення трафіка в програмно-конфігурованих мережах, використовуючи лямбда-архітектуру. У рамках створення цього стартапу були виконані наступні кроки:

- Описано основну концепцію стартапу, його функціональні можливості, визначено ключових конкурентів та проведено порівняльний аналіз із наявними продуктами на ринку для виявлення переваг нового продукту.
- Проведено аналіз ринкових умов для впровадження проекту та його успішного виходу на ринок. Виділено потенційні можливості та загрози для стартапу та розроблено план дій для їх уникнення.
- Здійснено порівняльне дослідження перспектив стартапу на фоні конкурентів, встановлено стратегію розвитку продукту на ринку, визначено цільові аудиторії та методи просування проекту.
- Сформульовано маркетинговий план для популяризації технології, описано основні канали дистрибуції, визначено ключові цільові групи та маркетингові стратегії для розширення бази клієнтів.

У підсумку, було отримано стартап-проект для запуску розробленої технології яка виникла на основі дипломного проекту, з метою виходу на ринок.

ВИСНОВКИ

Дана магістерська дисертація присвячена розробці методології для підвищення автономії об'єкта, в 3D просторі шляхом інтеграції передових технологій штучного інтелекту. Дослідження в першу чергу спрямоване на адаптацію та навчання алгоритму YOLOv8 для визначення об'єктів під час польоту, налаштування моделі MiDaS для оцінки глибини зображень та впровадження алгоритму глибокого підсиленого навчання для динамічного управління об’єктом.

Значну частину дисертації складала робота з вивчення різних моделей штучного інтелекту, які підходять для обробки зображень та автономного управління. Після ретельних досліджень були обрані YOLOv8 та MiDaS через їх ефективність та точність у визначенні об'єктів та сприйнятті глибини, відповідно. Додатково, модель проксимальної оптимізації політики (PPO) була вибрана через її ефективність у завданнях підсиленого навчання. Практичні аспекти цих моделей були досліджені та адаптовані для відповідності унікальним вимогам управління об’єктом в 3D просторі.

Для симуляції та тестування цих моделей використовувалось середовище gym-pybullet-drones, побудоване на платформі Gym від OpenAI. Це середовище надало реалістичну та універсальну платформу для експериментів з системами управління об’єктом та оцінки продуктивності інтегрованих моделей штучного інтелекту.

Методологія дослідження включала в себе вичерпне тестування та аналіз, включаючи створення спеціалізованого коду та використання спеціалізованих середовищ для симуляції. Робота також передбачала візуалізацію результатів для оцінки ефективності запропонованої системи.

Заключно, ця дисертація демонструє практичну можливість та масштабованість запропонованих методологій. Інтеграція передових технологій штучного інтелекту представляє собою значну роботу у галузі автономного управління об’єктами у три вимірному просторі. Розроблені методи та здобуті

висновки вносять суттєвий вклад у загальний дискурс щодо підтримки автономії за допомогою штучного інтелекту в 3D просторі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sultana, F., Sufian, A., & Dutta, P. A review of object detection models based on convolutional neural network. Intelligent Computing: Image Processing Based Applications, arXiv:1905.01614, 2020

2. Where to Start. Home - Ultralytics YOLOv8 Docs. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ultralytics.com/#where-to-start

3. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640, 2015

4. Lampert, C., Blaschko, M. and Hofmann, T. (2008) Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, 24-26 June 2008, 1-8. https://doi.org/10.1109/CVPR.2008.4587586

5. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 779-788.

6. Ultralytics - YOLO. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ultralytics/ultralytics

7. Brady Zhou, Philipp Krähenbühl, Vladlen Koltun. Does computer vision matter for action? arXiv:1905.12887, 2019

8. René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, Vladlen Koltun. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer. arXiv:1907.01341

9. Yingxiu Chang, Yongqiang Cheng, Umar Manzoor, John Murray, A review of UAV autonomous navigation in GPS-denied environments, Robotics and Autonomous Systems, Volume 170, 2023, 104533, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2023.104533

10. S. Rezwan and W. Choi, "Artificial Intelligence Approaches for UAV Navigation: Recent Advances and Future Challenges," in IEEE Access, vol. 10, pp. 26320-26339, 2022, doi: 10.1109/ACCESS.2022.3157626.

11. J. Li, D. Ye, M. Kolsch, J. Wachs and C. Bouman, "Fast and Robust UAV to UAV Detection and Tracking from Video" in IEEE Transactions on Emerging Topics in Computing. doi: 10.1109/TETC.2021.3104555 url: https://doi.ieeecomputersociety.org/10.1109/TETC.2021.3104555

12. Zhu, Pengfei and Wen, Longyin and Du, Dawei and Bian, Xiao and Fan, Heng and Hu, Qinghua and Ling, Haibin. Detection and tracking meet drones challenge - [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/VisDrone/VisDrone-Dataset

13. Yueru Chen, Pranav Aggarwal, Jongmoo Choi, C.-C. Jay Kuo. A Deep Learning Approach to Drone Monitoring. arXiv:1712.00863, 2017

14. Image Augmentation: Make it rain, make it snow. How to modify photos to train self-driving cars. [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/image-augmentation-make-it-rain-make-it-snow-how-to-modify-a-photo-with-machine-learning-163c0cb3843f/

15. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. arXiv:1512.02325, 2016

16. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497, 2016

17. Rubber Duck Detection Image Dataset. [Електронний ресурс] – Режим доступу до ресурсу: https://universe.roboflow.com/university-of-liverpool-od161/rubber-duck-detection-atuez/dataset/1

18. Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, Angela P. Schoellig. Learning to Fly -- a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. arXiv:2103.02142, 2021

19. Kalidas, A.P.; Joshua, C.J.; Md, A.Q.; Basheer, S.; Mohan, S.; Sakri, S. Deep Reinforcement Learning for Vision-Based Navigation of UAVs in Avoiding Stationary and Mobile Obstacles. Drones 2023, 7, 245. https://doi.org/10.3390/drones7040245

20. Hodge, V.J., Hawkins, R. & Alexander, R. Deep reinforcement learning for drone navigation using sensor data. Neural Comput & Applic 33, 2015–2033 (2021). https://doi.org/10.1007/s00521-020-05097-x

21. Arthur Moriggi Pimenta. Deep reinforcement learning for Simulated Autonomous Drone Control/ A. M. Pimenta - São Paulo, 2020-65 p. : il. (algumas color.)

22. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone Deep Reinforcement Learning: A Review. Electronics 2021, 10, 999. https://doi.org/10.3390/electronics10090999

ДОДАТОК А

Метод підвищення автономності об’єкту у 3D просторі

ЛІСТИНГ ПРОГРАМНОГО КОДУ АУГМЕНТАЦІЇ ДАНИХ ДЛЯ YOLOv8

Аркушів 10

Київ 2023

```
import cv2
import os
import glob
import random
import numpy as np
from imgaug import augmenters as iaa
import imgaug as ia
```

Головна функція:

```
def main():
  old_image_folder =
  old_annotation_folder =
  new_image_folder =
  new_annotation_folder =
  dataset = load_dataset(image_folder=old_image_folder,
annotation_folder=old_annotation_folder)

  augmented_dataset = random_non_symmetrical_scaling(dataset=dataset,
scale_percentage=50) # Scale
  augmented_dataset.extend(random_rotation(dataset= augmented_dataset +
dataset, rotation_percentage=50)) # Rotation
  augmented_dataset.extend(random_lighting_change(dataset= augmented_dataset
+ dataset, change_percentage=50)) # Lighting change
  augmented_dataset.extend(random_color_channel_mix(dataset=
augmented_dataset + dataset, mix_percentage=50)) # Color channel mix
  augmented_dataset.extend(apply_histogram_equalization(dataset=
augmented_dataset + dataset, equalization_percentage=50)) # Histogram change
  augmented_dataset.extend(add_artificial_rain(dataset= augmented_dataset +
dataset, rain_percentage=50)) # Rain
  augmented_dataset.extend(add_artificial_fog(dataset= augmented_dataset +
dataset, fog_percentage=50)) # Fog

  new_dataset = dataset + augmented_dataset
  save_dataset(new_dataset, image_folder=new_image_folder,
annotation_folder=new_annotation_folder)
```

Завантаження існуючого датасету:

```
def load_dataset(image_folder, annotation_folder):
    """
    Load images and their corresponding bounding box annotations.

    :param image_folder: Path to the folder containing images.
    :param annotation_folder: Path to the folder containing bounding box
annotations.
    :return: A list of tuples, each containing an image and its bounding box
data.
    """
    dataset = []

    # Find all image files in the image folder
```

```
    image_files = glob.glob(os.path.join(image_folder, "*.jpg"))

    for image_file in image_files:
        # Derive the corresponding annotation file path
        base_name = os.path.basename(image_file)
        annotation_file = os.path.join(annotation_folder,
os.path.splitext(base_name)[0] + ".txt")

        # Read the image
        image = cv2.imread(image_file)
        if image is None:
            continue

        # Read the annotation file
        if os.path.exists(annotation_file):
            with open(annotation_file, "r") as file:
                bounding_boxes = file.readlines()
        else:
            bounding_boxes = []

        # Add the image and its bounding boxes to the dataset
        dataset.append((image, bounding_boxes))

    return dataset
```

Збереження нового датасету:

```
def save_dataset(dataset, image_folder, annotation_folder):
    """
    Save the updated dataset to specified folders for images and
annotations.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param image_folder: Path to the folder where images will be saved.
    :param annotation_folder: Path to the folder where annotations will be
saved.
    """
    for i, (image, annotations) in enumerate(dataset):
        # Define file paths
        image_file = os.path.join(image_folder, f"image_{i}.jpg")
        annotation_file = os.path.join(annotation_folder, f"image_{i}.txt")

        # Save the image
        cv2.imwrite(image_file, image)

        # Save the annotations
        with open(annotation_file, 'w') as file:
            for annotation in annotations:
                file.write(annotation + '\n')
```

Випадкове а-симетричне масштабування:

```
def random_non_symmetrical_scaling(dataset, scale_percentage):
    """
    Randomly scale images non-symmetrically in the dataset.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param scale_percentage: Percentage of images to be scaled.
    :return: A new list of tuples with scaled images and their annotations.
    """
    scaled_dataset = []
    num_images = len(dataset)
    num_images_to_scale = int(num_images * scale_percentage / 100)

    # Randomly choose indices of images to scale
    indices_to_scale = random.sample(range(num_images), num_images_to_scale)

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_scale:
            # Randomly choose scale factors and a center point
            scale_factor_x, scale_factor_y = random.uniform(0.5, 1.5),
random.uniform(0.5, 1.5)
            center_x, center_y = random.randint(0, image.shape[1]),
random.randint(0, image.shape[0])

            # Apply scaling
            M = cv2.getRotationMatrix2D((center_x, center_y), 0,
max(scale_factor_x, scale_factor_y))
            scaled_image = cv2.warpAffine(image, M, (image.shape[1],
image.shape[0]))

            # Adjust bounding boxes
            scaled_bounding_boxes = []
            for box in bounding_boxes:
                box_data = list(map(float, box.split()))
                # Adjust bounding box coordinates and dimensions
                # Assuming the YOLO format (class_id, x_center, y_center,
width, height)
                box_data[1] = (box_data[1] * image.shape[1] - center_x) *
scale_factor_x + center_x
                box_data[2] = (box_data[2] * image.shape[0] - center_y) *
scale_factor_y + center_y
                box_data[3] *= scale_factor_x  # width
                box_data[4] *= scale_factor_y  # height

                # Clip bounding boxes to image boundaries
                box_data[1] = max(0, min(box_data[1], image.shape[1]))
                box_data[2] = max(0, min(box_data[2], image.shape[0]))
                box_data[3] = max(0, min(box_data[3], image.shape[1] -
box_data[1]))
```

```
                box_data[4] = max(0, min(box_data[4], image.shape[0] -
box_data[2]))

                scaled_bounding_boxes.append(' '.join(map(str, box_data)))

            scaled_dataset.append((scaled_image, scaled_bounding_boxes))

    return scaled_dataset
```

Випадкове обертання:

```
def random_rotation(dataset, rotation_percentage):
    """
    Randomly rotate images in the dataset.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param rotation_percentage: Percentage of images to be rotated.
    :return: A new list of tuples with rotated images and their annotations.
    """
    rotated_dataset = []
    num_images = len(dataset)
    num_images_to_rotate = int(num_images * rotation_percentage / 100)

    # Randomly choose indices of images to rotate
    indices_to_rotate = random.sample(range(num_images),
num_images_to_rotate)

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_rotate:
            # Randomly choose a rotation angle
            angle = random.uniform(-180, 180)

            # Rotate the image
            height, width = image.shape[:2]
            M = cv2.getRotationMatrix2D((width / 2, height / 2), angle, 1)
            rotated_image = cv2.warpAffine(image, M, (width, height))

            # Rotate the bounding boxes
            rotated_bounding_boxes = []
            for box in bounding_boxes:
                box_data = list(map(float, box.split()))
                # Assuming the YOLO format (class_id, x_center, y_center,
width, height)
                x_center, y_center = box_data[1] * width, box_data[2] *
height

                # Rotate point around image center
                new_x = M[0][0] * x_center + M[0][1] * y_center + M[0][2]
                new_y = M[1][0] * x_center + M[1][1] * y_center + M[1][2]
```

```
                box_data[1], box_data[2] = new_x / width, new_y / height

                rotated_bounding_boxes.append(' '.join(map(str, box_data)))

            rotated_dataset.append((rotated_image, rotated_bounding_boxes))
        # else:
        #     rotated_dataset.append((image, bounding_boxes))

    return rotated_dataset
```

Випадкова зміна освітленності:

```
def random_lighting_change(dataset, change_percentage):
    """
    Randomly change the lighting of images in the dataset.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param change_percentage: Percentage of images to have their lighting
changed.
    :return: A new list of tuples with lighting-changed images and their
original annotations.
    """
    changed_dataset = []
    num_images = len(dataset)
    num_images_to_change = int(num_images * change_percentage / 100)

    # Randomly choose indices of images to change
    indices_to_change = random.sample(range(num_images),
num_images_to_change)

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_change:
            # Randomly choose brightness and contrast changes
            brightness_factor = random.uniform(0.5, 1.5)
            contrast_factor = random.uniform(0.5, 1.5)

            # Apply brightness and contrast changes
            changed_image = cv2.convertScaleAbs(image,
alpha=contrast_factor, beta=brightness_factor)

            changed_dataset.append((changed_image, bounding_boxes))
        # else:
        #     changed_dataset.append((image, bounding_boxes))

    return changed_dataset
```

Випадкове змішування каналів кольорів:

```
def random_color_channel_mix(dataset, mix_percentage):
    """
    Randomly mix the color channels of images in the dataset.
```

```
    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param mix_percentage: Percentage of images to have their color channels
mixed.
    :return: A new list of tuples with color channel-mixed images and their
original annotations.
    """
    mixed_dataset = []
    num_images = len(dataset)
    num_images_to_change = int(num_images * mix_percentage / 100)

    # Randomly choose indices of images to mix
    indices_to_change = random.sample(range(num_images),
num_images_to_change)

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_change:
            # Convert the tuple of channels to a list, shuffle it, then
convert back to tuple
            channels = list(cv2.split(image))
            random.shuffle(channels)
            mixed_image = cv2.merge(channels)

            mixed_dataset.append((mixed_image, bounding_boxes))
        else:
            mixed_dataset.append((image, bounding_boxes))

    return mixed_dataset
```

Випадкова еквалізація гістограм зображень:

```
def apply_histogram_equalization(dataset, equalization_percentage):
    """
    Apply histogram equalization to a subset of images in the dataset.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param equalization_percentage: Percentage of images to apply histogram
equalization.
    :return: A new list of tuples with equalized images and their original
annotations.
    """
    equalized_dataset = []
    num_images = len(dataset)
    num_images_to_change = int(num_images * equalization_percentage / 100)

    # Randomly choose indices of images to equalize
    indices_to_change = random.sample(range(num_images),
num_images_to_change)
```

```
    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_change:
            # Check if the image is grayscale or color
            if len(image.shape) == 2 or image.shape[2] == 1:
                # Grayscale image
                equalized_image = cv2.equalizeHist(image)
            else:
                # Color image, equalize each channel
                channels = cv2.split(image)
                equalized_channels = [cv2.equalizeHist(channel) for channel
in channels]
                equalized_image = cv2.merge(equalized_channels)

            equalized_dataset.append((equalized_image, bounding_boxes))
        # else:
        #     equalized_dataset.append((image, bounding_boxes))

    return equalized_dataset
```

Випадкове додавання ефекту дощу:

```
def add_artificial_rain(dataset, rain_percentage):
    """
    Add artificial rain to a subset of images in the dataset.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param rain_percentage: Percentage of images to have artificial rain
added.
    :return: A new list of tuples with rain-added images and their original
annotations.
    """
    rain_added_dataset = []
    num_images = len(dataset)
    num_images_to_change = int(num_images * rain_percentage / 100)

    # Randomly choose indices of images to add rain
    indices_to_change = random.sample(range(num_images),
num_images_to_change)

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_change:
            # Generate rain effect
            rain_image = generate_rain(image.shape)

            # Overlay rain on the original image
            rain_added_image = cv2.addWeighted(image, 1, rain_image, 0.5, 0)

            rain_added_dataset.append((rain_added_image, bounding_boxes))
        # else:
        #     rain_added_dataset.append((image, bounding_boxes))
```

```
    return rain_added_dataset

def generate_rain(shape):
    """
    Generate an artificial rain effect image of a given shape.

    :param shape: Shape of the image (height, width, channels).
    :return: Image with rain effect.
    """
    rain_image = np.zeros(shape, dtype=np.uint8)
    number_of_drops = random.randint(100, 1000)

    for _ in range(number_of_drops):
        # Randomly set the position, size and direction of the raindrops
        length = random.randint(10, 20)
        width = random.randint(1, 2)
        angle = random.randint(-10, 10)

        x1 = random.randint(0, shape[1])
        y1 = random.randint(0, shape[0] - length)
        x2 = int(x1 + length * np.sin(np.radians(angle)))
        y2 = int(y1 + length * np.cos(np.radians(angle)))

        cv2.line(rain_image, (x1, y1), (x2, y2), (255, 255, 255), width)

    return rain_image
```

Випадкове додавання ефекту туману:

```
def add_artificial_fog(dataset, fog_percentage):
    """
    Add artificial fog to a subset of images in the dataset using imgaug's
CloudLayer.

    :param dataset: A list of tuples, each containing an image and its
bounding box data.
    :param fog_percentage: Percentage of images to have artificial fog
added.
    :return: A new list of tuples with fog-added images and their original
annotations.
    """
    fog_added_dataset = []
    num_images = len(dataset)
    num_images_to_change = int(num_images * fog_percentage / 100)

    # Randomly choose indices of images to add fog
    indices_to_change = random.sample(range(num_images),
num_images_to_change)

    # Define the fog augmenter
```

```
    fog_augmenter = iaa.CloudLayer(
        intensity_mean=0.2,  # Adjust the intensity of the fog
        intensity_freq_exponent=-2.5,
        intensity_coarse_scale=10,
        alpha_min=0.1,
        alpha_multiplier=0.6,
        alpha_size_px_max=2,
        alpha_freq_exponent=-2.5,
        sparsity=0.9,
        density_multiplier=0.7
    )

    for idx, (image, bounding_boxes) in enumerate(dataset):
        if idx in indices_to_change:
            # Apply fog effect
            fog_added_image = fog_augmenter.augment_image(image)

            fog_added_dataset.append((fog_added_image, bounding_boxes))
        # else:
        #     fog_added_dataset.append((image, bounding_boxes))

    return fog_added_dataset

def create_yaml_for_yolov8(image_dir, label_dir, output_yaml):
    """
    Create a YAML file for YOLOv8 with paths to images and labels.

    :param image_dir: Directory containing images.
    :param label_dir: Directory containing label files.
    :param output_yaml: Path to the output YAML file.
    """
    # Collecting all image paths
    image_files = [os.path.join(image_dir, f) for f in os.listdir(image_dir)
if f.endswith(('.jpg', '.jpeg', '.png'))]

    # Verifying and collecting corresponding label paths
    valid_image_label_pairs = []
    for image_file in image_files:
        label_file = os.path.join(label_dir,
os.path.splitext(os.path.basename(image_file))[0] + '.txt')
        if os.path.exists(label_file):
            valid_image_label_pairs.append({'path': image_file, 'label':
label_file})
        else:
            print(f"Skipping {image_file} as its label file does not
exist.")

    # Writing to YAML file
    with open(output_yaml, 'w') as file:
        yaml.dump(valid_image_label_pairs, file)
```

```
    print(f"YAML file created at {output_yaml}")
```

ДОДАТОК Б

Метод підвищення автономності об’єкту у 3D просторі

ЛІСТИНГ ПРОГРАМНОГО КОДУ ДЛЯ СИМУЛЯЦІЇ З ВХІДНИМИ ДАНИМИ З СИМУЛЯЦІЇ

Аркушів 6

Київ 2023

```
def _generateTargetPath(self, steps):

        steps = steps+500 # Just in case

        min_values = (-8, -8, 1)
        max_values = (8, 8, 8)
        min_distance = 4

        start = self.generate_random_position(min_values, max_values)
        end = self.generate_random_position(min_values, max_values)
        distance = 0
        while distance <= min_distance:
            end = self.generate_random_position(min_values, max_values)
            distance = self.distance_between_points(end, start)

        path = {}

        amplitude = (max_values[1] - min_values[1]) / 3  # Amplitude for the
sine wave
        wavelength = steps  # Wavelength of the sine wave

        for step in range(steps):
            fraction = step / (steps - 1)
            x = start[0] + (end[0] - start[0]) * fraction
            z = start[2] + (end[2] - start[2]) * fraction

            # Introduce curvature using a sine wave for the y-coordinate
            sine_wave = amplitude * math.sin((2 * math.pi * step) /
wavelength)
            y_mid = (start[1] + end[1]) / 2
            y = y_mid + sine_wave

            # Ensure y is within the min and max values
            y = min(max(y, min_values[1]), max_values[1])

            path[step] = (x, y, z)

        self.TARGET_PATH = path
        # print("<<<<<<<<<<<<<< TARGET PATH: start, end, middle",
self.TARGET_PATH[0], self.TARGET_PATH[2000], self.TARGET_PATH[1000])

    def distance_between_points(self, point1, point2):
        """
        Calculate the Euclidean distance between two 3D points.

        Parameters:
        point1 (tuple): The first point (x1, y1, z1).
        point2 (tuple): The second point (x2, y2, z2).

        Returns:
        float: The Euclidean distance between point1 and point2.
```

```
        """
        x1, y1, z1 = point1
        x2, y2, z2 = point2
        distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2 + (z2 - z1) **
2)
        return distance

    def generate_random_position(self, min_values, max_values):
        """
        Generate random position coordinates (x, y, z) within specified
ranges.

        Parameters:
        min_values (tuple): A tuple containing the minimum values (min_x,
min_y, min_z).
        max_values (tuple): A tuple containing the maximum values (max_x,
max_y, max_z).

        Returns:
        tuple: Randomly generated (x, y, z) coordinates.
        """
        min_x, min_y, min_z = min_values
        max_x, max_y, max_z = max_values

        x = random.uniform(min_x, max_x)
        y = random.uniform(min_y, max_y)
        z = random.uniform(min_z, max_z)
        return (x, y, z)

    def _housekeeping(self):
        """Housekeeping function.

        Allocation and zero-ing of the variables and PyBullet's
parameters/objects
        in the `reset()` function.

        """
        if self.SHOW_TARGET and not hasattr(self, 'TARGET_PATH'):
            self._generateTargetPath(self.EPISODE_LENGTH_STEPS)
        #### Initialize/reset counters and zero-valued variables ###
        self.RESET_TIME = time.time()
        self.step_counter = 0
        self.first_render_call = True
        self.X_AX = -1*np.ones(self.NUM_DRONES)
        self.Y_AX = -1*np.ones(self.NUM_DRONES)
        self.Z_AX = -1*np.ones(self.NUM_DRONES)
        self.GUI_INPUT_TEXT = -1*np.ones(self.NUM_DRONES)
        self.USE_GUI_RPM=False
        self.last_input_switch = 0
        self.last_clipped_action = np.zeros((self.NUM_DRONES, 4))
        self.gui_input = np.zeros(4)
```

```
        #### Initialize the drones kinemaatic information ##########
        self.pos = np.zeros((self.NUM_DRONES, 3))
        self.quat = np.zeros((self.NUM_DRONES, 4))
        self.rpy = np.zeros((self.NUM_DRONES, 3))
        self.vel = np.zeros((self.NUM_DRONES, 3))
        self.ang_v = np.zeros((self.NUM_DRONES, 3))
        if self.PHYSICS == Physics.DYN:
            self.rpy_rates = np.zeros((self.NUM_DRONES, 3))
        #### Set PyBullet's parameters #############################
        p.setGravity(0, 0, -self.G, physicsClientId=self.CLIENT)
        p.setRealTimeSimulation(0, physicsClientId=self.CLIENT)
        p.setTimeStep(self.PYB_TIMESTEP, physicsClientId=self.CLIENT)
        p.setAdditionalSearchPath(pybullet_data.getDataPath(),
physicsClientId=self.CLIENT)
        #### Load ground plane, drone and obstacles models #########
        self.PLANE_ID = p.loadURDF("plane.urdf",
physicsClientId=self.CLIENT)

        self.DRONE_IDS =
np.array([p.loadURDF(pkg_resources.resource_filename('gym_pybullet_drones',
'assets/'+self.URDF),
                                              self.INIT_XYZS[i,:],

p.getQuaternionFromEuler(self.INIT_RPYS[i,:]),
                                              flags =
p.URDF_USE_INERTIA_FROM_FILE,
                                              physicsClientId=self.CLIENT
                                              ) for i in
range(self.NUM_DRONES)])
        if self.GUI and self.USER_DEBUG:
            for i in range(self.NUM_DRONES):
                self._showDroneLocalAxes(i)
        if self.OBSTACLES:
            self._addObstacles()
        if self.SHOW_TARGET:
            self._addTarget(starting_point=self.TARGET_PATH[0])



def _observationSpace(self):
        """Returns the observation space of the environment.

        Returns
        -------
        ndarray
            A Box() of shape (NUM_DRONES,H,W,4) or (NUM_DRONES,15) depending
on the observation type.

        """
        if self.OBS_TYPE == ObservationType.RGB:
            return spaces.Box(low=0,
```

```
                                high=255,
                                shape=(self.NUM_DRONES, self.IMG_RES[1],
self.IMG_RES[0], 4), dtype=np.uint8)
        elif self.OBS_TYPE == ObservationType.KIN:
            ############################################################
            #### OBS SPACE OF SIZE 12 (9 from vector + TX, TY, TZ -
toTargetX, toTargetY, toTargetZ)
            #### Observation vector ### X        Y        Z       Q1   Q2
Q3   Q4   R       P       Y       VX       VY       VZ       WX       WY
WZ TX TY TZ
            lo = -np.inf
            hi = np.inf
            obs_lower_bound = np.array(
                [[lo, lo, lo, lo, lo, lo, lo, lo, lo, lo, lo, lo] for i in
range(self.NUM_DRONES)])
            obs_upper_bound = np.array(
                [[hi, hi, hi, hi, hi, hi, hi, hi, hi, hi, hi, hi] for i in
range(self.NUM_DRONES)])
            #### Add action buffer to observation space ################
            act_lo = -1
            act_hi = +1
            for i in range(self.ACTION_BUFFER_SIZE):
                if self.ACT_TYPE in [ActionType.RPM, ActionType.VEL]:
                    obs_lower_bound = np.hstack(
                        [obs_lower_bound, np.array([[act_lo, act_lo, act_lo,
act_lo] for i in range(self.NUM_DRONES)])])
                    obs_upper_bound = np.hstack(
                        [obs_upper_bound, np.array([[act_hi, act_hi, act_hi,
act_hi] for i in range(self.NUM_DRONES)])])
                elif self.ACT_TYPE == ActionType.PID:
                    obs_lower_bound = np.hstack(
                        [obs_lower_bound, np.array([[act_lo, act_lo, act_lo]
for i in range(self.NUM_DRONES)])])
                    obs_upper_bound = np.hstack(
                        [obs_upper_bound, np.array([[act_hi, act_hi, act_hi]
for i in range(self.NUM_DRONES)])])
                elif self.ACT_TYPE in [ActionType.ONE_D_RPM,
ActionType.ONE_D_PID]:
                    obs_lower_bound = np.hstack([obs_lower_bound,
np.array([[act_lo] for i in range(self.NUM_DRONES)])])
                    obs_upper_bound = np.hstack([obs_upper_bound,
np.array([[act_hi] for i in range(self.NUM_DRONES)])])
            result = spaces.Box(low=obs_lower_bound, high=obs_upper_bound,
dtype=np.float32)
            return result
        else:
            print("[ERROR] in BaseRLAviary._observationSpace()")


############################################################################
####
```

```
    def _computeObs(self):
        """Returns the current observation of the environment.

        Returns
        -------
        ndarray
            A Box() of shape (NUM_DRONES,H,W,4) or (NUM_DRONES,12) depending
on the observation type.

        """
        if self.OBS_TYPE == ObservationType.RGB:
            if self.step_counter % self.IMG_CAPTURE_FREQ == 0:
                for i in range(self.NUM_DRONES):
                    self.rgb[i], self.dep[i], self.seg[i] =
self._getDroneImages(i,

segmentation=False

)
                    #### Printing observation to PNG frames example
############
                    if self.RECORD:
                        self._exportImage(img_type=ImageType.RGB,
                                          img_input=self.rgb[i],
                                          path=self.ONBOARD_IMG_PATH +
"drone_" + str(i),
                                          frame_num=int(self.step_counter /
self.IMG_CAPTURE_FREQ)
                                          )
            return np.array([self.rgb[i] for i in
range(self.NUM_DRONES)]).astype('float32')
        elif self.OBS_TYPE == ObservationType.KIN:
            ############################################################
            #### OBS SPACE OF SIZE 12 (12 from vector, incl. TX, TY, TZ -
toTargetX, toTargetY, toTargetZ)
            obs_12 = np.zeros((self.NUM_DRONES, 12))
            for i in range(self.NUM_DRONES):
                # obs =
self._clipAndNormalizeState(self._getDroneStateVector(i))
                obs = self._getDroneStateVector(i)
                # to_target_coordinates =
np.array(self.TARGET_PATH[self.step_counter])
                obs_12[i, :] = np.hstack([obs[7:10], obs[10:13], obs[13:16],
obs[20:23]]).reshape(12, )
            ret = np.array([obs_12[i, :] for i in
range(self.NUM_DRONES)]).astype('float32')
            #### Add action buffer to observation ######################
            for i in range(self.ACTION_BUFFER_SIZE):
                ret = np.hstack([ret, np.array([self.action_buffer[i][j, :]
for j in range(self.NUM_DRONES)])])
```

```
            return ret
            ############################################################
        else:
            print("[ERROR] in BaseRLAviary._computeObs()")
```

ДОДАТОК В

Метод підвищення автономності об’єкту у 3D просторі

ЛІСТИНГ ПРОГРАМНОГО КОДУ ПОВНОЇ СИСТЕМИ

Аркушів 3

Київ 2023

```
from ultralytics import YOLO

class YOLOv8ObjectDetector:
    def __init__(self, model_path='yolov8n.pt'):
        # Load the YOLOv8 model
        self.model = YOLO(model_path)

    def detect_objects(self, image_array):
        image = self.convert_array_to_image(image_array)

        # Ensure the image is a list of images for the model
        results = self.model([image], verbose=False)
        confidence = 0
        selectedImageSpec = None

        for result in results:
            boxes = result.boxes
            if boxes.shape[0] > 0:  # Check if there are any detections
                conf, conf_index = torch.max(boxes.conf, 0)
                xywhn = boxes.xywhn[conf_index]
                if conf > confidence:
                    selectedImageSpec = (xywhn)
                    confidence = conf  # Update the highest confidence

        if selectedImageSpec is not None:
            output = self._calculate_distance_vector(selectedImageSpec)
        else:
            output = (0, 0)  # or some default value

        return output

    @staticmethod
    def _calculate_distance_vector(obj_center_x, obj_center_y, img_width,
img_height):
        img_center_x, img_center_y = img_width / 2, img_height / 2
        distance_x = obj_center_x - img_center_x
        distance_y = obj_center_y - img_center_y
        return (distance_x, distance_y)
```

```
import torch
import numpy as np
from scipy.stats import multivariate_normal

class DepthEstimator:
    def __init__(self, model_type="DPT_Large"):
        # Load the MiDaS model
        self.model = torch.hub.load("intel-isl/MiDaS", model_type)
        self.device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
        self.model.to(self.device)
```

```
        self.model.eval()

        # Load transforms
        self.transform = torch.hub.load("intel-isl/MiDaS",
"transforms").dpt_transform

    def estimate_depth(self, image, bbox, sigma=10, D_min=0.1):
        """
        Estimate the depth of the object inside the bounding box.

        Parameters:
        image (np.array): The input image.
        bbox (tuple): The bounding box (center_x, center_y, width, height).
        sigma (float): The parameter controlling the weight distribution.
        D_min (float): The minimum depth threshold for filtering.

        Returns:
        float: The estimated depth.
        """
        # Preprocess the image
        input_batch = self.transform(image).to(self.device)

        # Get depth prediction
        with torch.no_grad():
            prediction = self.model(input_batch)
            prediction = torch.nn.functional.interpolate(
                prediction.unsqueeze(1),
                size=image.shape[:2],
                mode="bicubic",
                align_corners=False,
            ).squeeze()
        depth_map = prediction.cpu().numpy()

        # Calculate weighted depth
        return self._calculate_weighted_depth(depth_map, bbox, sigma, D_min)

    def _calculate_weighted_depth(self, depth_map, bbox, sigma, D_min):
        x_center, y_center, width, height = bbox
        x, y = np.meshgrid(np.arange(depth_map.shape[1]),
np.arange(depth_map.shape[0]))
        pos = np.dstack((x, y))

        # Gaussian distribution centered at bbox
        rv = multivariate_normal([x_center, y_center], [[sigma**2, 0], [0,
sigma**2]])
        weights = rv.pdf(pos)

        # Apply weights and threshold
        weighted_depth = depth_map * weights
        weighted_depth[depth_map < D_min] = 0
```

```
        total_weight = np.sum(weights[depth_map >= D_min])
        if total_weight == 0:
            return 0
        return np.sum(weighted_depth) / total_weight

from YOLO_engine import YOLOv8ObjectDetector
from MiDaS_engine import DepthEstimator

class DroneController:
    def __init__(self):
        self.yolo_detector = YOLOv8ObjectDetector()
        self.depth_estimator = DepthEstimator()

    def process_image(self, image):
        # YOLO Detection
        detections = self.yolo_detector.detect_objects(image)
        if not detections:
            # Return stable RPM for hovering if no object is detected
            return (0, 0, 0)

        # Assuming we process the first detected object
        highest_confidence = 0
        confident_detection = None
        for detection in detections:
            if detection['confidence'] > highest_confidence:
                highest_confidence = detection['confidence']
                confident_detection = detection

        if confident_detection is not None:
            bbox = confident_detection['bbox']
            toTargetX, toTargetY = confident_detection['distance_vector']

            # MiDaS Depth Estimation
            toTargetZ = self.depth_estimator.estimate_depth(image, bbox)

            return (toTargetX, toTargetY, toTargetZ)
        else:
            return (0, 0, 0)
```