ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНЖЕНЕРІЇ ТА ТРАНСПОРТУ

КАФЕДРА АВТОМАТИЗАЦІЇ, РОБОТОТЕХНІКИ І МЕХАТРОНІКИ

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Розробка системи управління БПЛА з елементами штучного інтелекту»

«Development of a UAV control system with artificial intelligence elements»

Виконав: студент 2 курсу, групи 6А

спеціальність 174 – «Автоматизація, комп’ютерно-інтегровані технології та робототехніка»

Сологубов Олексій Ігорович

Керівник: к.т.н., доцент Єдинович М.Б.

Рецензент__Литвиненко В,І,

Херсон – 2024 рік

Форма № Н-9.01

Херсонський національний технічний університет

Факультет Інженерії та транспорту

Кафедра Автоматизації, робототехніки і мехатроніки

Ступінь вищої освіти магістр

Спеціальність 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

ЗАТВЕРДЖУЮ:
Завідувач кафедри автоматизації, робототехніки і мехатроніки
Сєліверстов І.А.
«___» _____ 2024 р.

З А В Д А Н Н Я
на дипломну роботу студенту

Сологубов Олексію ігоревичу

1. Тема проекту: Розробка системи управління БПЛА з елементами штучного інтелекту	
керівник проекту:	к.т.н., доцент Єдиноаич М.Б.
затверджена наказом вищого навчального закладу від 30.08.2024 р. № 444с	
2. Строк подання студентом проекту «16» грудня2024 р.	
3. Вихідні дані до проекту: технічні характеристики БПЛА.	
4. Зміст розрахунково-пояснювальної записки:	
1. Загальні питання штучного інтелекту	
2. Поняття безпілотного літального апарату	
3. Використання двомірного середовища на базі мови Python	
5. Перелік графічного матеріалу	
1. Принцип роботи навчання з підкріпленням	
2. Уявлення структури нейронної мережі	
3. Каскадна структура PID-контролерів	
4. Результати відстеження експерименту DQN	
5. Порівняння відстеження винагороди SAC алгоритму з DQN	
6. Порівняння відстеження винагороди SAC алгоритму з удосконаленим алгоритмом SAC_2	

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Єдинович М.Б., к.т.н., доцент	07.10.2024	

7. Дата видачі завдання «07» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Підбір і огляд літератури	07.10.2024-22.10.2024	
2	Опис загальних питань ШІ	23.10.2024-29.10.2024	
3	Використання двомірного середовища на базі мови програмування Python	30.10.2024-12.11.2024	
4	Формулювання висновків	13.11.2024-15.1.2024	
5	Оформлення ПЗ і графічного матеріалу	06.12.2024-15.12.2024	

Студент _____ Солгубов О.І.
(підпис)

Керівник проекту _____ Єдинович М.Б.
(підпис)

Номер рядка	Формат	Позначення	Найменування	Кіль-кість	Шифр до-кумента	Примі-тка
1	А4		Завдання на проектування	2		
2	А4	ХНТУ 174.КРМБ.24.ххх РФ	Реферат	1	РФ	
3	А4	ХНТУ 174.КРМБ.24.ххх ПЗ	Пояснювальна записка	110	ПЗ	
4	А4	Демонстраційне креслення	Принцип роботи навчання з підкріпленням	1		
5	А4	Демонстраційне креслення	Уявлення структури нейронної мережі	1		
6	А4	Демонстраційне креслення	Каскадна структура PID-контролерів	1		
7	А4	Демонстраційне креслення	Результати відстеження експерименту DQN	1		
8	А4	Демонстраційне креслення	Порівняння відстеження винагороди SAC алгоритму з DQN	1		
9	А4	Демонстраційне креслення	Порівняння відстеження винагороди SAC алгоритму з удосконаленим алгоритмом	1		

Зм.	Арк.	№ докум	Підпис	Дата	ХНТУ 174.КРМ.24.038 ВР		
Розроб.		Сологубов О.І.			Відомість об'єму роботи	Літ.	Аркуш 4 / Аркушів 83
Перевір.		Єдинович М.Б					
Реценз.						ХНТУ, гр. 2Амб	
Н. Контр.		Поліщук В.М.					
Затверд.		Селіверстов І.А					

РЕФЕРАТ

Кваліфікаційна робота магістра: 110 сторінок, 14 рисунків, 2 таблиці, 4 додатки.

Кваліфікаційна робота магістра присвячена дослідженню і розробці системи управління безпілотними літальними апаратами (БПЛА) з використанням елементів штучного інтелекту (ШІ). Робота містить комплексний аналіз сучасних підходів до створення систем автономного управління БПЛА, інтеграції ШІ та використання імітаційних моделей для тестування алгоритмів.

У дослідженні детально проаналізовано існуючі методи управління БПЛА, включаючи використання традиційних PID-контролерів, ручного управління, а також алгоритмів навчання з підкріпленням (Reinforcement Learning). Особливу увагу приділено адаптивним властивостям систем управління, які забезпечують БПЛА автономність і здатність до прийняття рішень у змінних умовах середовища. Виконано порівняльний аналіз ефективності зазначених підходів у різних сценаріях.

В результаті досліджень було розроблено математичну модель та алгоритми управління, що враховують динаміку руху БПЛА у двовимірному просторі, з урахуванням впливу інерції, гравітації та тяги пропелерів. Для тестування ефективності алгоритмів створено симуляційне середовище на основі мови програмування Python із використанням бібліотек NumPy, Matplotlib, Pygame та Stable-Baselines3. Середовище дозволяє моделювати завдання стабілізації польоту, навігації та уникнення перешкод.

З метою створення універсальної платформи для дослідження систем управління розроблено симуляційне середовище на основі мови Python. Це

середовище дозволяє тестувати алгоритми в умовах, близьких до реальних, без необхідності використання дорогого обладнання.

Ключові слова: БПЛА, ШІ, контролер, агент, винагорода, імітація, симуляція, управління, середовище, тяга, спостереження.

ABSTRACT

Master's thesis: 110 pages, 14 figures, 2 tables, 4 appendices.

The master's qualification work is dedicated to the study and development of a control system for unmanned aerial vehicles (UAVs) using artificial intelligence (AI). The work includes a comprehensive analysis of modern approaches to creating autonomous UAV control systems, integrating AI, and applying simulation models for algorithm testing.

The research provides a detailed analysis of existing UAV control methods, including traditional PID controllers, manual control, and reinforcement learning algorithms. Special attention is given to the adaptive capabilities of control systems, which enable UAVs to operate autonomously and make decisions in changing environmental conditions. A comparative analysis of the efficiency of these approaches in different scenarios has been conducted.

As a result of the research, a mathematical model and control algorithms were developed, considering the dynamics of UAV movement in a two-dimensional space, taking into account the effects of inertia, gravity, and propeller thrust. A simulation environment based on the Python programming language was created to test the efficiency of these algorithms, utilizing libraries such as NumPy, Matplotlib, Pygame, and Stable-Baselines3. This environment allows for modeling tasks related to flight stabilization, navigation, and obstacle avoidance.

To establish a universal platform for exploring control systems, a simulation environment was developed using Python. This environment enables algorithm testing under conditions close to reality, without requiring expensive equipment.

Key words: UAV, AI, controller, agent, reward, imitation, simulation, control, environment, thrust, observation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БПЛА – безпілотно літальний апарат

UAV – Unmanned Aerial Vehicle

ПІД – пропорціонально-інтегрально-диференціальний

PID - Proportional-Integral-Derivative

ШІ – штучний інтелект

AI – Artificial intelligence

2D – двомірний простір

RL – навчання з підкріпленням (Reinforcement Learning)

ВСТУП

Актуальність теми: На сьогоднішній день безпілотні літальні апарати (БПЛА) відіграють важливу роль у багатьох сферах людської діяльності. Вони активно використовуються для моніторингу навколишнього середовища, логістики та транспортування вантажів, наукових досліджень, а також у військових операціях. Застосування дронів дозволяє вирішувати завдання, які раніше були недосяжними або вимагали значних ресурсів, як людських, так і фінансових.

З розвитком технологій до безпілотників висуваються все більш жорсткі вимоги: вони мають бути автономними, маневреними, надійними та точними у виконанні завдань. Одним з основних напрямів, що сприяє цьому розвитку, є інтеграція систем штучного інтелекту. Саме алгоритми машинного навчання та навчання з підкріпленням (RL) відкривають можливість для створення автономних систем управління, здатних самостійно приймати рішення та адаптуватися до умов навколишнього середовища.

Попри високий потенціал таких розробок, їх успішне впровадження потребує експериментальної перевірки. У реальних умовах тестування алгоритмів управління зазвичай відбувається на випробувальних стендах, які дозволяють імітувати польотні ситуації та поступово відлагоджувати всі компоненти системи. Однак створення подібних стендів є досить дорогим і технічно складним процесом. Це особливо актуально для навчальних чи дослідницьких проєктів, які обмежені у ресурсах.

Саме тому симуляційні середовища стають альтернативою для проведення досліджень у галузі управління БПЛА. Вони дозволяють на базовому рівні моделювати польотні ситуації, тестувати алгоритми та аналізувати їх ефективність без залучення реального обладнання. У даній роботі для дослідження використовується 2D-симуляція, яка забезпечує можливість відтворити динаміку руху БПЛА у двох вимірах.

Мета та завдання дослідження:

Основна мета роботи – створення платформи для дослідження ефективності алгоритмів штучного інтелекту на основі методу навчання з підкріпленням (Reinforcement Learning) у завданнях навігації та управління БПЛА.

Для досягнення поставленої мети вирішено такі завдання:

- аналіз ефективності та адаптивності алгоритмів у змінних умовах середовища.

- порівняння результатів ручного управління, PID-контролера та алгоритмів навчання з підкріпленням у задачі досягнення цільових точок без аварій.

- визначення переваг і недоліків кожного з методів з метою їхнього подальшого використання у реальних системах управління БПЛА.

- на основі отриманих теоретичних даних розроблено датчик перекосу утокових ниток придатний для практичного використання у виробничих умовах;

Розроблене середовище стане дослідницьким інструментом для тестування та вдосконалення алгоритмів управління, надаючи можливість проводити безпечні та ефективні експерименти з можливістю масштабування отриманих результатів до реальних умов.

Розробка 2D-симуляційного середовища для дослідження управління безпілотним літальним апаратом (БПЛА) ставить перед собою **наступні завдання:**

По-перше, необхідно створити імітаційну модель БПЛА, яка відтворює його поведінку в реальних фізичних умовах. У процесі моделювання враховуються гравітація, інерція та вплив тяги пропелерів, що забезпечує наближену до реальності симуляцію. У середовищі дрон має можливість

рухатися у двох осях (X та Y), що є достатнім для виконання задач навігації та стабілізації.

По-друге, важливим є моделювання маршруту польоту дрона. Для цього реалізується механізм, за яким маршрутні точки генеруються випадково в різних місцях середовища, причому одна за одною. Встановлення часових обмежень для досягнення кожної точки дозволяє оцінити швидкість реакції системи та ефективність алгоритмів управління.

Третім завданням є впровадження різних підходів до управління дронами у симуляційному середовищі. Зокрема, передбачається можливість тестування ручного управління, PID-регулятора та алгоритмів навчання з підкріпленням. Підтримка різних методів керування дає змогу порівняти їх ефективність у досягненні поставлених цілей у динамічних умовах середовища.

Обґрунтування вибору підходу:

Відсутність доступу до фізичного випробувального стенда компенсується використанням імітаційного моделювання з використанням мови програмування Python. Для побудови моделі управління БПЛА обрано двовимірний простір, оскільки дві координатні осі (X та Y) є достатніми для виконання поставлених завдань навігації та стабілізації.

Рух у 2D-просторі дозволяє проводити аналіз роботи алгоритмів без необхідності врахування додаткових факторів, характерних для 3D-середовищ, таких як орієнтація у просторі чи висота. Крім того, використання 2D-моделі суттєво зменшує обчислювальну складність завдання та дозволяє виконувати симуляції на менш продуктивних апаратних платформах.

Аргументація вибору мови програмування Python:

Python обрано для реалізації симуляційного середовища завдяки його простоті, гнучкості та наявності широкого набору бібліотек. Зокрема, для моделювання та навчання використовуються NumPy, Pygame, Matplotlib і Stable-Baselines3, що забезпечують зручні інструменти для створення фізичних моделей, візуалізації результатів та реалізації алгоритмів машинного навчання.

Двовимірне моделювання в Python є ефективним завдяки відсутності потреби у складних розрахунках, характерних для 3D-середовищ, таких як AirSim, Gazebo чи Unreal Engine. Python дозволяє швидко розробляти симуляції з мінімальними вимогами до системних ресурсів, що робить його доступним для навчальних і дослідницьких проєктів.

Таким чином, Python є оптимальним вибором для реалізації симуляційного середовища через його доступність, гнучкість і ефективність для задач імітаційного моделювання у 2D-просторі.

Наукова новизна роботи полягає у розробці системи управління БПЛА з використанням симуляційних методів для проєктування та тестування алгоритмів з елементами штучного інтелекту.

Практична значущість полягає у тому, що розроблене симуляційне середовище та запропоновані методи можуть бути використані для розробки систем управління БПЛА в умовах обмеженого доступу до матеріально-технічної бази. Отримані результати дослідження можуть бути основою для подальшого вдосконалення алгоритмів і їх впровадження у реальні безпілотні системи.

1 ЗАГАЛЬНІ ПИТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Загальні питання штучного інтелекту

Штучний інтелект (ШІ) — це одна з найперспективніших і найдинамічніших галузей сучасної комп’ютерної науки, яка займається створенням програмних і апаратних систем, здатних виконувати завдання, що зазвичай вимагають участі людського інтелекту. Тобто основною метою ШІ є моделювання когнітивних функцій людини, таких як навчання, розпізнавання мови, прийняття рішень, розв'язання складних проблем тощо.

Штучний Інтелект поєднує в собі кілька підходів, методів і технологій, які дозволяють комп'ютерам і іншим пристроям імітувати інтелектуальну поведінку людини. На відміну від класичних програм, що діють за жорстко заданими інструкціями, системи ШІ працюють гнучко: вони здатні самостійно навчатися на основі досвіду і величезних обсягів даних.

Основні завдання штучного інтелекту включають:

1. Обробку і аналіз даних: штучний інтелект здатний швидко обробляти великі обсяги інформації, які людині було б важко охопити. Наприклад, аналіз великих баз даних у медичних або фінансових системах.
2. Навчання на прикладах (машинне навчання): системи штучного інтелекту можуть самостійно виявляти закономірності у даних і покращувати свою роботу завдяки досвіду. Ця здатність до навчання є однією з найважливіших характеристик штучного інтелекту.
3. Розпізнавання образів і мови: завдяки алгоритмам штучного інтелекту комп’ютери можуть ідентифікувати об’єкти на зображеннях, розпізнавати голосові команди, тексти чи навіть емоції.

4. Автоматичне прийняття рішень: штучний інтелект допомагає приймати оптимальні рішення у складних ситуаціях на основі аналізу даних і побудови прогнозів.

Таким чином, штучний інтелект не просто виконує запрограмовані дії, а "вчиться" і адаптується в процесі роботи, що дозволяє йому працювати в нових умовах і розв’язувати завдання, з якими він раніше не стикався.

1.2. Історія розвитку штучного інтелекту

Історія розвитку штучного інтелекту (ШІ) — це шлях від перших теоретичних ідей до сучасних складних алгоритмів та систем, що здатні виконувати завдання, які ще кілька десятиліть тому здавались фантастикою. Штучний інтелект пройшов довгий і нерівний шлях, де на зміну періодам активного розвитку приходили "зими" — періоди застою та скептицизму.

Поняття "машинного розуму" почало з'являтися ще задовго до офіційного виникнення терміна "штучний інтелект". Вже в стародавні часи філософи та математики намагались пояснити процеси мислення людини та створити моделі, що могли б їх відтворювати.

До 1950-х років серед перших ідей в XVII столітті Блез Паскаль і Готфрід Лейбніц створюють механічні обчислювальні машини, які автоматизують виконання простих математичних операцій.

У 1830-х роках Чарльз Беббідж запропонував ідею "аналітичної машини" — пристрою, що за допомогою програм, записаних на перфокартах, міг би виконувати обчислення. І хоча ця машина так і не була створена за життя Беббіджа, його робота стала основою для подальших винаходів. У середині XIX століття Ада Лавлейс, працюючи з Беббіджем, написала перший алгоритм для обчислювальної машини. Саме вона припустила, що машини можуть обробляти не тільки числа, а й символи, що було революційним на той час.

Справжнім поштовхом для розвитку ШІ стала середина XX століття, коли інженери та науковці почали активно досліджувати можливість створення машин, які могли б "думати".

Алан Тюрінг, британський математик, в 1950 році запропонував знаменитий "тест Тюрінга", що мав визначати, чи може машина імітувати людське мислення настільки добре, щоб людина не могла відрізнити її від іншої людини. Цей тест досі є однією з базових ідей у сфері ШІ.

1956-й рік вважається офіційним "народженням" штучного інтелекту. Американський вчений Джон Маккарті організував Дартмутську конференцію, на якій вперше прозвучав термін "штучний інтелект" (Artificial Intelligence). На конференції вчені висунули сміливу ідею: "Будь-який аспект навчання чи інтелектуальної діяльності можна описати настільки точно, що машину можна навчити виконувати ці завдання".

У 1960-х і 1970-х роках спостерігався бурхливий розвиток ШІ. З'явилися перші алгоритми та програми, які демонстрували можливості машинного мислення.

1966 рік. Створення першого чат-бота ELIZA, що імітував спілкування з психологом. Хоча ELIZA працювала за досить примітивними правилами, вона справила велике враження на публіку.

У 1970-х роках були створені перші експертні системи — програми, які намагалися імітувати процес мислення фахівців у певній галузі.

Проте через обмежені обчислювальні ресурси та відсутність великих обсягів даних, розвиток ШІ не відповідав очікуванням. Це призвело до першої "зими ШІ" — періоду зниження інтересу та фінансування.

У 1980-х роках інтерес до ШІ відновився завдяки розвитку комп'ютерних технологій. Основними досягненнями цього періоду стали:

Машинне навчання. З'явилися алгоритми, які дозволяли машинам навчатися на основі даних, а не працювати за жорстко визначеними правилами.

Нейронні мережі. Вчені почали розробляти моделі, натхненні біологічними нейронами, що дало можливість системам ШІ обробляти складні завдання, такі як розпізнавання образів.

Сучасним етапом є розвиток глибокого навчання (2010-ті – сьогодення). З початком XXI століття штучний інтелект досяг неймовірного прориву завдяки кільком ключовим факторам:

1. Збільшення обчислювальних потужностей. Завдяки сучасним графічним процесорам (GPU) стало можливим тренувати складні нейронні мережі.
2. Великі дані (Big Data). З розвитком інтернету та цифрових технологій стало доступним величезна кількість даних, на яких можна навчати системи ШІ.
3. Глибоке навчання (Deep Learning). Алгоритми глибокого навчання дозволили створювати системи, які можуть розпізнавати мову, обличчя, генерувати тексти та навіть керувати транспортом.

1.3. Технології штучного інтелекту та способи їх реалізації

Штучний інтелект використовує різноманітні підходи та алгоритми, які допомагають йому аналізувати дані, ухвалювати рішення, розпізнавати образи й мову, а також виконувати творчі завдання. Основні технології, на яких базується ШІ, можна виділити, але важливо розуміти їх не як окремі блоки, а як взаємопов’язані інструменти, що разом забезпечують ефективну роботу систем.

1.3.1. Машинне навчання (Machine Learning)

Машинне навчання стало основою сучасного ШІ. Це метод, завдяки якому комп’ютери навчаються виконувати завдання на основі аналізу даних, без потреби в чітко прописаних інструкціях. Наприклад, коли алгоритм навчається

розрізняти зображення кота і собаки, він переглядає тисячі прикладів і виявляє в них схожі закономірності. Розрізняють різні способи навчання: навчання з учителем, коли дані мають мітки (як-от назва об’єкта), навчання без учителя, коли система шукає групи чи шаблони самостійно, і навчання з підкріпленням, де алгоритм вчиться методом спроб і помилок, отримуючи "нагороди" за правильні рішення.

Машинне навчання з підкріпленням (Reinforcement Learning, RL) — це метод, за яким штучний інтелект навчається через взаємодію зі середовищем. Головна ідея полягає у тому, що агент (система, яка навчається) виконує дії у певному середовищі й отримує за це нагороди або штрафи. З часом агент на основі отриманих винагород і помилок вчиться обирати оптимальні дії, які приносять найбільшу користь.

Процес нагадує звичайне навчання через спроби та помилки. Наприклад, у грі агент рухається в різних напрямках, намагаючись знайти вихід або уникнути перешкод. Якщо він діє правильно — отримує позитивну нагороду, якщо ні — негативну. Спочатку його дії хаотичні, але поступово він "вчиться" і формує певну стратегію, яка дозволяє досягати поставленої мети.

Особливість RL полягає у тому, що агент намагається не лише отримати миттєву нагороду, а й навчитися приймати рішення, які принесуть найбільшу сумарну користь у довгостроковій перспективі. Наприклад, у грі він може пожертвувати миттєвим бонусом, щоб уникнути небезпеки й у майбутньому отримати більше.

Ця технологія успішно використовується для навчання роботів, керування автономними транспортами, створення розумних ігрових ботів і навіть у фінансових системах для оптимізації інвестицій. Навчання з підкріпленням дозволяє системам самостійно знаходити рішення, яких не очікували навіть розробники.

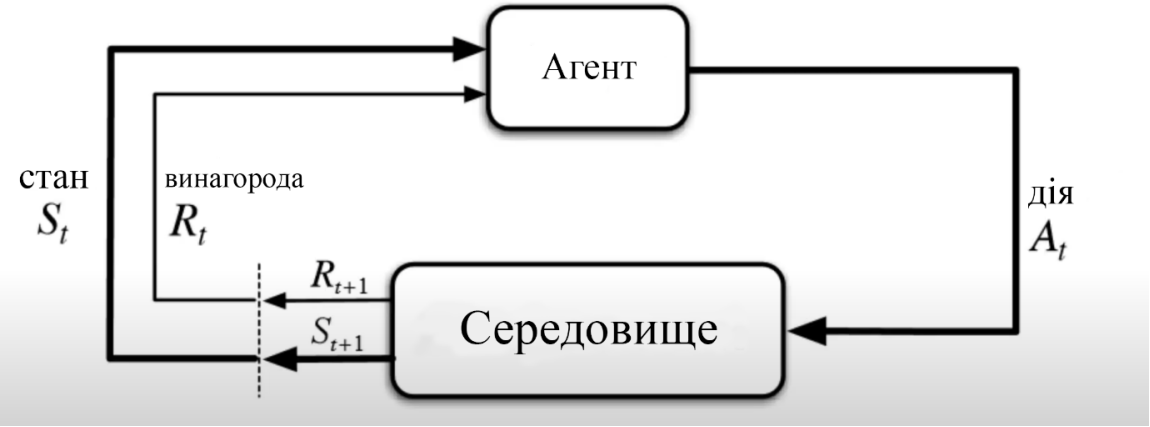


Рисунок 1.1 – Принцип роботи навчання з підкріпленням

1.3.2. Нейронні мережі та глибоке навчання (Deep Learning).

Нейронні мережі та глибоке навчання є ключовими інструментами для обробки складної інформації. Нейронні мережі влаштовані за принципом роботи людського мозку: вони складаються з "нейронів", з’єднаних між собою шарами, і кожен нейрон передає інформацію далі. Найкраще вони справляються з такими завданнями, як обробка зображень, мови та великих обсягів даних. Наприклад, згорткові нейронні мережі використовуються для розпізнавання облич і об’єктів на фото, а рекурентні мережі ефективні для роботи з текстами чи аудіофайлами, де важливий контекст і порядок даних.

Типова глибока нейронна мережа складається з таких шарів:

1. Вхідний шар: отримує початкові дані, такі як зображення, текст або числові значення.

2. Приховані шари: виконують математичні перетворення даних, використовуючи такі функції активації, як сигмоїдна функція чи тангенс.

3. Вихідний шар: формує остаточний результат, наприклад, класифікацію об'єкта чи передбачення числового значення.

У конкретному прикладі, проілюстрованому нижче, комп’ютер повинен бути здатним впоратися з величезним розмаїттям способів представлення даних. Кожна цифра від 0 до 9 може бути записана безліччю способів: Розмір і точна форма кожної цифри, написаної від руки, може значно відрізнятися залежно від того, хто пише і за яких обставин.

Щоб впоратися з мінливістю цих ознак і ще більшою плутаниною взаємодій між ними, стають у нагоді глибоке навчання і глибокі нейронні мережі.

Кожен нейрон нейронної мережі – це математична функція, яка отримує дані через вхід, перетворює їх у більш зручну для сприйняття форму, а потім видає їх через вихід.

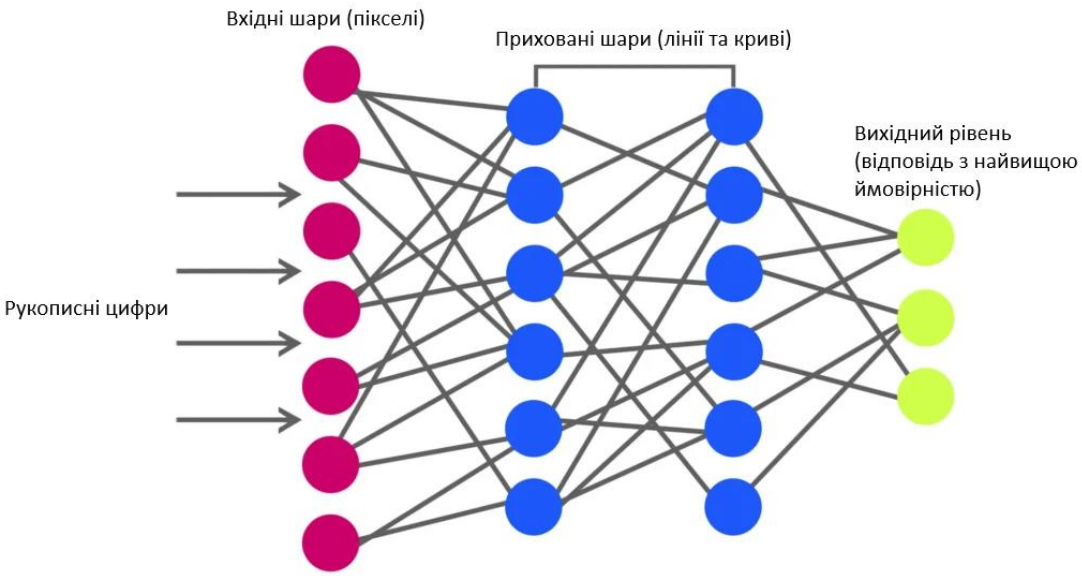


Рисунок 1.2 – Уявлення структури нейронної мережі

1.3.3. Обробка природної мови (NLP)

Обробка природної мови NLP (Natural Language Processing) дозволяє комп’ютерам працювати з людською мовою. Сьогодні ця технологія використовується в чат-ботах, голосових помічниках та автоматичних

перекладачах. Алгоритми NLP розуміють значення слів, аналізують тексти, визначають тональність повідомлень і навіть генерують нові тексти, як це роблять сучасні моделі, на кшталт GPT. Для цього використовуються як правила граматики, так і великі масиви текстових даних, на яких навчається система.

1.3.4. Комп’ютерний зір (Computer Vision)

Комп’ютерний зір відповідає за "зір" машин. Системи комп’ютерного зору дозволяють аналізувати та інтерпретувати візуальну інформацію, що надходить зі зображень, відео або сенсорів. Наприклад, автомобілі з автопілотом використовують комп’ютерний зір для виявлення пішоходів, знаків і перешкод на дорозі. У медицині ця технологія допомагає розпізнавати патології на рентгенівських знімках, а в агрокультурі — оцінювати стан рослин за допомогою дронів.

Окремо слід згадати генерацію контенту. ШІ сьогодні навчився не лише аналізувати дані, а й створювати нові об’єкти — тексти, зображення, музику чи відео. Технології на кшталт генеративно-змагальних мереж (GANs) дозволяють "винаходити" реалістичні картини, яких раніше не існувало, або редагувати відео з неймовірною точністю. Моделі, як-от DALL·E чи MidJourney, генерують зображення за текстовими описами, а GPT допомагає писати тексти, створювати сценарії чи вести діалоги в чатах.

Нарешті, важливим напрямом є робототехніка, де ШІ забезпечує автономну роботу машин і механізмів. Роботи можуть орієнтуватися у просторі, ухвалювати рішення та виконувати складні завдання завдяки поєднанню комп’ютерного зору, машинного навчання та сенсорів. Прикладом є безпілотні дрони, що самостійно облітають перешкоди, чи медичні роботи, які допомагають лікарям під час операцій.

Таким чином, технології штучного інтелекту — це комплекс інструментів, які доповнюють одне одного. Машинне навчання навчає системи, нейронні

мережі обробляють великі дані, NLP дозволяє працювати з мовою, а комп’ютерний зір і генерація контенту додають "зорові" й творчі можливості. Всі ці технології разом формують сучасний ШІ, який уже сьогодні здатен вирішувати завдання, що ще недавно здавались неможливими.

1.4. Переваги та обмеження ШІ

Переваги:

1. Швидкість обробки даних набагато перевищує людські можливості.
2. Виконання рутинних завдань, що дозволяє звільнити час для більш творчої діяльності.
3. Точність і зменшення ймовірності помилок завдяки алгоритмам.

Обмеження:

1. Залежність від наявності якісних даних для навчання. Штучний інтелект, зокрема алгоритми машинного навчання, працює на основі даних. Для того щоб ШІ міг успішно навчатися та виконувати завдання, потрібні великі обсяги якісних даних. Якщо дані є неповними, застарілими або мають похибки, це безпосередньо впливає на точність і надійність роботи алгоритму. У медичній сфері, якщо алгоритм ШІ навчається на неповних або неправильних даних про діагнози пацієнтів, він може робити помилкові прогнози, що ставить під загрозу життя людей. Також існує залежність від наявності "свіжих" даних, оскільки в умовах швидких змін старі дані стають неактуальними.

2. Високі вимоги до обчислювальних ресурсів. Розробка та впровадження систем ШІ потребує значних обчислювальних потужностей, особливо для навчання нейронних мереж і обробки великих обсягів даних. Навіть базовий алгоритм машинного навчання може потребувати тисячі годин обчислень на

потужних процесорах (CPU) або графічних процесорах (GPU). Наприклад, для глибокого навчання (Deep Learning), що лежить в основі багатьох сучасних ШІ-систем, вимагає надзвичайно великої кількості обчислювальних ресурсів. Наприклад, навчання моделі GPT-4o від OpenAI зайняло десятки тисяч GPU-годин і спожило мегавати електроенергії. Для працювання з якісними даними в великому обсязі, для обробки складних алгоритмів потрібне сучасне обладнання, вартість електроенергії та обчислювальних ресурсів є досить високою, що обмежує використання ШІ. Системи ШІ є енергоємними, що ставить питання екологічної ефективності.

3. Проблеми етики та конфіденційності, які виникають при використанні ШІ.

Широке застосування штучного інтелекту порушує важливі питання етики, приватності та захисту даних. Адже системи ШІ часто використовують особисту інформацію користувачів, що може призвести до витоків даних або їх неправильного використання. Серед основниї етичних викликів є дискримінація та упередженість, відповідальність за помилки ШІ (хто має нести відповідальність – розробники, користувачі чи компанія), прозорість алгоритмів, що в комплексі впливає на довіру до систем ШІ серед користувасів і потреба у регулюванні та законодавчому контролі.

1.5. Приклади сучасного використання штучного інтелекту в різних сферах

Сьогодні штучний інтелект активно проникає у найрізноманітніші сфери нашого життя, роблячи його більш зручним, ефективним та інноваційним. В ігровій індустрії, наприклад, ШІ відповідає за створення реалістичних персонажів, які адаптуються до стилю гри гравця. Завдяки цьому гравці отримують справді унікальний досвід. Більше того, технології ШІ допомагають

автоматично створювати величезні ігрові світи та покращують графіку, роблячи візуальне наповнення ігор якіснішим і деталізованішим.

У транспорті ШІ став основою для створення автопілотів, які сьогодні використовуються не тільки в автомобілях, а й у безпілотних літальних апаратах. Відома компанія, така як Tesla, розробляє системи, що здатні самостійно керувати транспортними засобами, аналізуючи дорожню ситуацію в реальному часі. Це не тільки зменшує кількість аварій, але й дозволяє людям використовувати час у дорозі більш продуктивно.

У медицині можливості штучного інтелекту вражають ще більше. Системи ШІ вміють аналізувати медичні знімки, ставити діагнози та навіть прогнозувати розвиток хвороб. Приміром, сучасні алгоритми здатні виявити рак на ранніх стадіях, що значно підвищує шанси на одужання. ШІ також використовується для створення персоналізованих схем лікування, які враховують індивідуальні особливості пацієнта.

В агрокультурі ШІ допомагає фермерам керувати своїми полями ефективніше. Завдяки дронам і сенсорам, які збирають дані про стан ґрунту та рослин, системи можуть аналізувати цю інформацію та підказувати, коли і де варто вносити добрива або поливати. Це дозволяє не тільки підвищити врожайність, але й зменшити витрати ресурсів.

Соціальні мережі — ще один приклад масового використання ШІ. Алгоритми вивчають поведінку користувачів, щоб пропонувати їм цікаві публікації, відео чи рекламу. Завдяки цьому стрічка новин стає індивідуальною для кожного користувача. Також ШІ допомагає боротися зі спамом і модерує контент, фільтруючи неприйнятні матеріали.

Окрім цього, сьогодні особливо популярні чат-боти, які забезпечують підтримку користувачів у різних службах. Наприклад, такі помічники, як ChatGPT або голосові асистенти на кшталт Siri та Google Assistant, можуть вести розмови, допомагати з вирішенням повсякденних завдань або навіть організовувати розклад.

Не можна не згадати про творчі можливості ШІ. Завдяки алгоритмам генерації контенту сьогодні можна створювати зображення, відео та музику. Наприклад, системи на кшталт DALL·E генерують реалістичні або художні зображення за текстовим описом, а інші моделі дозволяють створювати відео чи анімації. Це відкриває величезні можливості для дизайнерів, художників та маркетологів.

Таким чином, штучний інтелект стає невід'ємною частиною сучасного світу. Він не лише автоматизує рутинні завдання, а й допомагає вирішувати складні проблеми, які ще недавно були під силу тільки людині. Від медицини до ігор, від транспорту до агрокультури — ШІ змінює кожну з цих сфер, роблячи наше життя більш технологічним та прогресивним.

2 ПОНЯТТЯ БЕЗПІЛОТНОГО ЛІТАЛЬНОГО АПАРАТУ

Безпілотні літальні апарати (БПЛА) — це керовані або автономні літальні пристрої, які функціонують без присутності пілота на борту. Вони можуть виконувати широкий спектр завдань завдяки своїй маневреності, мобільності та здатності працювати в умовах, де використання традиційної авіації є неможливим або неефективним. БПЛА часто називають дронами, і вони активно використовуються у різних сферах (логістика, научне дослідження, екологія, сільське господарство, будівництво, телекомунікація, оборонна промисловість та ін.)

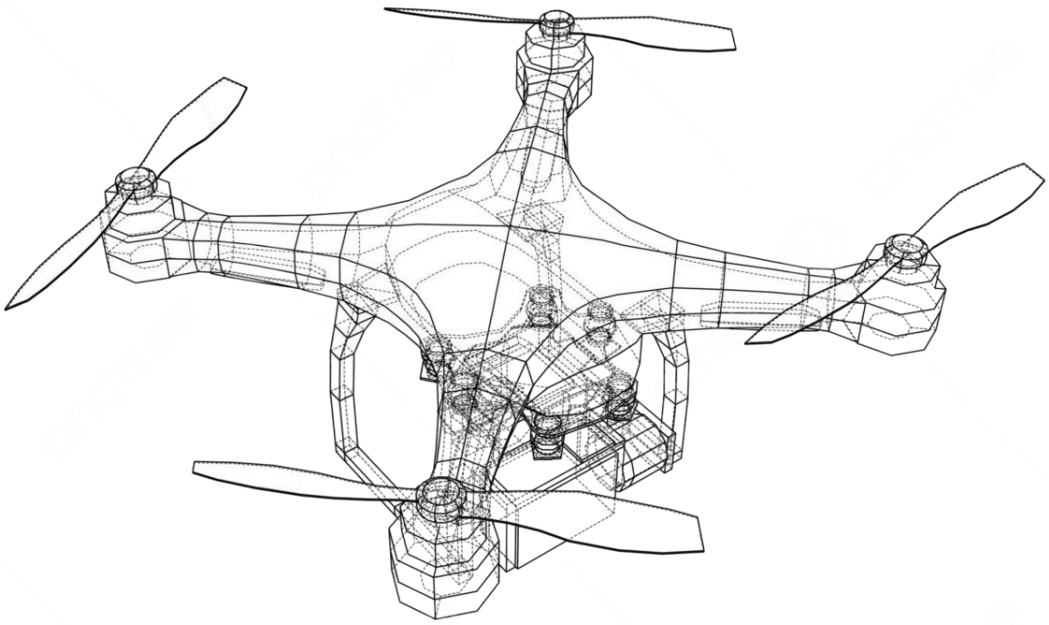


Рисунок 2 – Концепція структури БПЛА

Основні складові БПЛА

Будь-який сучасний БПЛА складається з кількох ключових компонентів:

1. Авіаційна платформа — конструктивна основа дрона (мультикоптер, літак або гібридні варіанти).

2. Система управління — програмне та апаратне забезпечення для контролю польоту та виконання завдань.

3. Датчики — навігаційні (GPS, інерціальні модулі IMU), оптичні (камери, лідари) та інші сенсори, які забезпечують орієнтацію у просторі та збір інформації.

4. Привідна система — двигуни та пропелери, що відповідають за рух і стабілізацію БПЛА.

5. Канал зв'язку — засоби передачі даних між дроном та наземною станцією управління.

6. Бортова обчислювальна система — платформи для обробки даних у реальному часі, часто із застосуванням штучного інтелекту.

2.1 Сучасні підходи до управління БПЛА

Останніми роками безпілотні літальні апарати (БПЛА) стали невід'ємною частиною різних галузей за рахунок підвищення автономності, ефективності та безпеки їхньої експлуатації. Інтеграція передових технологій і методів управління сприяє розширенню сфер застосування безпілотних літальних апаратів та підвищенню їхньої значущості у різних галузях діяльності.

Їхнє широке застосування зумовлене розвитком систем управління, які забезпечують надійну та ефективну роботу. Сучасні підходи до управління БПЛА охоплюють наступні ключові напрями:

1. Автономні системи управління

Сучасні БПЛА оснащуються системами, здатними виконувати польоти без безпосередньої участі оператора, використовуючи алгоритми штучного

інтелекту та машинного навчання. Це дозволяє апаратам самостійно приймати рішення в режимі реального часу, адаптуючись до змінних умов навколишнього середовища. Такі системи знаходять застосування у завданнях розвідки, моніторингу та доставки вантажів.

2. Групове (ройове) управління

Розробка методів координації груп БПЛА дозволяє виконувати складні завдання, що потребують взаємодії кількох апаратів. Групове управління забезпечує синхронізацію дій дронів, розподіл ролей і адаптацію до втрати окремих одиниць, що підвищує надійність та ефективність виконання місій.

3. Інтеграція з мережами зв’язку та управління

Інтеграція БПЛА у загальні мережі зв’язку та управління дозволяє реалізувати ефективне управління і координацію груп апаратів, а також обмін інформацією між ними та іншими учасниками системи. Це відкриває нові можливості у сфері розвідки, моніторингу, зв’язку та транспортування.

4. Використання методів штучного інтелекту

Впровадження алгоритмів штучного інтелекту в системи управління БПЛА сприяє підвищенню їх автономності та ефективності. Методи машинного навчання застосовуються для розпізнавання об'єктів, планування маршрутів та оптимізації енергоспоживання, що значно розширює функціональні можливості апаратів.

5. Забезпечення безпеки та надійності

Сучасні системи управління БПЛА розробляються з урахуванням вимог безпеки та надійності. Впроваджуються механізми виявлення та запобігання зіткненням, захисту від кібератак і забезпечення стійкості до відмов, що є особливо важливим при використанні дронів у густонаселених районах та для критично важливих інфраструктур.

2.1.1. Застосування елементів штучного інтелекту в розробці системи управління БПЛА

Включення методів штучного інтелекту (ШІ) в управління безпілотними літальними апаратами (БПЛА) зумовлене прагненням підвищити їх автономність, ефективність та адаптивність у виконанні складних завдань. ШІ дозволяє БПЛА самостійно приймати рішення, аналізуючи дані з бортових сенсорів і зовнішніх джерел, що мінімізує необхідність постійного контролю з боку оператора.

Основні переваги застосування ШІ в управлінні БПЛА:

Автономність та адаптивність: ШІ забезпечує здатність БПЛА функціонувати у динамічно змінних умовах, самостійно коригуючи свої дії у реальному часі. Це особливо важливо під час виконання місій у складних або непередбачуваних середовищах.

Обробка великих обсягів даних: ШІ дозволяє ефективно аналізувати інформацію, що надходить з різних сенсорів, таких як камери, лідари та радари. Це сприяє більш точному сприйняттю навколишнього середовища та ухваленню обґрунтованих рішень.

Оптимізація маршрутів та енергоспоживання: За допомогою алгоритмів машинного навчання БПЛА здатні планувати оптимальні траєкторії польоту,

враховуючи перешкоди, погодні умови та інші чинники. Це підвищує ефективність виконання завдань та подовжує час автономної роботи.

Розпізнавання та класифікація об’єктів: ШІ дозволяє БПЛА ідентифікувати різні об’єкти на місцевості, що розширює можливості їх застосування у таких галузях, як сільське господарство, моніторинг навколишнього середовища та пошуково-рятувальні операції.

Підвищення безпеки польотів: ШІ сприяє впровадженню систем запобігання зіткненням та інших механізмів для уникнення аварійних ситуацій, що є особливо важливим при експлуатації БПЛА поблизу населених пунктів або у обмеженому повітряному просторі.

Таким чином, використання методів штучного інтелекту в управлінні БПЛА відкриває нові горизонти для їх застосування, підвищуючи надійність, ефективність та безпеку виконання різноманітних завдань.

2.2 Існуючі рішення впровадження елементів ШІ у розробку систем управління БПЛА та їх аналіз

Впровадження штучного інтелекту (ШІ) у системи управління безпілотними літальними апаратами (БПЛА) відкриває можливості для підвищення автономності, ефективності та адаптивності дронів у різноманітних умовах. Існуючі рішення вже включають конкретні технології та підходи, які успішно застосовуються у розробці БПЛА.

1. Автономне керування та планування маршрутів

Одним із найбільш значущих застосувань ШІ є автономне управління БПЛА. Алгоритми машинного навчання дозволяють дронам будувати оптимальні маршрути з урахуванням рельєфу місцевості, перешкод та динамічних умов. Використання методів навчання з підкріпленням

(Reinforcement Learning) дозволяє агентам дронів самостійно навчатися на реальних та симуляційних даних.

Приклад:

• NASA розробила автономні алгоритми для БПЛА, які забезпечують планування польоту та уникнення зіткнень у повітряному просторі.

• DJI Matrice – комерційні дрони з функцією автоматичного побудови маршруту та повернення до точки запуску.

Такі рішення підвищують ефективність управління дронами та мінімізують необхідність у операторі, однак їх реалізація потребує великих обчислювальних потужностей та якісних сенсорів для збору даних про навколишнє середовище.

2. Комп'ютерний зір для розпізнавання об'єктів та навігації

Системи комп'ютерного зору на основі YOLO (You Only Look Once) та CNN (Convolutional Neural Networks) дозволяють БПЛА розпізнавати та класифікувати об'єкти в реальному часі. Це особливо важливо для задач моніторингу, пошуково-рятувальних операцій та автономної навігації.

Приклад:

• Дрони з YOLO використовують детекцію об'єктів для розпізнавання людей або автомобілів на великих територіях.

• Parrot Anafi AI – дрон, оснащений камерами та ШІ для створення карт місцевості та аналізу даних.

Аналіз:

Комп'ютерний зір значно розширює функціональність дронів, але вимагає якісних камер, стабільної роботи алгоритмів у складних умовах освітлення та великих обчислювальних ресурсів для обробки даних у реальному часі.

3. Групове управління та рій дронів

ШІ використовується для координації кількох БПЛА, які працюють у складі рою. Алгоритми на основі згорткових мереж та оптимізації маршрутів дозволяють розподіляти задачі між дронами, щоб вони діяли як єдина система.

Приклад:

• Swarm Intelligence від DARPA – програма розробки скоординованого управління групами БПЛА для виконання військових завдань.

• Громадянські проєкти, де рій дронів застосовується для аерофотозйомки та моніторингу полів.

Аналіз:

Такі системи мають величезний потенціал для виконання складних і масштабних задач. Однак вони вимагають високої точності синхронізації та безперебійного обміну даними між дронами.

4. Системи адаптивного управління

Завдяки ШІ дрони можуть адаптуватися до змінних умов навколишнього середовища. Це досягається шляхом аналізу даних у реальному часі та коригування алгоритмів управління польотом.

Приклад:

• Дрони з функцією розпізнавання перешкод на основі LiDAR та комп'ютерного зору, які змінюють траєкторію при виявленні нових об'єктів.

• Auterion Skynode – платформа з ШІ, що адаптує поведінку дронів при змінах місії або зовнішніх умов.

Аналіз:

Адаптивне управління дозволяє БПЛА працювати у складних та непередбачуваних умовах, але потребує інтеграції різноманітних датчиків та високої швидкості обробки даних.

В загальному аналізі серед переваг існуючих рішень застосування елементів штучного інтелекту в безпілотних літальних апаратах є:

1. Підвищення автономності та ефективності БПЛА.
2. Широкий спектр застосування: від моніторингу полів до пошуково-рятувальних операцій.
3. Можливість ефективного виконання завдань без участі людини.

Серед недоліків та викликів застосувань подібних рішень є:

1. Високі вимоги до обчислювальних ресурсів та сенсорів.
2. Обмеження роботи в умовах низької видимості або нестабільного зв'язку.
3. Проблеми з енергоефективністю, оскільки алгоритми ШІ споживають багато енергії.

Існуючі рішення впровадження елементів ШІ у системи управління БПЛА вже демонструють високу ефективність та великий потенціал. Проте для подальшого розвитку необхідно вирішувати питання оптимізації алгоритмів, підвищення енергоефективності та створення надійних сенсорних систем для роботи у складних умовах.

2.3. Основні вимоги до систем управління БПЛА

Надійність та відмовостійкість

Системи управління БПЛА повинні гарантувати стабільну роботу навіть у випадку відмови окремих компонентів. Це досягається завдяки використанню резервування ключових елементів, таких як канали зв’язку, джерела живлення та процесорні модулі. Відмовостійкість також забезпечується розробкою алгоритмів, які можуть коректно обробляти помилки й автоматично реагувати на них, щоб запобігти втраті контролю над апаратом. Наприклад, при відмові

одного з датчиків система повинна використовувати інформацію з інших датчиків для продовження виконання завдань.

Безпека

Безпека є критично важливим аспектом систем управління БПЛА, оскільки вони працюють в умовах, де навіть незначні збої можуть призвести до серйозних наслідків. Для забезпечення безпеки використовуються такі механізми:

1. Захист даних: шифрування каналів зв’язку, щоб запобігти несанкціонованому доступу.
2. Системи запобігання зіткнень: інтеграція сенсорів (лідарів, камер) для визначення перешкод і планування безпечних маршрутів.
3. Екстрене управління: можливість автоматичного повернення на базу у разі втрати зв’язку або критичних несправностей.

Точність та швидкість реагування

Системи управління повинні забезпечувати високий рівень точності та миттєву реакцію на змінні умови. Це особливо важливо при виконанні завдань у динамічних умовах, таких як обхід перешкод або точне приземлення. Досягнення таких характеристик забезпечується завдяки:

1. Використанню високоточного обладнання (GPS, інерційні системи).
2. Швидкодійним алгоритмам обробки даних.
3. Оптимізації програмного забезпечення для реального часу.

Автономність та адаптивність

Сучасні БПЛА мають бути здатними виконувати місії без постійного втручання оператора, адаптуючись до змін середовища. Автономність забезпечується алгоритмами машинного навчання, які дозволяють апарату самостійно приймати рішення, наприклад, при зміні погодних умов або появі нових перешкод. Адаптивність включає можливість перепланування маршрутів, зміни висоти польоту або повернення на базу у разі небезпеки.

Енергоефективність

Енергоємність системи управління впливає на тривалість польоту і ефективність виконання місії. Використання легких матеріалів, оптимізація алгоритмів управління, застосування енергоефективних компонентів, таких як сучасні літій-полімерні батареї, дозволяє збільшити час автономної роботи апарата. Крім того, енергоефективні рішення забезпечують зменшення витрат на експлуатацію.

Відповідність стандартам

Системи управління повинні відповідати міжнародним і національним стандартам, таким як STANAG 4586 (для інтерфейсів управління) та ICAO (для правил безпеки польотів). Це гарантує їх сумісність з іншими системами, а також легальність використання в різних країнах. Дотримання стандартів також забезпечує інтеграцію БПЛА в єдиний повітряний простір.

Зручність експлуатації

Інтерфейс системи управління повинен бути інтуїтивно зрозумілим, щоб оператори могли швидко освоїти його використання. Простота у навчанні та технічному обслуговуванні дозволяє скоротити витрати часу і ресурсів. Окрім

цього, модульна архітектура системи спрощує модернізацію і впровадження нових функцій без значних витрат.

Загалом, системи управління БПЛА мають відповідати всім зазначеним вимогам, що забезпечує їхню ефективність, безпеку та надійність у будь-яких умовах експлуатації.

2.4. Імітаційне моделювання для тестування систем управління

Розробка систем управління безпілотними літальними апаратами (БПЛА) вимагає високої надійності, безпеки та ефективності їх роботи. Для досягнення цих цілей я вирішив використати методи імітаційного моделювання, оскільки вони дозволяють проводити тестування та оптимізацію у віртуальному середовищі, що робить процес розробки більш керованим і безпечним.

2.4.1. Огляд існуючих симуляційних середовищ для дослідження БПЛА

1. Gazebo

Gazebo є одним із найпотужніших інструментів для моделювання безпілотних літальних апаратів та інших робототехнічних систем. Це програмне середовище відоме своєю реалістичною фізикою та можливістю створення складних сценаріїв для тестування алгоритмів управління. Завдяки підтримці фізичних моделей, Gazebo дозволяє моделювати аеродинаміку, силу тертя, гравітацію та інші фактори, які впливають на поведінку дрона. Особливо важливим є те, що Gazebo легко інтегрується з Robot Operating System (ROS), що робить його універсальним інструментом для розробників. Окрім цього, середовище має велику бібліотеку готових моделей і сцен, які можуть бути використані для досліджень.

Однак, не зважаючи на свої переваги, Gazebo має і певні обмеження. Найбільшим недоліком є високі вимоги до апаратних ресурсів, особливо при моделюванні великих та деталізованих сценаріїв. Крім того, середовище може бути складним у налаштуванні для новачків, оскільки потребує поглиблених знань для ефективного використання.

2. Simulink

Simulink — це потужне середовище для моделювання систем управління безпілотними літальними апаратами (БПЛА), що є частиною пакету MATLAB. Воно дозволяє створювати динамічні моделі польоту з урахуванням фізичних сил, аеродинаміки та зовнішніх умов за допомогою візуального інтерфейсу блок-схем. Simulink дає можливість тестувати PID-регулятори, алгоритми навчання з підкріпленням та інші методи управління, а також моделювати роботу сенсорів, таких як гіроскопи та GPS. Завдяки інтеграції з Simscape можна створювати фізичні моделі компонентів дрона.

Важливою перевагою є можливість генерації коду для вбудованих систем, що дозволяє перенести алгоритми з симуляції на реальні платформи, наприклад Arduino або Raspberry Pi. Використання Simulink вимагає значних обчислювальних ресурсів.

3. AirSim

AirSim, розроблений компанією Microsoft, є однією з найсучасніших платформ для симуляції дронів та автономних транспортних засобів. Відмінною рисою AirSim є те, що він працює на базі Unreal Engine, відомого своєю високоякісною графікою та можливістю створення візуально реалістичних середовищ. Це робить AirSim особливо корисним для завдань, де важлива

деталізована візуалізація, наприклад для тестування алгоритмів, що працюють із комп'ютерним зором чи сенсорними системами. Платформа підтримує моделювання різних типів сенсорів, зокрема камер, лідарів та інерціальних датчиків (IMU), що дозволяє проводити складні дослідження на основі реалістичних даних.

AirSim має відкритий вихідний код і активно підтримується спільнотою, що робить його доступним для широкого кола розробників. Проте слід зазначити, що робота на базі Unreal Engine потребує значних обчислювальних ресурсів, що може стати проблемою при відсутності потужного обладнання. Крім того, новачкам може бути складно налаштувати платформу для простих експериментів через її орієнтованість на візуально інтенсивні сценарії.

4. PyBullet

PyBullet є легковаговим симулятором фізичних процесів, який набув популярності завдяки своїй швидкодії та простоті інтеграції з Python. Це середовище дозволяє ефективно моделювати динаміку руху у реальному часі, що робить його ідеальним для тестування алгоритмів навчання з підкріпленням. PyBullet широко використовується для навчання агентів у простих середовищах, зокрема у 2D-просторі, де важлива швидкість обчислень, а не складність візуалізації.

На відміну від Gazebo чи AirSim, PyBullet менш вимогливий до ресурсів і дозволяє швидко створювати та тестувати моделі. Водночас його головний недолік полягає у менш реалістичному рівні візуалізації, що обмежує його застосування для завдань, де потрібна деталізована графіка чи сенсорні дані.

5. Webots

Webots є ще одним популярним середовищем для моделювання робототехнічних систем, включаючи безпілотні літальні апарати. Це програмне забезпечення дозволяє створювати реалістичні сцени з підтримкою фізики, а також моделювати роботу сенсорів, таких як камери та лідарами. Середовище надає зручний інтерфейс для створення сценаріїв і підтримує можливість програмування на популярних мовах, зокрема Python, C/C++ та Java.

Webots є гарним інструментом для досліджень у навчальних цілях, оскільки його інтерфейс порівняно простий для освоєння. Проте воно менш популярне у професійній спільноті порівняно з такими платформами, як Gazebo чи AirSim, і має певні обмеження у функціональності для складних задач.

6. Unity ML-Agents

Unity ML-Agents — це платформа, що базується на ігровому рушії Unity, і призначена для навчання агентів за допомогою методів машинного навчання, зокрема навчання з підкріпленням. Завдяки можливостям Unity, ML-Agents дозволяє створювати як прості, так і високодеталізовані 3D-середовища для тестування алгоритмів управління дронами. Unity підтримує навчання у реальному часі та надає інструменти для інтеграції з нейронними мережами через Python.

Перевагою цієї платформи є її гнучкість та можливість створювати складні симуляції з високою візуальною якістю. Однак, як і у випадку з AirSim, робота у Unity вимагає значних ресурсів, що може стати обмеженням для користувачів без потужного обладнання.

2.4.2. Основні методи імітаційного моделювання

У практиці розробки систем управління використовується кілька підходів до імітаційного моделювання:

1. Системна динаміка

Цей метод базується на використанні диференційних рівнянь для моделювання потоків та накопичень у системі. Системна динаміка дозволяє аналізувати поведінку системи у довгостроковій перспективі та оцінювати вплив різних факторів. Такий підхід є корисним для вивчення глобальних процесів у складних системах.

2. Дискретно-подієве моделювання

У цьому підході акцент робиться на моделюванні подій, які спричиняють зміни стану системи у конкретні моменти часу. Дискретно-подієве моделювання є особливо актуальним для систем, де важливою є послідовність і часові характеристики подій.

3. Агентне моделювання

Цей метод передбачає поділ системи на безліч автономних агентів, кожен з яких має власну поведінку, цілі та стратегії взаємодії. Агентне моделювання чудово підходить для дослідження складних динамічних систем, де необхідний аналіз взаємодії великої кількості елементів.

2.4.3. Вибір агентного імітаційного моделювання

Агентне імітаційне моделювання є оптимальним методом для дослідження та розробки систем управління безпілотними літальними апаратами (БПЛА), оскільки дозволяє ефективно аналізувати взаємодію автономних об'єктів у складному середовищі.

Однією з головних причин вибору цього підходу є його здатність моделювати автономну поведінку агентів (БПЛА і об’єктів), які можуть мати власні стратегії, цілі та правила взаємодії. У рамках нашої роботи цей метод дозволяє детально дослідити реакцію дрона на різні умови, такі як наявність перешкод чи динамічні зміни у середовищі. Кожен елемент системи моделюється як окремий агент, що діє незалежно та впливає на загальну поведінку всієї системи.

Гнучкість агентного підходу дозволяє налаштовувати різні сценарії для тестування алгоритмів управління: від оптимізації траєкторії руху до ухиляння від перешкод і адаптації до непередбачуваних ситуацій. Це особливо важливо для систем, де взаємодія багатьох елементів потребує ретельного аналізу, як у нашому дослідженні.

Також агентне моделювання є доступним та зручним для реалізації у рамках досліджень, оскільки воно поєднує простоту створення середовища з можливістю проведення численних ітерацій для вдосконалення алгоритмів. Завдяки цьому підходу можна швидко отримати необхідні дані для аналізу, а результати легко інтерпретувати у контексті тестування систем управління.

Таким чином, агентне імітаційне моделювання було обране для нашої дипломної роботи через його здатність глибоко аналізувати поведінку системи управління БПЛА, а також через можливість адаптації під конкретні умови дослідження, що є важливим для досягнення поставлених цілей.

2.5. Застосування БПЛА в різних сферах

Сільське господарство

У сільському господарстві дрони використовуються для моніторингу стану посівів, оцінки якості ґрунту та точного розпилення добрив і засобів захисту

рослин. Завдяки цьому фермери отримують можливість швидко реагувати на зміни у стані своїх угідь, знижувати витрати та підвищувати врожайність.

Моніторинг навколишнього середовища

БПЛА активно застосовуються для оцінки стану довкілля, виявлення забруднень у воді, повітрі та ґрунті. Вони також використовуються для спостереження за популяціями диких тварин, моніторингу лісових пожеж і боротьби з браконьєрством, забезпечуючи цінні дані для природоохоронних організацій.

Інфраструктура та будівництво

Дрони спрощують інспекцію будівель і мостів, дозволяючи виявляти дефекти без необхідності залучення складного обладнання. Вони також використовуються для створення карт місцевості, 3D-моделей ділянок, а також для моніторингу прогресу будівництва, що знижує витрати та оптимізує ресурси.

Логістика

Безпілотники змінюють підхід до доставки, забезпечуючи швидке транспортування товарів у важкодоступні райони. Їх використовують для доставки ліків, продуктів харчування та інших необхідних вантажів, особливо у зони катастроф або в гірські місцевості, що значно підвищує ефективність логістики.

Рятувальні операції та медицина

Дрони дозволяють швидко знаходити постраждалих у зонах катастроф завдяки тепловізорам і камерам. Вони також доставляють ліки, кров і інші медичні засоби до віддалених районів, де традиційні методи транспортування можуть бути недоступними або занадто повільними.

Військове застосування

У військовій сфері БПЛА виконують завдання розвідки, забезпечуючи ситуаційну обізнаність і передачу даних у реальному часі. Вони також використовуються для нанесення точкових ударів, забезпечення радіоелектронної боротьби, транспортування спорядження та координації дій у складі групових операцій.

Охорона та безпека

Дрони забезпечують контроль масових заходів, патрулюють стратегічно важливі об’єкти та сприяють запобіганню злочинам. Вони ефективно моніторять великі території, відстежуючи будь-які підозрілі активності та підвищуючи рівень громадської безпеки.

Наука і дослідження

У наукових проектах дрони використовуються для кліматичних досліджень, моніторингу змін у навколишньому середовищі, а також для археологічних розкопок. Вони дозволяють створювати тривимірні моделі об’єктів і досліджувати верхні шари атмосфери, значно спрощуючи процес збору даних.

Розваги та медіа

Дрони відкрили нові можливості у сфері зйомок, дозволяючи створювати унікальні кадри з висоти пташиного польоту. Вони використовуються для відеозйомок спортивних подій, кінофільмів і рекламних кампаній. Окрім цього, гонки дронів стали популярним видом спорту, що приваблює багато ентузіастів.

2.5.1. Приклади готових БПЛА та їх характеристики

1. Bayraktar TB2 (Туреччина)

Рисунок 2.5.1 – Bayraktar TB2

Тип: Ударно-розвідувальний БПЛА середньої висоти з тривалим часом польоту.

Характеристики:

- Розмах крила: 12 метрів.
- Максимальна швидкість: 222 км/год.
- Тривалість польоту: до 27 годин.
- Вантажопідйомність: до 150 кг.

- Оснащення: електрооптичні сенсори, лазерні цілевказівники, можливість нести керовані бомби MAM-L.

Призначення: Використовується для розвідки, спостереження та точкових ударів. Застосовується у військових конфліктах у різних країнах, включаючи Україну.

2. MQ-9 Reaper (США)

Рисунок 2.5.2 – MQ-9 Reaper

Тип: Ударно-розвідувальний БПЛА великої дальності.

Характеристики:

- Розмах крила: 20 метрів.
- Максимальна швидкість: 482 км/год.
- Тривалість польоту: до 27 годин.

- Вантажопідйомність: до 1700 кг.
- Оснащення: ракети Hellfire, керовані бомби GBU-12 Paveway II, системи радіоелектронної боротьби.

Призначення: Виконує завдання стратегічної розвідки, забезпечення вогневої підтримки та ударів по важливих цілях.

3. Heron TP (Ізраїль)

Рисунок 2.5.3 – Heron TP

Тип: Розвідувально-ударний БПЛА великої дальності.

Характеристики:

- Розмах крила: 26 метрів.
- Максимальна швидкість: 207 км/год.
- Тривалість польоту: до 36 годин.
- Вантажопідйомність: до 450 кг.
- Оснащення: високоточні сенсори, камери з великою роздільною здатністю, ракети Spike.

Призначення: Використовується для стратегічної розвідки, моніторингу та забезпечення ударних операцій.

4. RQ-4 Global Hawk (США)

Рисунок 2.5.4 – RQ-4 Global Hawk

Тип: Стратегічний розвідувальний БПЛА.

Характеристики:

- Розмах крила: 39,9 метра.
- Максимальна швидкість: 575 км/год.

- Тривалість польоту: до 34 годин.
- Дальність польоту: до 22 000 км.
- Оснащення: системи радіолокаційного моніторингу, оптико-електронні сенсори, супутниковий зв’язок.

Призначення: Виконує стратегічну розвідку для військових операцій та моніторинг територій.

5. DJI Matrice 300 RTK (Китай)

Рисунок 2.5.5 – DJI Matrice 300 RTK

Тип: Цивільний дрон для промислового використання.

Характеристики:

- Максимальна швидкість: 83 км/год.
- Тривалість польоту: до 55 хвилин.
- Вантажопідйомність: до 2,7 кг.
- Оснащення: тепловізори, лазерні далекоміри, модулі фотограмметрії.

Призначення: Використовується у будівництві, геодезії, сільському господарстві та рятувальних операціях.

6. Switchblade (США)

Рисунок 2.5.6 – Switchblade

Тип: Камікадзе-дрон (лоїтуючий боєприпас).

Характеристики:

- Вага: до 3,5 кг.
- Максимальна швидкість: 160 км/год.
- Дальність польоту: до 10 км.
- Тривалість польоту: до 15 хвилин.
- Оснащення: високоточна вибухівка для знищення конкретних цілей.

Призначення: Застосовується для знищення техніки чи живої сили противника з мінімальним ризиком для операторів.

Ці приклади демонструють широкий спектр можливостей БПЛА, які можуть виконувати завдання від цивільного моніторингу до стратегічних військових операцій. Кожен із дронів має специфічні характеристики, що роблять його ефективним у певних умовах.

2.5.2. Особливості застосування БПЛА в бойових умовах

Розвідка та спостереження

БПЛА відіграють ключову роль у зборі інформації про позиції супротивника, його техніку та інфраструктуру. Завдяки камерам високої роздільної здатності, інфрачервоним сенсорам і тепловізорам дрони можуть виявляти цілі навіть у складних умовах, наприклад, вночі або в густих лісах. Вони забезпечують оперативне передавання зібраних даних у реальному часі, що дозволяє командуванню приймати ефективні тактичні рішення. Дрони можуть патрулювати великі території без участі людини, мінімізуючи ризик для військового персоналу.

Ударні операції

Ударні БПЛА використовуються для нанесення точкових ударів по цілях з мінімальними супутніми втратами. Вони оснащені керованими боєприпасами, здатними точно вражати об'єкти навіть у складних умовах, таких як міська забудова. Завдяки високій маневреності та автономності, дрони можуть

залишатися непомітними для засобів ППО противника. Ударні операції з використанням БПЛА знижують ризик втрат серед екіпажу традиційних літальних апаратів, оскільки керування здійснюється дистанційно.

3. Радіоелектронна боротьба

Дрони можуть бути обладнані системами радіоелектронної боротьби для придушення зв'язку противника, створення перешкод для його радарів або навігаційних систем. Вони здатні перехоплювати радіосигнали для збору розвідувальної інформації або навіть виводити з ладу електронне обладнання супротивника. Така тактика дозволяє послабити бойові можливості ворога без прямого контакту.

Ройові операції

Сучасні технології дозволяють використовувати рої БПЛА, де кілька дронів діють скоординовано для досягнення спільної мети. Такий підхід має кілька переваг:

1. Дрони можуть виконувати різні ролі, наприклад, одні здійснюють розвідку, інші забезпечують вогневу підтримку.
2. Навіть у разі втрати частини дронів місія продовжується.
3. Завдяки обміну даними між дронами рій здатен адаптуватися до змінної ситуації на полі бою, наприклад, обійти засоби ППО противника.

Транспортування та логістика

БПЛА ефективно використовуються для доставки боєприпасів, продуктів харчування, медичних засобів і обладнання до передових позицій. Вони можуть транспортувати вантажі в складних умовах, таких як гірські райони чи зони

активних бойових дій. Завдяки автономності дрони виконують логістичні завдання швидше, ніж традиційні засоби транспорту.

Адаптація до змінних умов бою

БПЛА здатні швидко змінювати свою місію в залежності від ситуації на полі бою. Наприклад, вони можуть перейти від розвідки до підтримки наземних військ або евакуації поранених. Завдяки високій маневреності дрони можуть ефективно працювати в умовах обмеженого простору, наприклад, у міських боях.

Безпілотні літальні апарати є надзвичайно ефективним інструментом у сучасних військових операціях, здатним значно підвищити результативність бойових дій. Їх ключовою перевагою є здатність швидко адаптуватися до змінних умов, виконувати широкий спектр завдань — від розвідки та ударних операцій до підтримки наземних військ. Крім того, використання БПЛА мінімізує ризики для військового персоналу, дозволяючи зосередитись на стратегічних цілях. Завдяки своїй автономності, маневреності та технологічному оснащенню дрони стали незамінним елементом у сучасних конфліктах.

Недоліки та виклики використання

Попри широкий спектр можливостей, застосування БПЛА у бойових умовах має свої обмеження:

1. Вразливість до радіоелектронної боротьби: Противник може використовувати засоби для блокування або перехоплення управління дронами, що значно ускладнює виконання місій.

2. Залежність від джерел енергії: Обмежений заряд акумуляторів або запас пального впливає на тривалість польоту і виконання завдань, особливо у довготривалих операціях.

3. Високі витрати: Розробка, виробництво та обслуговування сучасних моделей БПЛА потребують значних фінансових ресурсів, що ускладнює їхнє масове впровадження.

4. Етичні питання: Використання ударних дронів у населених районах викликає суперечки через можливість завдання шкоди мирному населенню та інфраструктурі.

Загальний підсумок

Для ефективного використання БПЛА необхідно постійно вдосконалювати їхні технічні характеристики, розробляти нові засоби захисту від електронних атак та оптимізувати витрати. Незважаючи на виклики, дрони залишаються потужним засобом, який змінює правила ведення бойових дій у XXI столітті.

3 ВИКОРИСТАННЯ ДВОМІРНОГО СЕРЕДОВИЩА НА БАЗІ МОВИ ПРОГРАМУВАННЯ PYTHON

У межах роботи для дослідження та тестування ефективності алгоритмів управління БПЛА буде використано симуляційне 2D-середовище, розроблене на базі мови програмування Python. Обраний підхід дозволяє змоделювати рух БПЛА в двовимірному просторі, що є достатнім для імітації роботи апарата в осях X та Y. Це середовище забезпечує всі необхідні умови для перевірки ефективності алгоритмів штучного інтелекту, таких як навчання з підкріпленням (Reinforcement Learning), а також для порівняння з іншими підходами, такими як PID-контролери та ручне управління.

Для дослідження алгоритмів управління безпілотними літальними апаратами у межах цієї роботи було обрано симуляційне середовище, оскільки воно забезпечує низку важливих переваг у порівнянні з реальними випробуваннями. По-перше, симуляція значно скорочує фінансові витрати, які неминуче виникають під час експлуатації реальних дронів, включаючи їх технічне обслуговування та ремонт. По-друге, використання віртуального середовища дозволяє оптимізувати час, необхідний для підготовки та проведення експериментів, адже змінити налаштування симуляції значно простіше, ніж налаштовувати фізичне обладнання. Крім того, симуляція забезпечує повну безпеку, виключаючи ризик пошкодження дрона або небезпеку для оточуючих, що є важливим фактором у дослідницькій роботі.

Серед інших переваг симуляції слід відзначити її незалежність від зовнішніх обмежень. Законодавчі вимоги України регулюють використання дронів у реальних умовах, зокрема, щодо польотів у населених пунктах чи поблизу аеропортів, тоді як симуляція таких обмежень не має. До того ж, погодні умови, які можуть впливати на реальні експерименти, у симуляційному середовищі можна ігнорувати або моделювати за допомогою додаткових регулюючих параметрів. Важливо також, що симуляція усуває необхідність у

великих відкритих просторах для проведення випробувань, адже у віртуальному середовищі немає таких просторових обмежень.

Для створення моделі управління безпілотником у симуляції був обраний двовимірний простір. Такий підхід обумовлений достатністю двох координатних осей (X та Y) для вирішення поставлених завдань. У межах цього дослідження рух у 2D-просторі дозволяє оцінити ефективність алгоритмів стабілізації та навігації без потреби врахування додаткових факторів, характерних для тривимірного моделювання, таких як висота чи орієнтація в просторі. Застосування 2D-моделі також суттєво спрощує реалізацію симуляційного середовища і знижує обчислювальну складність завдання, що дозволяє проводити експерименти швидше та на менш потужних апаратних платформах.

Аргументація вибору мови програмування Python для створення симуляційного простору:

Python є одним із найпопулярніших і найпоширеніших мов програмування завдяки своїй простоті, гнучкості та широкому набору бібліотек, які полегшують розробку програмного забезпечення в різних галузях, включаючи робототехніку та симуляції. Його вибір для даного проекту обґрунтований наступними перевагами:

1. Великий набір бібліотек і фреймворків

Python має широкий вибір бібліотек, таких як NumPy, Pygame, Matplotlib та Stable-Baselines3, які забезпечують зручні інструменти для створення фізично коректних симуляцій, візуалізації результатів та реалізації алгоритмів навчання з підкріпленням.

2. Ефективність для 2D-моделювання

Двовимірний простір є спрощеною, але достатньою моделлю для аналізу ефективності управління БПЛА. На відміну від складних 3D-симуляторів, таких як Unreal Engine, AirSim або Gazebo, Python потребує значно менших обчислювальних ресурсів. Це знижує вимоги до апаратного забезпечення та дозволяє проводити експерименти швидше.

3. Зниження складності і часу розробки

Симуляції в 2D-просторі, створені на Python, не вимагають обробки великих обсягів даних, характерних для тривимірних середовищ. Наприклад, у порівнянні з AirSim або Gazebo, Python дозволяє зосередитися на алгоритмічній частині роботи, уникаючи складностей з рендерингом і опрацюванням 3D-графіки.

4. Популярність серед розробників

Python широко використовується у сфері машинного навчання, робототехніки та наукових досліджень. Велика кількість документації та підтримка спільноти робить цей інструмент ідеальним для використання в академічних проектах.

5. Мінімальні системні вимоги

На відміну від ресурсомістких середовищ, таких як Unreal Engine або Gazebo, Python дозволяє запускати симуляції на звичайних комп'ютерах без спеціалізованого апаратного забезпечення. Це робить платформу доступною для навчальних та дослідницьких проектів.

Таким чином, використання Python для розробки симуляційного середовища забезпечує оптимальне співвідношення простоти, гнучкості та ефективності. Крім того, такий підхід значно скорочує час на підготовку експериментів і дозволяє зосередитися на дослідженні ефективності алгоритмів управління БПЛА.

3.1 Архітектура системи управління

3.1.1. Моделювання БПЛА у двовимірному просторі

Для моделювання поведінки безпілотного літального апарата у двовимірному просторі ключовими аспектами є амплітуда підйому та різниця підйому. Ці параметри визначають здатність дрона змінювати висоту, стабілізувати своє положення та здійснювати маневри.

Амплітуда підйому характеризує силу тяги, яка створюється пропелерами, і прямо впливає на швидкість вертикального переміщення апарата. У симуляції цей параметр моделюється як підсумкове значення тяги лівого і правого пропелерів, що дозволяє точно відтворити процес підйому чи опускання дрона.

Різниця підйому, у свою чергу, визначає нахил апарата, що впливає на горизонтальне переміщення. Цей показник є критично важливим для маневреності дрона, оскільки дозволяє змінювати його траєкторію та орієнтацію в просторі. У 2D-середовищі різниця підйому відображає дисбаланс тяги між лівим і правим пропелерами, завдяки чому дрон може нахилятися та рухатися в потрібному напрямку.

Використання цих параметрів у моделюванні дозволяє забезпечити реалістичну поведінку дрона в двовимірному просторі, що дає змогу точно оцінити ефективність алгоритмів управління. Амплітуда підйому відповідає за базову стабільність та виконання вертикальних завдань, тоді як різниця підйому забезпечує можливість маневрування та досягнення заданих маршрутних точок.

3.1.2. Навколишнє середовище

Для оцінки різних підходів до керування ми використовуємо 2D середовище, що моделює фізику твердого тіла, в якому 2D квадрокоптер повинен орієнтуватися на випадково згенеровані точки, представлені у вигляді повітряних кульок досягаючи їх за певну кількість часу, що допоможе нам вирахувати ефективність і продуктивність управління дроном різними агентами.

3.1.3. Математична модель руху БПЛА у 2D-просторі

Для моделювання динаміки безпілотного літального апарата (БПЛА) у 2D-середовищі використовуються рівняння жорстко-твірного тіла, які базуються на законах Ньютона. Метою є обчислення прискорень, швидкостей та позицій дрона на основі тяги пропелерів, враховуючи його кут нахилу та вплив гравітації.

1. Обчислення прискорень дрона

На кожному кроці симуляції, значення тяги лівого (T_l) та правого (T_r) пропелерів задаються в межах [−0.003; 0.083] (ці значення обрані довільно). Ці значення використовуються для обчислення прискорень дрона за такими рівняннями:

$$\ddot{x} = -\frac{(T_l + T_r)\sin(\theta)}{m}$$

$$\ddot{y} = -\frac{(T_l + T_r)\cos(\theta)}{m} + g$$

$$\ddot{\theta} = \frac{(T_r - T_l)\cdot l}{m}$$

де:

$\ddot{x}, \ddot{y}, \ddot{\theta}$ - прискорення дрона по осі *x*, *y* та кутове прискорення навколо осі *z*.

T_l, T_r - величини тяги лівого та правого пропелерів.

Θ - кут нахилу дрона відносно вертикальної осі.

m - маса дрона (константа).

g - прискорення вільного падіння (гравітація).

l — відстань від центра мас дрона до пропелерів.

Сумарна тяга обох пропелерів визначає вертикальне прискорення дрона, а їх різниця — обертальний момент, що спричиняє обертання навколо центральної осі.

3.1.4. Обчислення швидкостей та позицій

Для визначення положення дрона в наступний момент часу використовується метод кінцевих різниць. Цей метод наближено обчислює зміни швидкостей та координат дрона на основі значень прискорень за крок симуляції *dt*=1:

Оновлення швидкості:

$$\dot{x}(t+1) = \ddot{x} \cdot dt + \dot{x}(t)$$

$$\dot{y}(t+1) = \ddot{y} \cdot dt + \dot{y}(t)$$

Оновлення позиції:

$$x(t+1) = \dot{x} \cdot dt + x(t)$$

$$y(t+1) = \dot{y} \cdot dt + y(t)$$

де:

$\dot{x}$, $\dot{y}$ - швидкості дрона по осях *x* та *y*;

x, *y* - поточні координати дрона;

t – крок симуляції.

Для реалізації руху в симуляції, на кожному кроці система розраховує:

1. Прискорення дрона за заданими значеннями тяги пропелерів.

2. Швидкості на основі отриманих прискорень за методом кінцевих різниць.

3. Позицію дрона, інтегруючи його швидкість.

3.2 Оцінка (Scoring)

У симуляційному середовищі динаміка польоту дрона моделюється з частотою 60 кроків на секунду. Це означає, що за кожну секунду симуляція обчислює новий стан дрона, включаючи його прискорення, швидкість і позицію, 60 разів. Така частота дозволяє забезпечити високу точність і стабільність симуляції, що є важливим для коректного моделювання фізики польоту.

Важливим елементом симуляції є система оцінки продуктивності (scoring), яка базується на завданні навігації до випадкових точок у просторі, що представлені у вигляді "повітряних кульок". Алгоритм оцінювання полягає у наступному:

- Дрон отримує 1 бал щоразу, коли досягає цільової точки.
- Після досягнення цілі нова точка з’являється у випадковому місці в межах симуляційного середовища.
- Якщо дрон відхиляється занадто далеко від цільової точки, це вважається крахом, і апарат "відроджується" на початковій позиції з паузою у 3 секунди.

Завдання дрона — зібрати якомога більше балів протягом обмеженого часу, який у симуляції складає 100 секунд. Загальний результат визначається кількістю досягнутих цільових точок (повітряних кульок) за цей час. Така система оцінки дозволяє порівнювати ефективність різних підходів до

управління, зокрема ручного керування оператором, PID-контролера та алгоритмів навчання з підкріпленням.

Частота 60 кроків за секунду забезпечує плавну динаміку польоту та дозволяє точно фіксувати зміни у положенні та швидкості дрона, що є особливо важливим при аналізі його реакції на вхідні команди або зміну умов управління.

3.3. Агенти

3.3.1. Людина як агент

Агент в ролі людини, де користувач керує дроном, має простий принцип управління: за допомогою клавіш зі стрілками на клавіатурі, при цьому можливе одночасне натискання кількох клавіш для комбінованих дій.

Управління починається зі значень тяги для обох пропелерів: $T_l = 0.04$ і $T_r = 0.04$. Зміни в керуванні реалізуються за допомогою клавіш зі стрілками на клавіатурі, що дозволяють змінювати тягу лівого та правого пропелерів відповідно:

• Стрілка «Вгору»: тяга обох пропелерів збільшується на 0.04:

$$T_l + = 0.04;\ T_r + = 0.04$$

• Стрілка «Вниз»: тяга обох пропелерів зменшується на 0.04:

$$T_l - = 0.04;\ T_r - = 0.04$$

• Стрілка «Вліво»: тяга лівого пропелера зменшується на 0.003, а правого збільшується на 0.003, що призводить до нахилу вліво:

$$T_l - = 0.003;\ T_r + = 0.003$$

• Стрілка «Вправо»: тяга лівого пропелера збільшується на 0.003, а правого зменшується на 0.003, що викликає нахил вправо:

$$T_l + = 0.003;\ T_r - = 0.003$$

Керування дроном у симуляційному середовищі вимагає від користувача постійного контролю та корекції дій. Через фізичні властивості дрона (наявність інерції, вплив гравітації та реакції на тягу) навіть невеликі помилки можуть призвести до нестабільного польоту або втрати контролю над апаратом.

Людина, як агент, схильна до помилок через:

- Фактор реагування на зміну положення дрона.
- Труднощі з одночасним контролем вертикального підйому та горизонтального переміщення.
- Схильність до перекорекції, що може призвести до нестабільності польоту.

3.3.2. Агент: PID-контролер

PID-агент є автоматизованим підходом до управління дроном, який базується на принципах теорії управління. В основі роботи цього агента лежить використання PID-контролерів (пропорційно-інтегрально-диференціальних контролерів), які регулюють тягу пропелерів на основі позиційної помилки між поточним положенням дрона та заданою ціллю. Для реалізації управління використовуються чотири PID-контролери, що працюють у каскадній структурі.

1. Вертикальне управління

Вертикальне управління дрона здійснюється за допомогою тяги обох пропелерів. У цьому випадку PID-контролер отримує вертикальну помилку й обчислює оптимальну тягу для стабілізації руху:

Вертикальна відстань між дроном і ціллю передається до PID-контролера, який генерує оптимальну вертикальну швидкість.

Різниця між поточною вертикальною швидкістю та оптимальною вертикальною швидкістю використовується для обчислення тяги пропелерів у межах від 0 до 0.08.

Якщо обидва пропелери мають однакову тягу на рівні 0.08, дрон піднімається вертикально вгору.

2. Горизонтальне управління

Горизонтальний рух дрона забезпечується різницею в тязі між лівим і правим пропелерами. Цей процес також контролюється PID-контролерами у два етапи:

Горизонтальна відстань між дроном і ціллю передається до PID-контролера для обчислення оптимального кута нахилу дрона.

Різниця між поточним кутом нахилу й оптимальним кутом використовується для регулювання різниці тяги між пропелерами в межах від -0.003 до +0.003

Наприклад, якщо ліва тяга зменшується до -0.003, а права збільшується до +0.003, дрон нахиляється вліво і починає рухатися у відповідному напрямку.

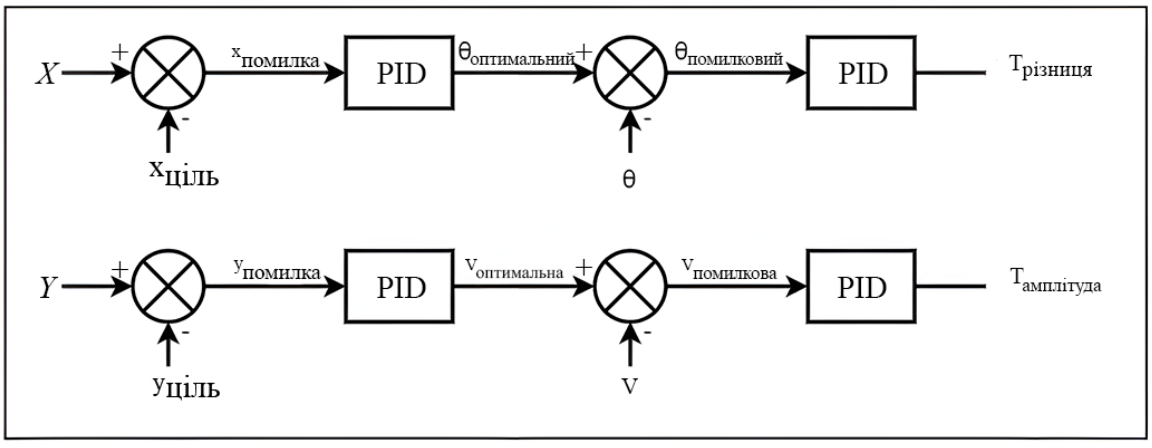


Рисунок 3.1 – Каскадна структура PID-контролерів

Система використовує чотири PID-контролери, що працюють у каскадній схемі:

• Два контролери для вертикального управління: один контролює відстань до цілі, інший стабілізує вертикальну швидкість.

• Два контролери для горизонтального управління: один регулює кут нахилу дрона, інший забезпечує баланс тяги для досягнення бажаного кута.

Програма автоматично передає PID-контролеру поточну ціль зі списку targets. Коли дрон досягає однієї цілі, програма оновлює індекс на наступний елемент у списку, і PID-контролер отримує нові координати для досягнення.

Як PID-контролер визначає, де знаходиться шар:

• Координати шара (маршрутної точки) беруться зі списку цілей (в коді «targets»), де кожен елемент — це пара значень x та y, які випадково генеруються програмою.

• Позиція дрона — це його поточні координати по x та y, які постійно оновлюються в симуляції.

Як PID-контролер розуміє задачу досягти шара:

1. Обчислюється помилка (error):

Різниця між координатами шара та позицією дрона:

• По горизонталі (вісь *x*):

$$X_{\text{помилка}} = x_{\text{ціль}} - x_{\text{позиція}}$$

• По вертикалі (вісь y):

$$y_{\text{помилка}} = y_{\text{ціль}} - y_{\text{позиція}}$$

Ці помилки передаються в PID-контролери, які розраховують потрібну тягу пропелерів, щоб дрон наблизився до цільової точки.

Перевірка досягнення шара:

У коді є умова: якщо відстань між дроном і шаром стає меншою за 50 пікселів, вважається, що дрон досягнув цілі. Тоді програма переходить до наступного шара зі списку.

Додаткова характеристика PID-контролера:

PID-контролер є ефективним інструментом для стабілізації та управління об'єктами у системах автоматичного регулювання завдяки своїй здатності мінімізувати похибку між поточним та цільовим значенням. Проте його ключовою особливістю є те, що коефіцієнти пропорційного (P), інтегрального (I) та диференціального (D) компонентів зазвичай є постійними, тобто вони не змінюються залежно від конкретної ситуації чи відстані до цілі.

Ця статичність параметрів є потенційним обмеженням PID-контролера. Зокрема:

- Коли дрон знаходиться далеко від цілі, велика помилка призводить до значного вихідного сигналу пропорційного компонента (P), що збільшує агресивність руху. Проте цього недостатньо для досягнення оптимальної швидкості, оскільки статичний інтегральний компонент (I) не адаптується до зміни умов і не враховує необхідність швидкого наближення на початковому етапі.
- При наближенні до цілі статичні коефіцієнти PID можуть призвести до перевищення або коливань, оскільки система не має гнучкості для плавного зниження швидкості та переходу до стабілізації.

Ідеальним рішенням у такій ситуації було б адаптивне налаштування параметрів P, I та D залежно від відстані до цілі чи динамічних умов польоту. Наприклад:

- На великій відстані можна збільшити інтегральний коефіцієнт (I) для досягнення більш агресивного руху.

• При наближенні до цілі можна поступово зменшити інтегральний та пропорційний коефіцієнти, що дозволить контролеру діяти більш обережно для уникнення перевищення або нестабільності.

Саме ця обмеженість PID-контролера є однією з причин, чому методи навчання з підкріпленням (RL) демонструють вищу гнучкість. RL-агенти можуть адаптувати поведінку на основі поточної ситуації, змінюючи стратегію управління залежно від відстані, швидкості та інших факторів. Це дозволяє досягти більш оптимального й ефективного керування у широкому діапазоні умов.

Таким чином, PID-контролер є надійним та ефективним у стабільних і добре налаштованих умовах, але його статичні коефіцієнти обмежують здатність до адаптації у динамічних середовищах, що особливо важливо для задач, де потрібна швидка реакція на зміну положення чи умов польоту.

3.4. Формування середовища для RL агента

Формування середовища для навчання з підкріпленням (Reinforcement Learning) є критично важливим етапом, який включає визначення доступних дій агента, спостережень та системи винагород. Оптимально сформоване середовище дозволяє агенту ефективно навчатися, балансуючи між складністю завдання та можливостями його виконання.

1. Формування дій (Action Shaping)

Дії, доступні агенту, мають бути досить повними, щоб забезпечити повний контроль над дроном, але водночас достатньо простими, щоб агент міг їх ефективно використовувати. У рамках проєкту простір дій представлений у вигляді двох дій:

• Амплітуда тяги: визначає загальну силу тяги обох пропелерів для вертикального руху дрона.

• Різниця тяги: контролює дисбаланс між лівим і правим пропелером, що дозволяє регулювати нахил дрона для горизонтального переміщення.

Цей підхід дозволяє агенту мати повний контроль над своїм рухом у 2D-просторі без зайвої складності.

2. Формування спостережень (Observation Shaping)

Спостереження визначають, яку інформацію агент отримує про стан середовища на кожному кроці симуляції. Для уникнення зайвої складності та забезпечення ефективного навчання агенту передаються лише необхідні числові значення:

- Відстань до цільової точки (шару).
- Кут до цілі: напрямок від дрона до цільової точки.
- Нахил (кутова орієнтація) дрона.
- Швидкість руху дрона (вектор швидкості).
- Кут між напрямком руху дрона та напрямком до цілі.

Такий набір спостережень виключає необхідність обробки зображень середовища, що значно спрощує завдання для агента. Замість аналізу візуальної інформації агент отримує точні числові дані, які допомагають йому швидко визначати оптимальні дії.

3. Формування винагород (Reward Shaping)

Система винагород є ключовим елементом навчання, оскільки вона допомагає агенту визначати виграшні стратегії. При формуванні винагород

важливо знайти баланс між підтримкою бажаної поведінки та відсутністю зайвого упередження. У даному середовищі система винагород реалізована наступним чином:

1) Штраф за зіткнення (крах):

Агент отримує значний негативний штраф у разі краху або виходу за межі допустимої зони. Це дозволяє агенту першочергово навчитися уникати аварій, що є базовим рівнем управління.

2) Нагорода за виживання:

На кожному кроці симуляції агент отримує невелику позитивну винагороду за перебування в активному стані. Це стимулює його продовжувати навчання та стабільно виконувати завдання.

3) Штраф за віддаленість від цілі:

Чим далі дрон знаходиться від цільової точки, тим більший негативний штраф він отримує. Це спонукає агента природно рухатися у напрямку до цілі, щоб мінімізувати відстань.

4) Нагорода за досягнення цілі:

Після досягнення цільової точки агент отримує значну позитивну винагороду. Це формує головну мету завдання — досягнення послідовних цілей (шарів).

3.4.1. Навчальне середовище для агента RL

Для навчання агента на основі підкріплення було створене середовище, яке моделює керування безпілотним літальним апаратом (БПЛА) у 2D-просторі. Середовище створене з чітко визначеними спостереженнями, діями та винагородами, що сприяють більш ефективному та послідовному навчанню агента.

Спостереження (Observations)

Щоб агент міг приймати обґрунтовані рішення, на кожному кроці симуляції він отримує ключову інформацію про свій стан та положення цілі:

1. Відстань до цілі (distance to target):

Відстань між дроном і цільовою точкою, яку потрібно досягти.

2. Кут до цілі (angle to target):

Кут між вертикальною віссю дрона та вектором, що вказує на ціль.

3. Кут нахилу (angle to up):

Кут між вертикальною віссю дрона та світовою вертикальною віссю, що дозволяє агенту враховувати вплив гравітації.

4. Швидкість дрона (velocity):

Модуль швидкості руху дрона у просторі.

5. Кутова швидкість (angle velocity):

Кут між вектором швидкості дрона та вертикальною віссю.

6. Кут між напрямком швидкості та ціллю (angle target and velocity):

Кут між вектором швидкості дрона та вектором, що вказує на ціль.

Даний набір спостережень дозволяє агенту ефективно оцінювати свою позицію у середовищі та планувати дії для досягнення цілі.

Система винагород (Reward Shaping)

Система винагород побудована таким чином, щоб стимулювати агента до стабільної та оптимальної поведінки:

1. Нагорода за виживання:

На кожному кроці симуляції агент отримує невелику позитивну винагороду:

$$\text{нагорода} = 1/60$$

Це заохочує агента підтримувати політ якомога довше.

2. Штраф за віддаленість від цілі:

Агент отримує негативний штраф за відстань до цілі, що нормалізується у межах:

$$\text{нагорода} = -\text{дистанція}/100*60$$

Це стимулює агента поступово наближатися до цільової точки.

3. Нагорода за досягнення цілі:

При досягненні цільової точки агент отримує велику позитивну винагороду:

$$\text{нагорода} = +100$$

4. Штраф за аварію:

Якщо дрон виходить за межі допустимих умов або «падає» (занадто далеко від цілі), агент отримує значний негативний штраф і епізод завершується:

$$\text{нагорода} = -1000$$

3.5. Перша спроба навчання агента з підкріпленням за допомогою алгоритму DQN

У першій спробі навчання агента навчання з підкріпленням використовувався **алгоритм DQN (Deep Q-Network)** із бібліотеки Stable-Baselines3. Метою було перевірити працездатність процесу навчання та встановити базовий рівень ефективності для подальших експериментів.

1. Утримання стабільної тяги: $T_l = 0.04$, $T_r = 0.04$
2. Підйом: $T_l = 0.08$, $T_r = 0.08$
3. Зменшення тяги: $T_l = 0$, $T_r = 0$
4. Нахил ліворуч: $T_l = 0.0394$, $T_r = 0.0406$
5. Нахил праворуч: $T_l = 0.0394$, $T_r = -0.0406$

Тип дій алгоритму DQN: дискретний простір дій. Дії були чітко визначені як фіксовані значення, що відповідають конкретним змінам тяги.

Агент обирає дію кожні 5 кроків симуляції, і обрана дія виконується протягом цих 5 кроків.

Оновлення Q-функції:

DQN оновлює нейронну мережу для апроксимації Q-значень (очікуваних винагород) для кожної дії:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

Досвід (стан, дія, винагорода, новий стан) зберігається у буфері, щоб навчатися на рандомізованих вибірках, знижуючи кореляцію між даними.

Навчання проводилося на 1 мільйоні кроків симуляції.

Стан агента (спостереження) та результати його дій відстежувалися за допомогою інструменту Weights and Biases.

Результати навчання

1. Етап початкового навчання (0–400 тис. кроків):

На початку навчання рівень дослідження середовища був дуже високим, що змушувало агента часто виконувати випадкові дії.

Це призводило до частих аварій та негативної винагороди, яка досягала -1000 через крахи.

2. Середній етап (400 тис.–800 тис. кроків):

Зі зменшенням коефіцієнта дослідження агент поступово почав навчатися уникати аварій та показував середню винагороду близько -400.

Однак нестабільність у навчанні призводила до періодичних провалів між 600 тис. та 800 тис. кроками.

3. Завершальний етап (приблизно 1 млн. кроків):

Агент досяг максимальної середньої винагороди 345 завдяки вдосконаленій стратегії, але цей прогрес був нетривалим — після цього агент знову почав часто зазнавати невдач.

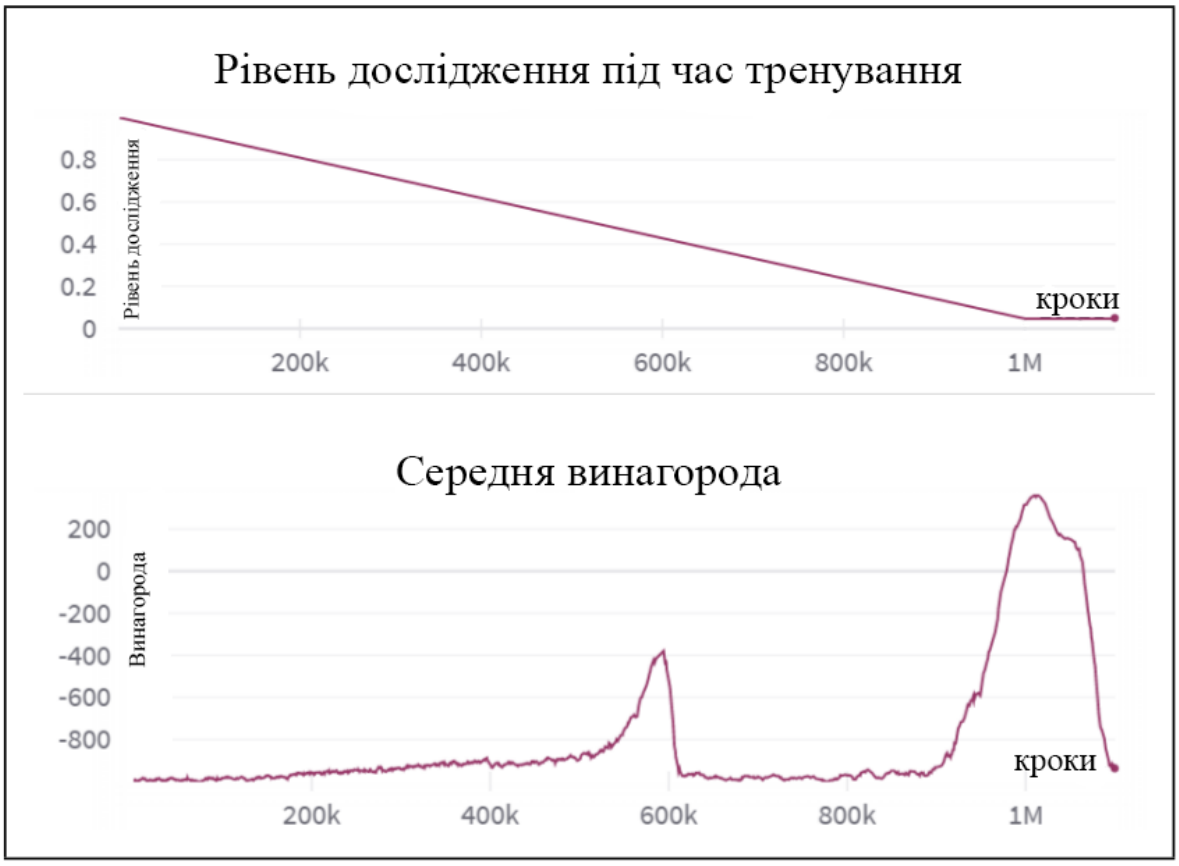


Рисунок 3.2 - Результати відстеження експерименту DQN

Попри обмеження, використання DQN дозволило агенту продемонструвати базову здатність до навчання в нестабільному середовищі. Досягнуті результати підтверджують ефективність підходу навчання з підкріпленням, але вказують на необхідність переходу до більш стабільних алгоритмів (наприклад, SAC) або розширення простору дій для покращення управління дроном.

3.6. Навчання агента з підкріпленням за допомогою алгоритму SAC

SAC є набагато досконалішим алгоритмом і на сьогоднішній день є найсучаснішим, коли мова йде про безперервні дії. Обраний простір дій наступний: вихід агента - це два плаваючих значення дія0 і дія1 (у коді програми

представлено як action0 і action1). дія0 представляє амплітуду тяги, прикладену до обох роторів. дія1 представляє відносну різницю тяги, прикладену до обох роторів. Сили тяги можуть бути отримані з дій за допомогою наступних формул (перетворення дій у рівняння сил):

- $T_l = дія_0 * 0.04 + дія_1 * 0.0006 + 0.04$
- $T_l = дія_0 * 0.04 – дія_1 * 0.0006 + 0.04$

Для програмної реалізації використовувалась бібліотека Stable-Baselines3, клас SAC.

Тип дій: Безперервний простір дій. SAC генерує два плаваючих значення (дія0 і дія1)

Відмінності в навчанні SAC:

1. Actor-Critic архітектура:

SAC використовує дві нейронні мережі (Critic) для обчислення Q-значень і Actor для вибору дій.

Actor генерує дії для максимізації нагороди з урахуванням ентропії.

Critic оцінює якість цих дій.

2. Soft Q-функція:

SAC додає ентропію до функції винагороди для збереження різноманітності дій:

$$Q_{\text{soft}}(s,a) = r + \gamma \mathbb{E}\left[Q_{\text{soft}}(s',a')\right] - \alpha \log \pi(a|s)$$

α - коефіцієнт, що контролює вплив ентропії.

3. Безперервний вихід:

На відміну від DQN, який працює з дискретними діями, SAC дозволяє агенту плавно регулювати тягу.

Таблиця 3.1 – Порівняння алгоритмів DQN і SAC у середовищі

Особливість	DQN	SAC
Тип простору дій	Дискретний (5 фіксованих дій)	Безперервний (2 плавучих значення)
Вибір дій	Вибір індексу дії зі списку	Генерація дій нейронною мережею
Відповідність діям	Фіксовані значення тяги	Обчислення тяги за формулами
Архітектура мережі	Q-функція	Actor-Critic з ентропією
Гнучкість навчання	Обмежена жорстким вибором дій	Гнучка, з безперервними діями
Ефективність навчання	Складно стабільно навчатися	Стабільне навчання з оптимізацією

Навчання відбувалося з тривалістю в 3.3 мільйони кроків.

Вже після 50 тисяч кроків агент показав винагороду вище 0, що означає, що він навчився уникати аварій. Це значно перевершує результат DQN. Винагороди продовжували зростати впродовж тренування, демонструючи, що SAC успішно навчається та адаптується до середовища.

Середня винагорода стабілізувалася на рівні 950 балів на епізод, що еквівалентно збору приблизно 9 цілей (шарів) кожні 20 секунд.

SAC продемонстрував набагато кращу продуктивність порівняно з DQN, як показано на графіку.

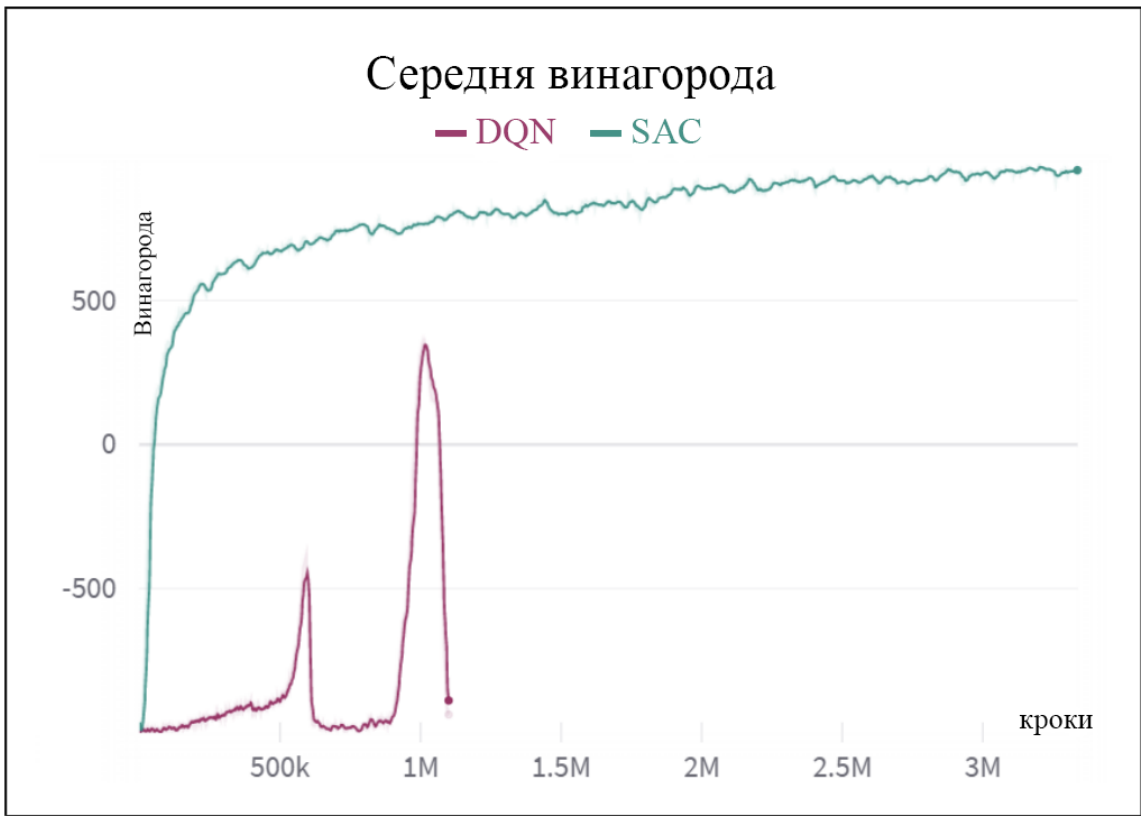


Рисунок 3.3 - Порівняння відстеження винагороди SAC алгоритму з DQN

Під час аналізу було виявлено, що дія1 часто досяла своїх граничних значень (-1 або 1). Агент працював на межі своїх можливостей, що могло обмежити його продуктивність. Для перевірки було заплановано новий запуск тестування з удосконаленою конфігурацією.

3.6.1. Запуск RL агента з більшею диференціальною тягою

У третьому запуску було використано алгоритм SAC, але з модифікацією рівня диференціальної тяги для підвищення агресивності обертання дрона. Це дозволило агенту швидше та ефективніше змінювати напрямок польоту.

Для підвищення ефективності обертання дрона було змінено коефіцієнт диференціальної тяги у формулах, що описують тягу лівого та правого пропелерів:

- $T_l = дія_0 * 0.04 + дія_1 * 0.003 + 0.04$
- $T_r = дія_0 * 0.04 – дія_1 * 0.003 + 0.04$

Збільшення коефіцієнта 0.003 у диференціальній тязі дозволяє дрону обертатися агресивніше та швидше реагувати на необхідність зміни напряму.

Навчання відбувалося з тривалістю в 5 мільйони кроків.

Агент SAC навчився використовувати нову можливість швидкого обертання для точнішого досягнення цільових точок (шарів).

За результатами навчання на графіку середня винагорода стабілізувалась приблизно на рівні 1200 балів за епізод. Це відповідає збору приблизно 12 шарів кожні 20 секунд, що є значним покращенням продуктивності.

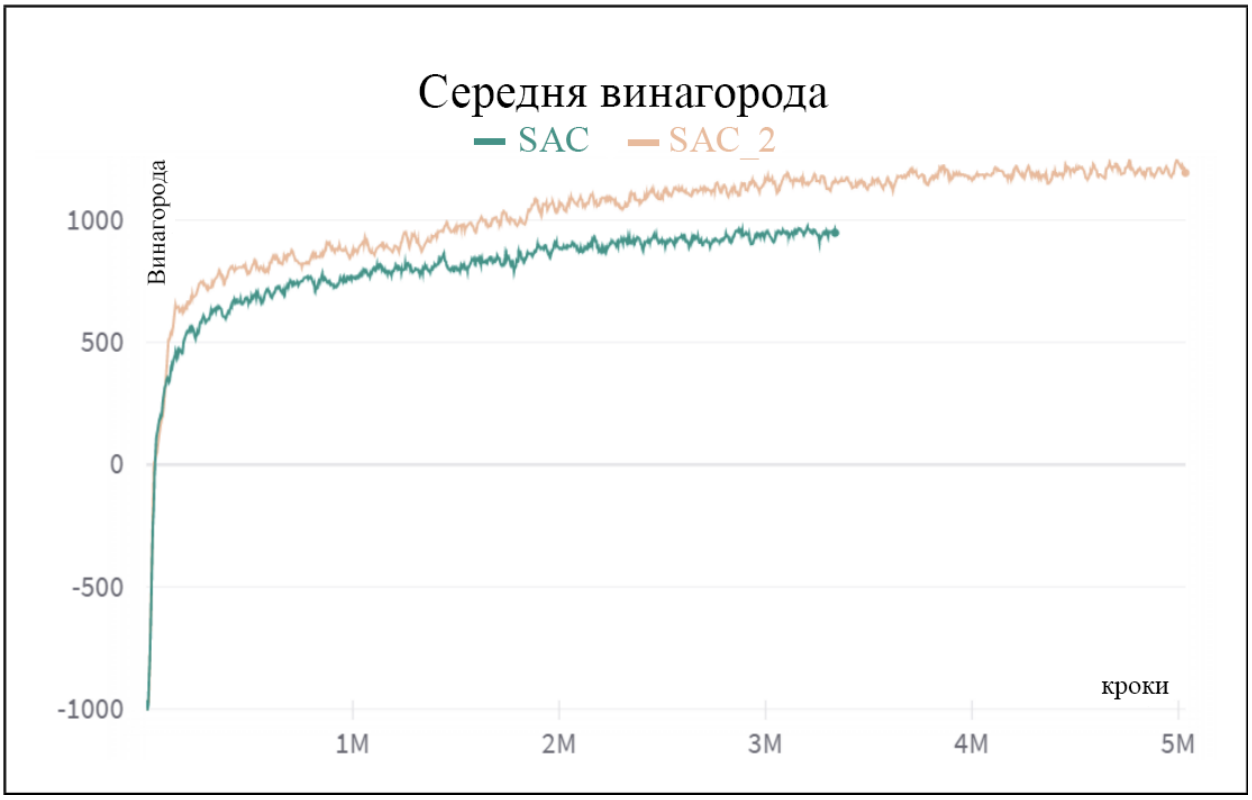


Рисунок 3.4 - Порівняння відстеження винагороди SAC алгоритму з удосконаленим алгоритмом SAC_2

Як видно на графіку, SAC_2 демонструє стабільний прогрес із мінімальними коливаннями винагороди на пізніх етапах навчання.

3.7. Порівняння продуктивності агентів.

У середовищі було проведено порівняльний аналіз трьох агентів:

Таблиця 3.2 – Порівняння результатів з різними агентами

Агент	Результат (бал)
Людина	55

PID	42
SAC_2	66

Результат уявляє собою кількість досягнутих шарів (балів) за 100 секунд.

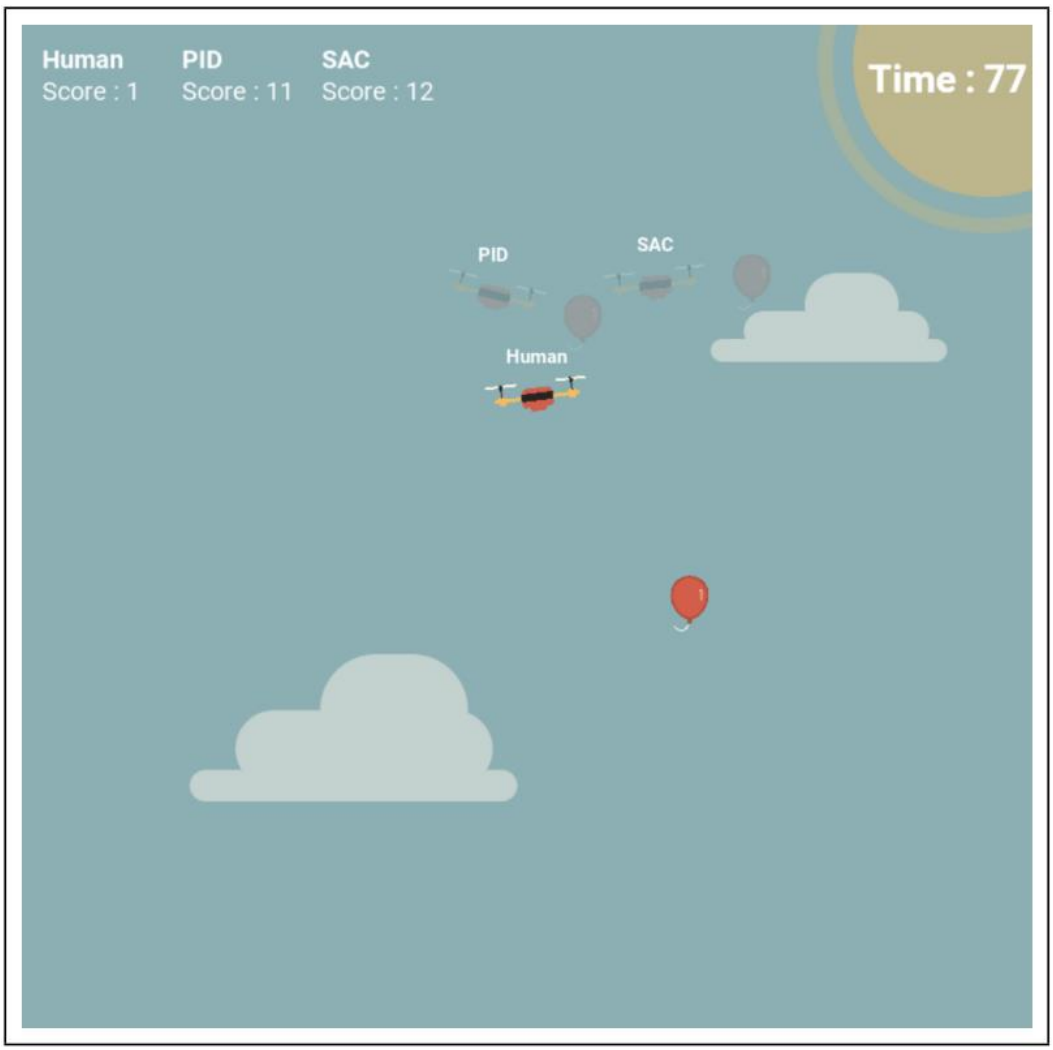


Рисунок 3.5 – Скріншот з середовища під час одночасного тестування різних агентів

SAC перевершив продуктивність як PID-контролера, так і людини завдяки своїй гнучкості та можливості адаптації.

Висновок

Модифікація формул для вищої диференціальної тяги дозволила агенту SAC досягти кращої продуктивності та підвищити ефективність керування дроном. Агент навчався швидше реагувати на необхідність зміни напряму та демонстрував вищі результати порівняно з іншими агентами, зокрема PID-контролером та людиною.

Це підтверджує, що адаптація параметрів керування у поєднанні з SAC дозволяє досягти високої продуктивності у створеному 2D середовищі.

ВИСНОВКИ

У межах дослідження було створено двовимірне симуляційне середовище, яке використовується для тестування різних підходів до управління безпілотним літальним апаратом (БПЛА). Основна увага приділялася порівнянню ефективності алгоритмів навчання з підкріпленням (Reinforcement Learning, RL) із традиційними методами, зокрема PID-контролером та ручним управлінням оператором.

Проведене дослідження дозволило отримати наступні результати:

Імітація поведінки БПЛА у двовимірному просторі є ефективним підходом для початкового тестування систем управління. Врахування ключових фізичних параметрів, таких як гравітація, інерція та диференційна тяга пропелерів, дозволяє створити реалістичну модель, яка відтворює базові особливості польоту дрона.

Порівняльний аналіз агентів показав, що алгоритми з підкріпленням мають значну перевагу у швидкості та адаптивності.

1. RL-агент навчився досягати цільових точок значно швидше, демонструючи гнучкість у виборі оптимальних траєкторій польоту.

2. PID-контролер, хоч і забезпечував стабільний політ, мав обмеження у швидкості досягнення цілей через консервативний характер своїх дій.

3. Людина-оператор, незважаючи на здатність адаптуватися до умов, часто припускалася помилок, пов’язаних з нестабільним керуванням та переоцінкою траєкторій.

Переваги та обмеження RL-алгоритмів:

Використання навчання з підкріпленням дозволяє системам управління адаптуватися до різних ситуацій у динамічному середовищі. Однак ефективність RL сильно залежить від тривалості навчання та налаштувань середовища.

Практична значущість розробленої симуляції:

Створене середовище є ефективним інструментом для дослідження методів управління БПЛА в умовах обмежених ресурсів. Воно дозволяє проводити безпечні та відтворювані експерименти, які можна масштабувати для реальних систем.

Даний досвід може бути використаний для практичних експериментів у реальному житті з переміщенням дрона в умовах двовимірного середовища (переміщення по осях X та Y). Для цього необхідно, щоб середовище не створювало опору при русі БПЛА (наприклад, відсутність вітру або інших збурень).

Для інтеграції алгоритмів у реальний дрон слід враховувати:

1. Сам дрон та його комплектуючі для визначення необхідної потужності тяги та габаритів пристрою.

2. Наявність чотирьох пропелерів для забезпечення умов стабільного переміщення у двох осях.

3. Розмір простору, який повинен відповідати умовам тестування та забезпечувати достатнє поле для виконання маневрів.

У практичних умовах для визначення маршрутної точки дрона слід використовувати:

1. Датчики виявлення цілей, такі як лідар чи камера, для визначення місця розташування цілі.

2. Датчики локалізації, що отримують інформацію від сторонніх джерел про координати точки, до якої повинен переміститися дрон.

Однак, у реальних умовах також необхідно адаптувати та, можливо, переписати алгоритми, враховуючи специфіку роботи реального обладнання та додаткові компоненти БПЛА. Це зумовлено тим, що симуляція працює у спрощеному середовищі, де немає впливу непередбачуваних факторів, як-от вітер, турбулентність чи перешкоди на маршруті. У реальному світі алгоритми повинні включати обробку даних з додаткових сенсорів, наприклад:

1. Системи орієнтації та стабілізації: використання даних з інерціальних датчиків (IMU), компаса чи гіроскопа для коригування курсу.

2. Обробка відеопотоків: аналіз зображень з камер для виявлення та ідентифікації цільових точок.

3. Використання SLAM (Simultaneous Localization and Mapping): побудова карти навколишнього середовища для навігації в умовах, де неможливе використання GPS.

Зміна структури алгоритмів також може бути необхідною через обмежену обчислювальну потужність бортової системи дрона. У симуляції обробка даних виконується на зовнішніх платформах з високою продуктивністю, тоді як у реальних БПЛА необхідно оптимізувати код для швидкої обробки на бортовому контролері.

Таким чином, створене симуляційне середовище є важливим етапом для тестування та валідації алгоритмів управління БПЛА. Проте, для їх практичного застосування у реальних умовах необхідно враховувати додаткові сенсори, системи стабілізації та фізичні обмеження обладнання, що потребує відповідної адаптації алгоритмів та оптимізації програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ankit Goel, Abdulazeez Mohammed Salim, Ahmad Ansari, Sai Ravela, and Dennis Bernstein. “Adaptive digital pid control of a quadcopter with unknown dynamics”, 2020.

2. Volodymyr Mnih, Adria Puigdom ` enech Badia, Mehdi Mirza, ` Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. 2016.

3. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing atari with deep reinforcement learning”, 2013.

4. Dingqi Zhang, Antonio Loquercio, Xiangyu Wu, Ashish Kumar, Jitendra Malik, and Mark W. Mueller. “A zero-shot adaptive quadcopter controller”, 2022.

5. Посилання на інтернет-ресурс: Андрієвський Б. Р., Попов А. М., Михайлов В. А., Попов Ф. А. – “Застосування методів штучного інтелекту для управління польотом безпілотних літальних апаратів” URL: https://cyberleninka.ru/article/n/primenenie-metodov-iskusstvennogo-intellekta-dlya-upravleniya-poletom-bespilotnyh-letatelnyh-apparatov/viewer

6. Сілкін А. А., Райко Г. О., Гнєзділов В. Ю. – "Огляд сучасних програмних симуляторів для підготовки операторів безпілотних систем мультикоптерного типу"

7. Посилання на інтернет-ресурс: Смирнов С. В., Кветкін Г. О., Ажгірєвич І. Л., Зінов'єв П. Д., Ізмайлов-Перкін О. В. – "Розробка програмно-апаратного комплексу імітаційного моделювання автоматичної посадки безпілотного літального апарата" URL: https://cyberleninka.ru/article/n/razrabotka-programmno-apparatnogo-kompleksa-imitatsionnogo-modelirovaniya-avtomaticheskoy-posa-dki-bespilotnogo-letatelnogo

8. Посилання на інтернет-ресурс: Borland – "Модель БПЛА з пілотажно-навігаційним комплексом" URL: https://hub.exponenta.ru/post/nelineynaya-model-bpla-s-pilotazhno-navigatsionnym-kompleskom181

9. Посилання на інтернет-ресурс: Kong Chun-Wei – " Deep Neural Network altitude controller for Quadcopter" URL: https://jordan787878.github.io/firstweb/6dofQuadcopter/6dofQuad_DNN.html

10. Osim Kumar Pal, MD Sakib Hossain Shovon, M. F. Mridha – "A Comprehensive Review of AI-enabled Unmanned Aerial Vehicle: Trends, Vision, and Challenges"

11. K. Sujatha, NPG. Bhavani, VictoSudha George, T. Kalpatha Reddy, A. Kannan, A. Ganesan, Naresh – "AI-Based Controller Design for Autopilot System of UAV".

12. Посилання на інтернет-ресурс: “Python Software” URL: https://www.python.org/

ДОДАТОК А

2D-середовище

```
import pygame
import math
import random

# Constants
WIDTH, HEIGHT = 800, 600
FPS = 60
DRONE_RADIUS = 15
BALLOON_RADIUS = 10
TARGET_RADIUS = 5
MAX_THRUST = 0.1
GRAVITY = 0.0005

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

class Drone:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.vx = 0
```

```
            self.vy = 0
            self.angle = 0
            self.thrust_left = 0
            self.thrust_right = 0

        def update(self):
            # Apply thrust to velocity
            self.vx += (self.thrust_left + self.thrust_right) *
math.sin(math.radians(self.angle))
            self.vy -= (self.thrust_left + self.thrust_right) *
math.cos(math.radians(self.angle))

            # Apply gravity
            self.vy += GRAVITY

            # Update position
            self.x += self.vx
            self.y += self.vy

            # Keep within bounds
            self.x = max(0, min(WIDTH, self.x))
            self.y = max(0, min(HEIGHT, self.y))

        def draw(self, screen):
            pygame.draw.circle(screen, BLUE, (int(self.x), int(self.y)),
DRONE_RADIUS)

    class Balloon:
        def __init__(self):
```

```
        self.x = random.randint(0, WIDTH)
        self.y = random.randint(0, HEIGHT)

    def draw(self, screen):
        pygame.draw.circle(screen, RED, (self.x, self.y), BALLOON_RADIUS)

class Game:
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((WIDTH, HEIGHT))
        pygame.display.set_caption("Balloon Game")
        self.clock = pygame.time.Clock()

        self.drone = Drone(WIDTH // 2, HEIGHT // 2)
        self.balloon = Balloon()
        self.running = True

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.render()
            self.clock.tick(FPS)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
```

```
            keys = pygame.key.get_pressed()
            if keys[pygame.K_LEFT]:
                self.drone.thrust_left = MAX_THRUST
            else:
                self.drone.thrust_left = 0

            if keys[pygame.K_RIGHT]:
                self.drone.thrust_right = MAX_THRUST
            else:
                self.drone.thrust_right = 0

        def update(self):
            self.drone.update()

            # Check collision with balloon
            distance = math.hypot(self.drone.x - self.balloon.x, self.drone.y -
self.balloon.y)
            if distance < DRONE_RADIUS + BALLOON_RADIUS:
                self.balloon = Balloon()

        def render(self):
            self.screen.fill(WHITE)
            self.drone.draw(self.screen)
            self.balloon.draw(self.screen)
            pygame.display.flip()

    if __name__ == "__main__":
        game = Game()
        game.run()
```

ДОДАТОК Б

Код для управління дроном людиною

```
import pygame
import os
from pygame.locals import *
from math import sin, cos, pi, sqrt
from random import randrange

# Game constants
FPS = 60
WIDTH = 800
HEIGHT = 800

# Physics constants
gravity = 0.08
# Propeller force for UP and DOWN
thruster_amplitude = 0.04
# Propeller force for LEFT and RIGHT rotations
diff_amplitude = 0.003
# By default, thruster will apply a force of thruster_mean
thruster_mean = 0.04
mass = 1
# Length from center of mass to propeller
arm = 25

# Initialize Pygame, load sprites
FramePerSec = pygame.time.Clock()
```

```
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))

# Loading player and target sprites
player_width = 80
player_animation_speed = 0.3
player = []
for i in range(1, 5):
    image = pygame.image.load(
        os.path.join(
            "assets/balloon-flat-asset-pack/png/objects/drone-sprites/drone-"
            + str(i)
            + ".png"
        )
    )
    image.convert()
    player.append(
        pygame.transform.scale(image, (player_width, int(player_width * 0.30)))
    )

target_width = 30
target_animation_speed = 0.1
target = []
for i in range(1, 8):
    image = pygame.image.load(
        os.path.join(
            "assets/balloon-flat-asset-pack/png/balloon-sprites/red-plain/red-plain-"
            + str(i)
```

```
                + ".png"
            )
        )
        image.convert()
        target.append(
            pygame.transform.scale(image, (target_width, int(target_width * 1.73)))
        )

    # Loading background sprites
    cloud1 = pygame.image.load(
        os.path.join("assets/balloon-flat-asset-pack/png/background-elements/cloud-
1.png")
    )
    cloud2 = pygame.image.load(
        os.path.join("assets/balloon-flat-asset-pack/png/background-elements/cloud-
2.png")
    )
    sun = pygame.image.load(
        os.path.join("assets/balloon-flat-asset-pack/png/background-
elements/sun.png")
    )
    cloud1.set_alpha(124)
    (x_cloud1, y_cloud1, speed_cloud1) = (150, 200, 0.3)
    cloud2.set_alpha(124)
    (x_cloud2, y_cloud2, speed_cloud2) = (400, 500, -0.2)
    sun.set_alpha(124)

    # Loading fonts
    pygame.font.init()
```

```
info_font = pygame.font.Font("assets/fonts/Roboto-Regular.ttf", 30)
respawn_font = pygame.font.Font("assets/fonts/Roboto-Bold.ttf", 90)

# Initialize physics variables
(angle, angular_speed, angular_acceleration) = (0, 0, 0)
(x_position, x_speed, x_acceleration) = (400, 0, 0)
(y_position, y_speed, y_acceleration) = (400, 0, 0)
x_target = randrange(200, 600)
y_target = randrange(200, 600)

# Initialize game variables
target_counter = 0
time = 0
step = 0
time_limit = 30
dead = False
respawn_timer_max = 3
respawn_timer = 3

# Game loop
while True:
    pygame.event.get()

    # Display background
    screen.fill((131, 176, 181))

    x_cloud1 += speed_cloud1
    if x_cloud1 > WIDTH:
        x_cloud1 = -cloud1.get_width()
```

```
screen.blit(cloud1, (x_cloud1, y_cloud1))

x_cloud2 += speed_cloud2
if x_cloud2 < -cloud2.get_width():
    x_cloud2 = WIDTH
screen.blit(cloud2, (x_cloud2, y_cloud2))

screen.blit(sun, (630, -100))

time += 1 / 60
step += 1

if dead == False:
    # Initialize accelerations
    x_acceleration = 0
    y_acceleration = gravity
    angular_acceleration = 0

    # Calculate propeller force in function of input
    thruster_left = thruster_mean
    thruster_right = thruster_mean
    pressed_keys = pygame.key.get_pressed()
    if pressed_keys[K_UP]:
        thruster_left += thruster_amplitude
        thruster_right += thruster_amplitude
    if pressed_keys[K_DOWN]:
        thruster_left -= thruster_amplitude
        thruster_right -= thruster_amplitude
    if pressed_keys[K_LEFT]:
```

```
    thruster_left -= diff_amplitude
if pressed_keys[K_RIGHT]:
    thruster_right -= diff_amplitude

# Calculate accelerations according to Newton's laws of motion
x_acceleration += (
    -(thruster_left + thruster_right) * sin(angle * pi / 180) / mass
)
y_acceleration += (
    -(thruster_left + thruster_right) * cos(angle * pi / 180) / mass
)
angular_acceleration += arm * (thruster_right - thruster_left) / mass

# Calculate speed
x_speed += x_acceleration
y_speed += y_acceleration
angular_speed += angular_acceleration

# Calculate position
x_position += x_speed
y_position += y_speed
angle += angular_speed

# Calculate distance to target
dist = sqrt((x_position - x_target) ** 2 + (y_position - y_target) ** 2)

# If target reached, respawn target
if dist < 50:
    x_target = randrange(200, 600)
```

```
            y_target = randrange(200, 600)
            target_counter += 1

        # If to far, die and respawn after timer
        elif dist > 1000:
            dead = True
            respawn_timer = respawn_timer_max
    else:
        # Display respawn timer
        respawn_text = respawn_font.render(
            str(int(respawn_timer) + 1), True, (255, 255, 255)
        )
        respawn_text.set_alpha(124)
        screen.blit(
            respawn_text,
            (
                WIDTH / 2 - respawn_text.get_width() / 2,
                HEIGHT / 2 - respawn_text.get_height() / 2,
            ),
        )

        respawn_timer -= 1 / 60
        # Respawn
        if respawn_timer < 0:
            dead = False
            (angle, angular_speed, angular_acceleration) = (0, 0, 0)
            (x_position, x_speed, x_acceleration) = (400, 0, 0)
            (y_position, y_speed, y_acceleration) = (400, 0, 0)
```

```
# Ending conditions
if time > time_limit:
    break

# Display target and player
target_sprite = target[int(step * target_animation_speed) % len(target)]
screen.blit(
    target_sprite,
    (
        x_target - int(target_sprite.get_width() / 2),
        y_target - int(target_sprite.get_height() / 2),
    ),
)

player_sprite = player[int(step * player_animation_speed) % len(player)]
player_copy = pygame.transform.rotate(player_sprite, angle)
screen.blit(
    player_copy,
    (
        x_position - int(player_copy.get_width() / 2),
        y_position - int(player_copy.get_height() / 2),
    ),
)

# Update text
target_target = info_font.render(
    "Collected : " + str(target_counter), True, (255, 255, 255)
)
screen.blit(target_target, (20, 20))
```

```
    time_text = info_font.render(
        "Time : " + str(int(time_limit - time)), True, (255, 255, 255)
    )
    screen.blit(time_text, (20, 60))

    pygame.display.update()
    FramePerSec.tick(FPS)

print("Score : " + str(target_counter))
```

ДОДАТОК В

Код для ПІД-контролеру

```
class PID:
    def __init__(self, KP, KI, KD, saturation_max, saturation_min):
        self.kp = KP
        self.ki = KI
        self.kd = KD
        self.error_last = 0
        self.integral_error = 0
        self.saturation_max = saturation_max
        self.saturation_min = saturation_min

    def compute(self, error, dt):
        derivative_error = (error - self.error_last) / dt
        self.integral_error += error * dt
        output = (
            self.kp * error + self.ki * self.integral_error + self.kd * derivative_error
        )
        self.error_last = error
        if output > self.saturation_max and self.saturation_max is not None:
            output = self.saturation_max
        elif output < self.saturation_min and self.saturation_min is not None:
            output = self.saturation_min
        return output
```

ДОДАТОК Г

Тренування SAC агента

```
import os

import numpy as np
from stable_baselines3 import SAC
from stable_baselines3.common.monitor import Monitor
from stable_baselines3.common.callbacks import CheckpointCallback
import wandb
from wandb.integration.sb3 import WandbCallback

from env_SAC import droneEnv

params = ["gamma", "learning_rate", "buffer_size", "tau", "batch_size"]
gamma_range = []
learning_rate_range = []
buffer_size_range = [1, 500, 5000, 50000, 500000]
tau_range = [0.00001, 0.001, 0.1, 0.5, 0.99]
batch_size_range = [1, 32, 64, 128, 256]
ranges = [
    gamma_range,
    learning_rate_range,
    buffer_size_range,
    tau_range,
    batch_size_range,
]
defaults = [0.99, 0.0003, 50000, 0.005, 64]
```

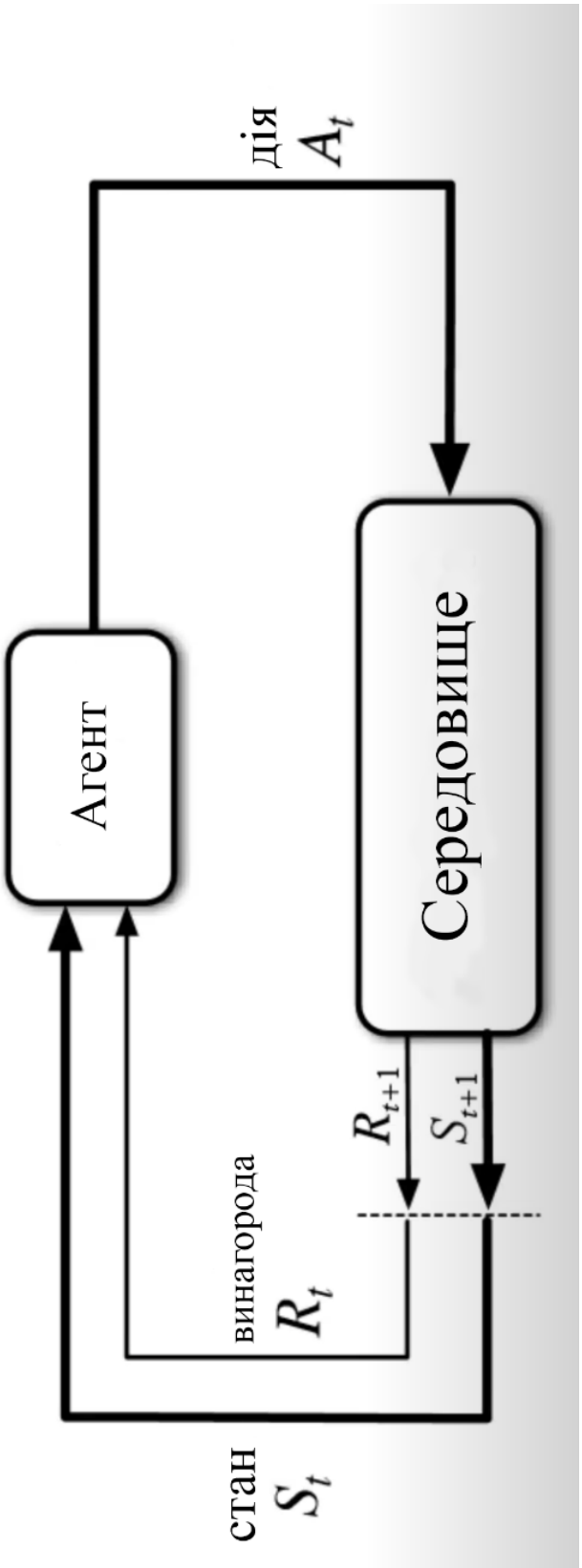
```
for i in range(len(params)):
    for j in range(len(ranges[i])):
        # Set hyperparameters
        gamma = defaults[0]
        learning_rate = defaults[1]
        buffer_size = defaults[2]
        tau = defaults[3]
        batch_size = defaults[4]

        if params[i] == "gamma":
            gamma = ranges[i][j]
        elif params[i] == "learning_rate":
            learning_rate = ranges[i][j]
        elif params[i] == "buffer_size":
            buffer_size = ranges[i][j]
        elif params[i] == "tau":
            tau = ranges[i][j]
        elif params[i] == "batch_size":
            batch_size = ranges[i][j]

        run = wandb.init(
            # CHANGE THIS to quadai-params
            project="quadai-params",
            sync_tensorboard=True,
            monitor_gym=True,
            name=f"{params[i]}_{ranges[i][j]}",
            config={
                "gamma": gamma,
```

```
        "learning_rate": learning_rate,
        "buffer_size": buffer_size,
        "tau": tau,
        "batch_size": batch_size,
    },
)

# Create log dir
log_dir = "tmp/"
os.makedirs(log_dir, exist_ok=True)

# Create and wrap the environment
env = droneEnv(False, False)
env = Monitor(env, log_dir)

# Create SAC agent
model = SAC(
    "MlpPolicy",
    env,
    verbose=2,
    tensorboard_log=log_dir,
    gamma=gamma,
    learning_rate=learning_rate,
    buffer_size=buffer_size,
    tau=tau,
    batch_size=batch_size,
)

# Create checkpoint callback
```

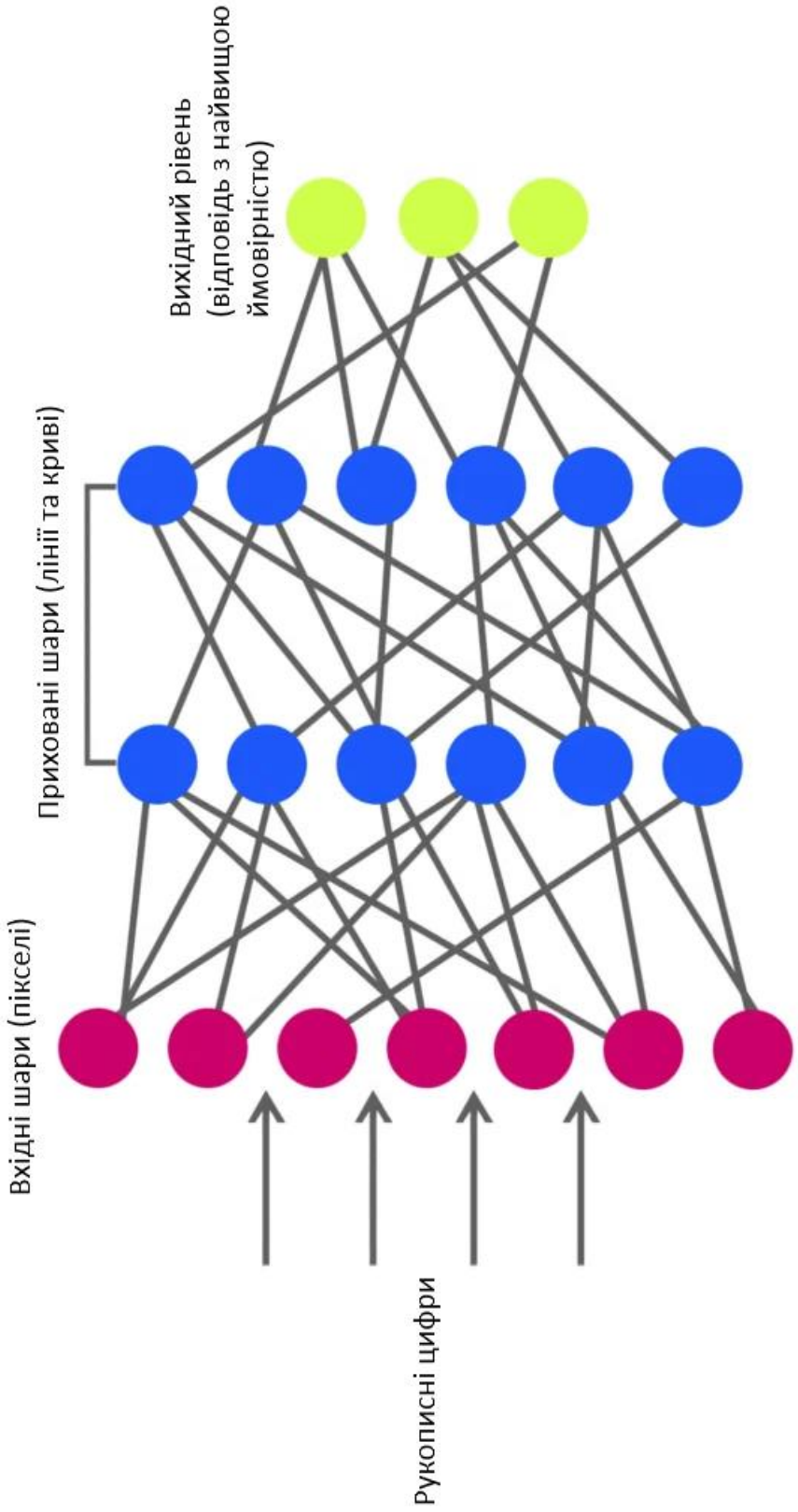
```
checkpoint_callback = CheckpointCallback(
    save_freq=100000, save_path=log_dir, name_prefix="rl_model_lr"
)

# Train the agent
model.learn(
    # CHANGE THIS TO 500000
    total_timesteps=500000,
    callback=[
        checkpoint_callback,
        WandbCallback(
            # CHANGE THIS TO 5000
            gradient_save_freq=5000,
            model_save_path=f"models/{run.id}",
            model_save_freq=100000,
            verbose=2,
        ),
    ],
)

# Close
env.close()
run.finish()
```

ДОДАТОК Д

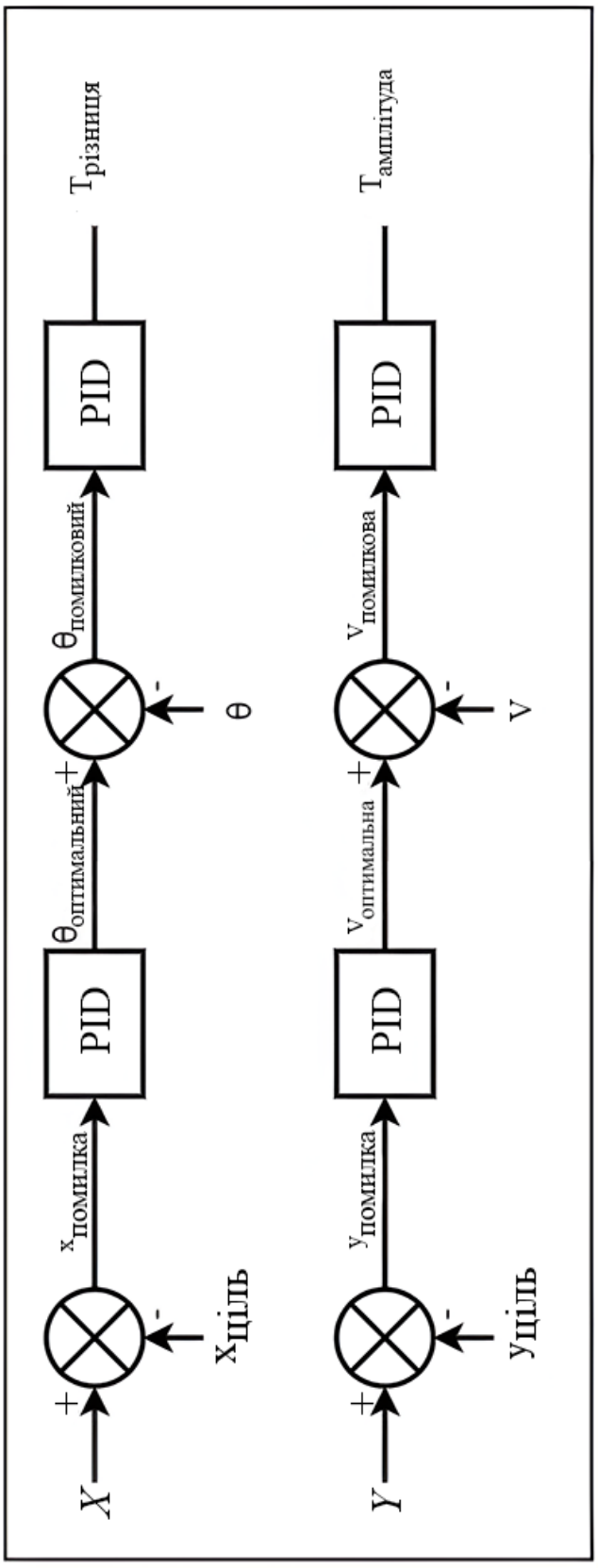
ПРИНЦИП РОБОТИ НАВЧАННЯ З ПІДКРІПЛЕННЯМ



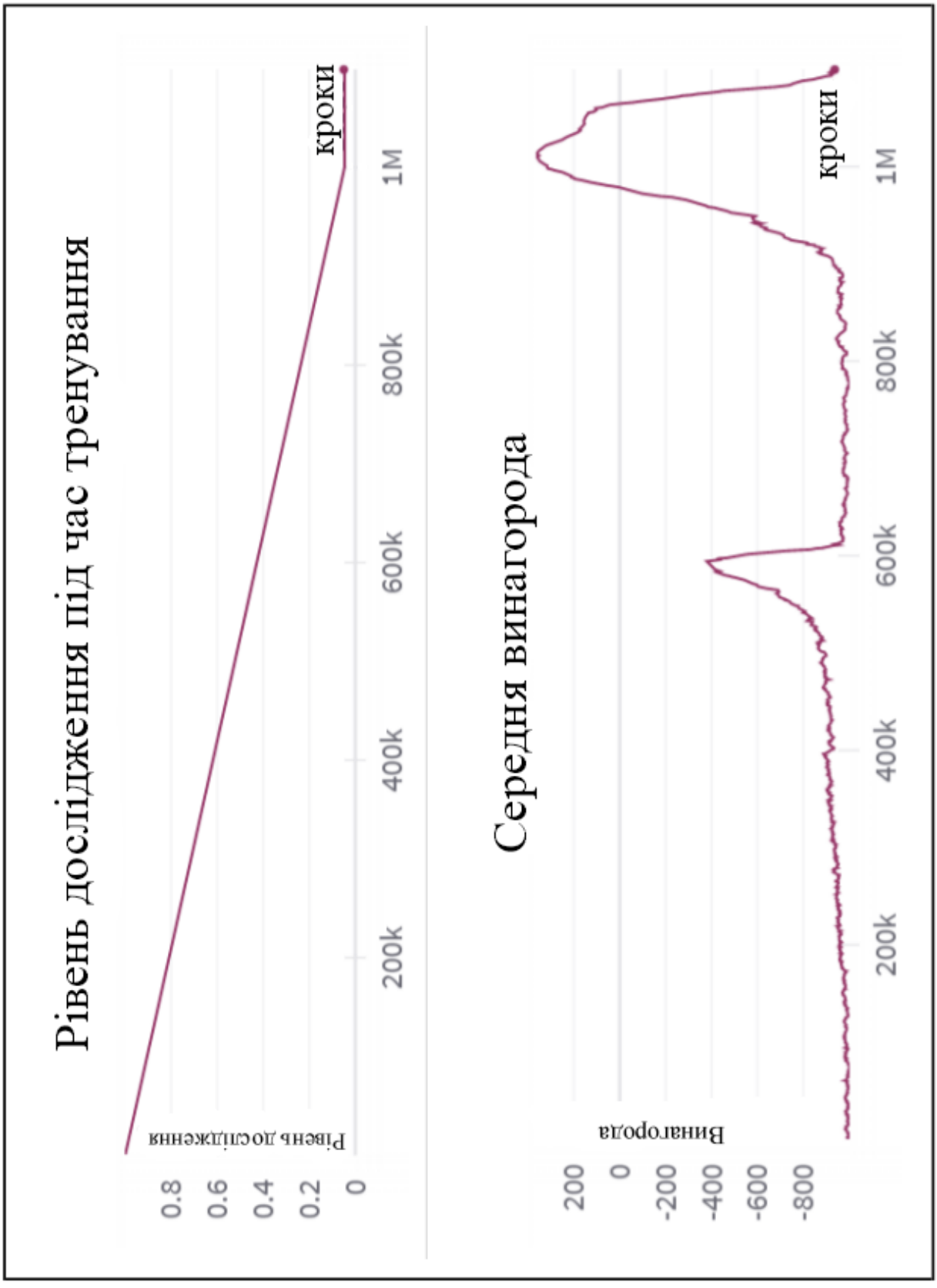
ДОДАТОК Е
УЯВЛЕННЯ СТРУКТУРИ НЕЙРОННОЇ МЕРЕЖІ



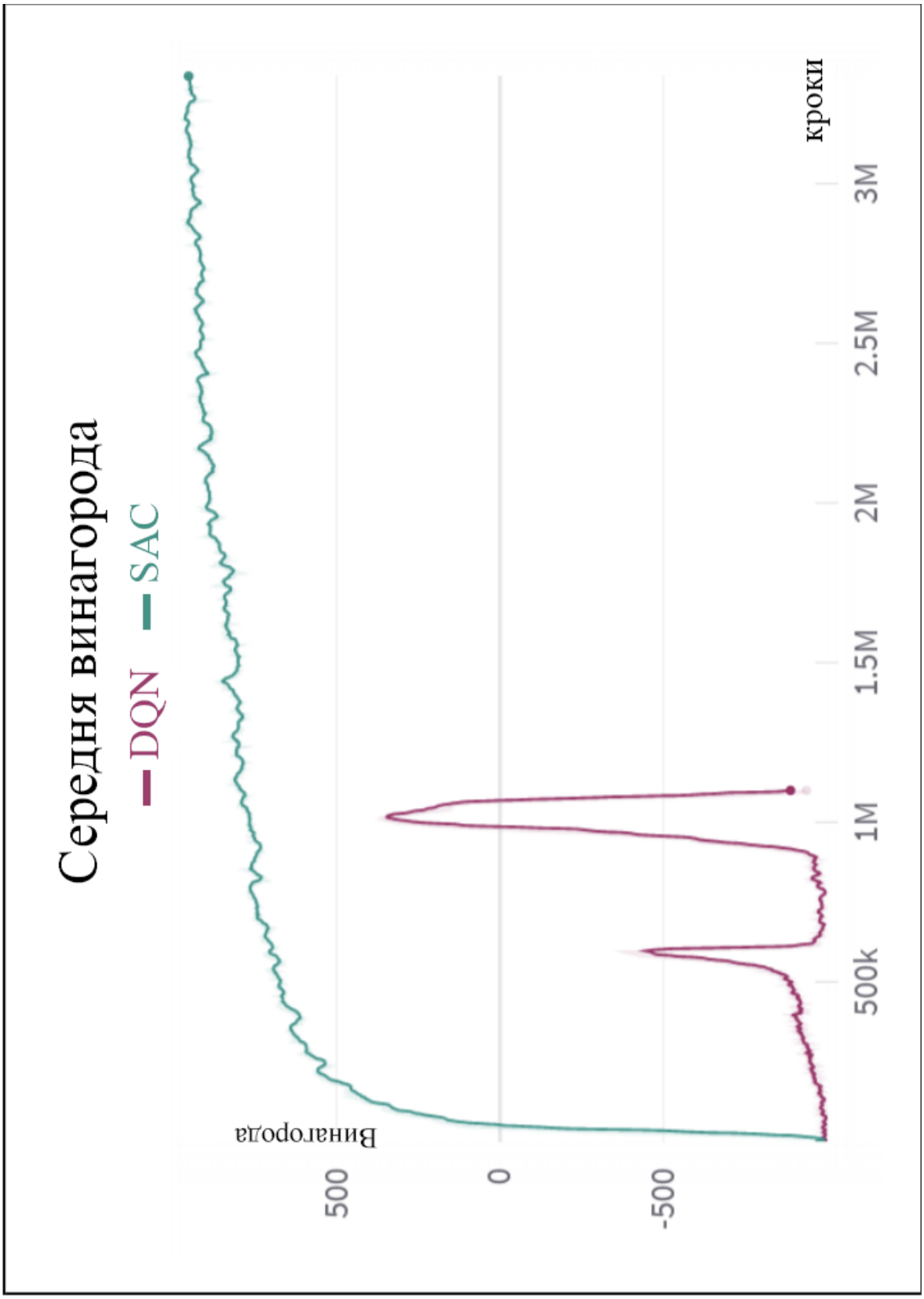
ДОДАТОК Ж
КАСКАДНА СТРУКТУРА PID-КОНТРОЛЕРІВ



ДОДАТОК З
РЕЗУЛЬТАТИ ВІДСТЕЖЕННЯ ЕКСПЕРИМЕНТУ DQN



ДОДАТОК И
ПОРІВНЯННЯ ВІДСТЕЖЕННЯ ВИНАГОРОДИ SAC АЛГОРИТМУ З DQN



ДОДАТОК І
ПОРІВНЯННЯ ВІДСТЕЖЕННЯ ВИНАГОРОДИ SAC АЛГОРИТМУ З УДОСКОНАЛЕНИМ АЛГОРИТМОМ SAC_2