

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Альперт Максим Іоганович

УДК 004.9; 004.4; 004.8; 519.8

ДИСЕРТАЦІЯ

**Інформаційна технологія керування безпілотними апаратами на базі ігрового
підходу та нейронних мереж**

126 – Інформаційні системи та технології

12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії.

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

_____ М.І. Альперт

Науковий керівник: Вікторія Валеріївна Онищенко, д.т.н., проф.

КИЇВ – 2025

АНОТАЦІЯ

Альперт М.І. Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 126 – Інформаційні системи та технології в галузі знань 12 – Інформаційні технології. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2025.

Дисертаційна робота присвячена розробці методів керування безпілотними апаратами та їх взаємодії, що дозволяють підвищити успішність пошуково-рятувальних операцій та доставки життєво-необхідних вантажів у важкодоступні місця, в умовах обмеженості ресурсів. Тому існує потреба у створенні нової інформаційної технології взаємодії безпілотних апаратів (БпА) різних типів (наземного та літального) із застосуванням нейронних мереж в умовах ризику та невизначеності, реалізація якої запропонована на базі ігрового підходу, а також у злагодженому функціонуванні програмно-технічного комплексу.

У **першому розділі** висвітлено стислий огляд сучасного стану розвитку інформаційних технологій, їх всебічне застосування в різних галузях: виробництві, сільському господарстві, науці, освіті, зв'язку та управлінні.

Наведено огляд інформаційних технологій, новітніх наукових досліджень, напрямів, їх практичної реалізації на даний час. Виконано аналіз позитивних та проблемних аспектів практичної реалізації розглянутих наукових досягнень.

Проведено аналіз актуального стану інформаційних технологій керування безпілотними апаратами із застосуванням ігрового підходу та нейронних мереж.

Надано огляд сучасних прикладних ігрових методів, побудованих на базі апарату теорії ігор, з використанням яких вже отримана низка перспективних результатів та проаналізовані останні наукові та практичні дослідження в цій галузі.

Розглянуто різні типи нейронних мереж, їх застосування у різноманітних сферах життя. Також розглянуто використання нейронних мереж у поєднанні з навчанням з підкріпленням, що дозволяє приймати оптимальні рішення керування у системі.

Використання нейронних мереж у поєднанні з ігровими методами має перспективу розвитку для підвищення безпеки виконання місії безпілотними апаратами.

Запит суспільства для вирішення складних задач із застосуванням безпілотних апаратів визначає попит на подальший розвиток та вдосконалення інформаційних технологій, у тому числі при розробці нових підходів, методів та моделей керування безпілотними апаратами.

У **другому розділі** виконаного наукового дослідження здійснено огляд, порівняння, аналіз існуючих методів, надано обґрунтування обрання апарату дослідження, який застосовано в процесі проєктування інформаційної технології керування безпілотними апаратами (БпА); отримано нові математичні моделі та розроблена архітектура інформаційної системи.

Математичні моделі типу кооперативних ігрових моделей спроможні адекватно відтворюють керування коаліцією безпілотних апаратів, надають змогу розглядати безпілотні апарати в якості гравців, враховують технічні обмеження та способи взаємодії БпА між собою в умовах коаліції.

Проведено огляд і порівняльний аналіз централізованого і децентралізованого методів керування БпА.

Зроблено висновок, що централізований метод керування коаліцією БпА у порівнянні з децентралізованим має переваги у контролі, простоті та безпеці.

Враховуючи переваги централізованого методу над децентралізованим, в науковому дослідженні обрано централізований метод керування БпА.

В результаті проведення дисертаційного дослідження отримано дві математичні моделі.

1. Побудовано комбіновану централізовано-кооперативну математичну ігрову модель керування БпА в процесі наукового дослідження.

В процесі створення цієї моделі використано підходи і апарат ігрового методу. В якості загальної функції виграшу обрано функцію, яка визначає головну ціль гри – успішне виконання місії від її початку і до кінця. Результатом моделювання є знаходження максимуму загальної функції виграшу.

Експериментальне порівняння результатів оптимізації функцій успішності, отриманої комбінованої централізовано-кооперативної математичної ігрової моделі, з однієї сторони з відповідними результатами функцій успішності ідеальної та децентралізованої моделі, з іншої сторони, дає змогу зробити висновок про значно більшу ефективність функції виграшу отриманої моделі.

Практичне застосування комбінованої централізовано-кооперативної математичної ігрової моделі дозволить при виконанні місії обрати ігровий підхід; обрати централізовано-кооперативний метод керування БпА; визначитися з вибором найбільш підходящих технічних характеристик коаліції БпЛА та БпНА.

2. Розроблена оптимізаційна ігрова математична модель керування безпілотними апаратами із застосуванням нейронних мереж в умовах ризику та невизначеності.

В науковому дослідженні детально розглянуто підхід до вирішення проблеми обмеженості та/або неточності вхідних даних, відповідно до якого розглядаються два різних види ситуацій, які зазвичай необхідно дослідити, а саме: рішення приймаються в умовах ризику; рішення приймаються в умовах невизначеності.

В залежності від характеру небажаних ситуацій залежить вибір апарату дослідження. Якщо рішення приймаються в умовах ризику, то появу небажаної перешкоди моделюють із застосуванням апарату теорії ймовірностей. Якщо рішення приймається в умовах невизначеності, функцію розподілу ймовірностей знайти практично неможливо.

В науковому дослідженні розглядаються обидві ситуації, які виникають при моделюванні керування БпЛА та БпНА: ризик появи небажаного явища, зокрема весняного/осіннього бездоріжжя; раптова поява перешкоди (ями) в процесі виконання місії (невизначеності).

Визначена функція розподілу ймовірностей для кожного i -го ребра з використанням метеорологічних даних за визначений період.

Результат застосування першої частини оптимізаційної ігрової математичної моделі – отримання шляху з виключеними проблемними ребрами (ділянками шляху).

Друга частина моделі вирішує проблему подолання раптової перешкоди (невизначеності) на шляху в процесі виконання місії. Запропоновано вирішувати цю проблему за допомогою можливостей нейронних мереж та навчання з підкріпленням.

Проведено аналіз ефективності оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

Найбільшу ефективність з розглянутих реальних моделей має модель, отримана в результаті наукового дослідження, яка врахувала і ризики, і невизначеність.

У третьому розділі проведено порівняльний аналіз відомих симуляторів, які необхідні для проведення практичних експериментів. Розглянуто найпоширеніші симулятори (Ardupilot, PX4, ROS, Gazebo, Microsoft AirSim), а також проаналізовані їхні переваги та недоліки.

Для проведення досліджень та експериментів обрано симулятор Microsoft AirSim.

В науковому дослідженні запропоновано такі модифікації симулятора Microsoft AirSim: прибрано центральний контролер та окремий обчислювальний мікрокомп'ютер; додано нові фізичні контролери; додано блок вбудованих алгоритмів; налагоджено взаємодію вбудованих алгоритмів з API Layer.

Для проведення експериментів наукового дослідження використано два типи БпА (наземний та літальний), які вбудовані у Microsoft AirSim.

В дисертаційному дослідженні наведено огляд різноманітних середовищ, які дозволяють використовувати обраний симулятор та які потенційно можна

використовувати для перевірки ігрового підходу і нейронних мереж у керуванні безпілотними апаратами.

Для проведення експериментів з виявлення великих статичних та раптових перешкод обрано комбіновану згорткову нейронну мережу, яка поєднує швидкість розпізнавання MobileNet, точність виявлення об'єктів за допомогою SSD та переваги трансферного навчання.

Проведені експерименти склалися з навчання згорткової нейронної мережі розпізнавати великі статичні перешкоди за допомогою трансферного навчання.

Відповідно до отриманих наукових результатів запропонована у дослідженні комбінована згорткова нейронна мережа ефективно розпізнала нові об'єкти, а саме – великі статичні блоки та раптові перешкоди (вибоїни, ями).

Застосоване навчання з підкріпленням в науковому дослідженні надало можливість успішно подолати великі статичні та раптові перешкоди наземним БПА.

У **четвертому розділі** дисертаційного дослідження багато уваги приділено питанню обробки та збереження даних стосовно керування безпілотними апаратами.

В якості інструменту проєктування багатошарової архітектури взаємодії з БД обрано мову програмування Python та бібліотека FastAPI.

В процесі дослідження розроблено діаграму класів для інформаційної технології керування безпілотними апаратами, побудовано модифікований образ Kafka Connect, а також розроблено схему інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Результати наукового дослідження мають як теоретичне, так і практичне значення.

Наукове значення отриманих результатів полягає в отриманні нових методів та математичних моделей для проєктування інформаційних технологій керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

На основі одержаних в результаті дослідження алгоритмів розроблено програмне забезпечення і проведено комп'ютерне моделювання різних можливих ситуацій. Запропоновані методи і алгоритми можуть стати основою для керування взаємодією безпілотних апаратів.

Ключові слова: інформаційна технологія, теорія ігор, ігровий підхід, ігровий метод, математична модель, пошук найкоротшого шляху, машинне навчання, нейронна мережа, трансферне навчання, розпізнавання зображень, навчання з підкріпленням, БпЛА, БпНА.

Список публікацій здобувача

1. Alpert M. (2024). Using Game Theory to Improve Drone Operations. In Control Systems and Computers (Vol. 1, pp. 57-61). <https://doi.org/10.15407/csc.2024.01.057>.
2. Альперт М. І., Онищенко В.В. (2023). Пошук найкращого алгоритму найкоротшого шляху для розумної валізи. Управління розвитком складних систем (Вип. 55, с. 92-97). <http://dx.doi.org/10.32347/2412-9933.2023.55.92-97>.
3. Alpert M., Onyshchenko V. (2022). Recognition of Potholes with Neural Network Using Unmanned Ground Vehicles. In Advances in Computer Science for Engineering and Education (Vol. 134., pp. 209–220). https://doi.org/10.1007/978-3-031-04812-8_18.
4. Літвінова Н., Альперт М., Погульський А. (2021). Підвищення ефективності обміну даними сутностей у реляційному представленні та їх обробки. Технічні науки та технології (Вип. 1(23), с. 81–86). [https://doi.org/10.25140/2411-5363-2021-1\(23\)-81-86](https://doi.org/10.25140/2411-5363-2021-1(23)-81-86).

ABSTRACT

Alpert M.I. Information technology for controlling unmanned vehicles based on a game approach and neural networks. – Qualifying scientific work is presented on the rights of the manuscript.

Thesis for the Doctor of Philosophy degree in specialty 126 – Information systems and technologies in the field of knowledge 12 – Information Technologies. – National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, 2025.

The dissertation is devoted to the development of methods for controlling unmanned vehicles and their interaction that allow to increase the success of search and rescue operations and the delivery of vital cargo to hard-to-reach places in conditions of limited resources. That is why there is a need to create a new information technology for the interaction of unmanned vehicles of various types (ground and aerial) using neural networks in conditions of risk and uncertainty. The implementation is proposed on the basis of a game approach and neural networks, as well as the coordinated functioning of the software and hardware complex.

The **first chapter** provides a brief overview of the current state of information technology development, their comprehensive application in various industries: production, agriculture, science, education, communications and management.

An overview of information technologies is provided including the latest scientific research, directions, and their practical implementation at the present time. An analysis of the positive and problematic aspects of the practical implementation of the considered scientific achievements is performed.

The current state of information technologies is highlighted for controlling unmanned vehicles using the game approach and neural networks in this work.

An overview of modern applied gaming methods is provided which are built on the basis of the apparatus of game theory. A number of promising results have already been obtained and the latest scientific and practical research is analyzed in this area.

Different types of neural networks and their application are considered in various spheres of life. It is also considered the use of neural networks in combination with reinforcement learning, which allows making optimal control decisions in the system.

The use of neural networks in combination with game methods has a development perspective for increasing the safety of mission execution by unmanned vehicles.

The request of society for solving complex tasks using unmanned vehicles (UV) determines the demand for further development and improvement of information technologies, including in the development of new approaches, methods and models for controlling unmanned vehicles.

The **second chapter** of the scientific research provides a comparison and analysis of existing methods. A justification was provided for the choice of the research apparatus that was applied in the process of designing information technology for controlling unmanned vehicles. New mathematical models were obtained and the architecture of the information system was determined.

Mathematical models such as cooperative game models are capable of adequately reproducing the coalition control of UV. These models make it possible to consider unmanned vehicles as players and can take into account technical limitations and methods of unmanned vehicles interaction with each other in coalition conditions.

A review and comparative analysis of centralized and decentralized UV control methods were conducted.

It was concluded that the centralized UV coalition control method has advantages in control, simplicity and security compared to the decentralized one.

Considering the advantages of a centralized method over a decentralized one, the centralized method of controlling UV was chosen in this scientific study.

Two mathematical models were obtained as a result of the dissertation research.

1. A combined centralized-cooperative mathematical game model of UV control was built in the process of scientific research.

The approaches and apparatus of the game method were used in the process of creating this model. The general payoff function determines the main goal of the game -

successful completion of the mission from its beginning to the end. The result of the modeling is finding the maximum of the general payoff function.

Experimental comparison of the optimization results of success functions is obtained by the combined centralized-cooperative mathematical game model with the corresponding results of the success functions of the ideal and decentralized model. It allows to conclude that the payoff function of the obtained model is much more effective.

Practical application of the combined centralized-cooperative mathematical game model will allow to choose a game approach when performing a mission; choose a centralized-cooperative method of UV control; decide on the choice of the most suitable technical characteristics of the coalition of unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV).

2. An optimization game mathematical model of controlling unmanned vehicles using neural networks in conditions of risk and uncertainty has been developed.

The scientific study examines the approach to solving the problem of limited and/or inaccurate input data in detail where two different types of situations are considered. For example, decisions are made at risk conditions and decisions are made under uncertainty conditions.

The choice of research apparatus depends on the nature of the undesirable situations. If decisions are made at risk conditions, then the appearance of an undesirable obstacle is modeled using the apparatus of probability theory. If a decision is made under uncertainty conditions, then it is practically impossible to find the probability distribution function.

The scientific study considers both situations that arise when modeling UAV and UGV control: the risk of the appearance an undesirable phenomenon. For example, it can be spring/autumn off-road conditions and/or the sudden appearance of an obstacle (pit) during the mission (uncertainty).

The probability distribution function for each i -th edge is determined using meteorological data for a certain period.

The first part of the optimization game mathematical model is obtaining a path with excluded problematic edges (path sections).

The second part of the model solves the problem of overcoming a sudden obstacle (uncertainty) on the path during the mission. It is proposed to solve this problem using the capabilities of neural networks and reinforcement learning.

The effectiveness of the optimization game mathematical model of UAV and UGV control using neural networks in conditions of risk and uncertainty has been analyzed.

The most effective of the considered real models is the model obtained as a result of scientific research that has taken into account both risks and uncertainty.

The **third chapter** provides a comparative analysis of known simulators that are necessary for conducting practical experiments. The most common simulators are overviewed (Ardupilot, PX4, ROS, Gazebo, Microsoft AirSim). Their advantages and disadvantages are analyzed.

The Microsoft AirSim simulator was chosen for conducting research and experiments.

The scientific study proposed the following modifications of the Microsoft AirSim simulator: the central controller and a separate computing microcomputer are removed; new physical controllers are added; a block of built-in algorithms is added; the interaction of built-in algorithms with the API Layer is established.

Two types of UV (ground and aerial) are used for the scientific study experiments. Both of them are built into Microsoft AirSim.

The dissertation study provides an overview of various environments that allow the use of the selected simulator and which can potentially be used to test the game approach and neural networks in controlling unmanned vehicles.

A combined convolutional neural network was selected for the experiments on detecting large static and sudden obstacles. It combines the recognition speed of MobileNet, the accuracy of object detection using SSD and the advantages of transfer learning.

The experiments conducted consisted of training the convolutional neural network to recognize large static obstacles using transfer learning.

According to the obtained scientific results, the combined convolutional neural network proposed in the study effectively recognized new objects. These were large static blocks and sudden obstacles (potholes, pits).

The applied reinforcement learning in the scientific study has made it possible to successfully overcome large static and sudden obstacles by UGV.

The **fourth chapter** pays attention to the issue of processing and storing data related to the control of unmanned vehicles.

A description is provided to the structure of the software for working with the control center database and the UV database.

The Python programming language and the FastAPI library were used as a tool for designing a multi-layer architecture for interaction with the database.

A class diagram for the information technology of controlling unmanned vehicles was developed during the research process. A modified Kafka Connect image has been built. In addition, a scheme of the information technology of controlling unmanned vehicles has been developed based on a game approach and neural networks.

The results of the scientific research have both theoretical and practical significance.

The scientific significance of the results lies in obtaining new methods and mathematical models for the design of information technologies of controlling unmanned vehicles based on a game approach and neural networks.

Software was developed and computer modeling of various possible situations was carried out based on the algorithms obtained as a result of the research. The proposed methods and algorithms can become the basis for controlling the interaction of unmanned vehicles.

ЗМІСТ

ЗМІСТ	14
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	17
ВСТУП.....	19
1 АНАЛІЗ СУЧАСНИХ РІШЕНЬ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ.....	24
1.1 Принципи сучасних інформаційних технологій керування безпілотними апаратами	24
1.2 Огляд основних понять ігрових методів	31
1.3 Кооперативні ігрові методи та їх застосування у моделюванні складних систем	32
1.4 Некооперативні ігрові методи	35
1.5 Динамічні ігрові методи	36
1.6 Нейронні мережі.....	39
1.7 Аналіз поточного стану інформаційних технологій керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.....	46
1.8 Висновки до розділу 1	51
2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ.....	52
2.1 Обґрунтування використання кооперативного ігрового методу у керуванні безпілотними апаратами.....	52
2.2 Централізований і децентралізований метод керування безпілотними апаратами	56
2.3 Комбінована централізовано-кооперативна математична ігрова модель керування БпЛА та БпНА	59
2.4 Огляд механізмів прийняття рішень для пошуку оптимального шляху. Результати проведення порівняльного аналізу стандартних алгоритмів прийняття рішень	65
2.5 Оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.....	70
2.6 Опис архітектури інформаційної системи керування з використанням нейронних мереж та ігрових підходів.....	83
2.7 Висновки до розділу 2	92

3 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ У КЕРУВАННІ БЕЗПІЛОТНИМИ АПАРАТАМИ.....	97
3.1 Аналіз можливостей сучасних симуляторів. Вибір симулятора для проведення експериментів наукового дослідження.....	97
3.2 Впровадження симуляції.....	105
3.3 Моделювання та впровадження симуляційного середовища.....	109
3.4 Актуальні способи навчання та налаштування нейронних мереж у рамках інформаційних технологій	114
3.5 Комплексне застосування трансферного навчання для вирішення проблеми розпізнавання великих статичних та раптових перешкод	118
3.6 Навчання з підкріпленням.....	125
3.7 Висновки до розділу 3	128
4 СТРУКТУРА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ	132
4.1 Обробка і збереження даних в процесі керування безпілотними апаратами	132
4.2 Розробка діаграми класів для інформаційної технології керування безпілотними апаратами.....	136
4.3 Забезпечення системи обміну даними при керуванні безпілотними апаратами. Інтеграція Apache Kafka у систему інформаційної технології керування БпА .	139
4.4 Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.....	149
4.5 Висновки до розділу 4	152
ВИСНОВКИ.....	154
СПИСОК ЛІТЕРАТУРИ.....	161
ДОДАТОК А. АКТ ВПРОВАДЖЕННЯ У НАВЧАЛЬНИЙ ПРОЦЕС	169
ДОДАТОК Б. ЛІСТИНГ КОДУ КОМБІНОВАНОЇ ЦЕНТРАЛІЗОВАНО-КООПЕРАТИВНОЇ МОДЕЛІ КЕРУВАННЯ БПЛА ТА БПНА.....	170
ДОДАТОК В. ЛІСТИНГ КОДУ ОПТИМІЗАЦІЙНОЇ ІГРОВОЇ МАТЕМАТИЧНОЇ МОДЕЛІ КЕРУВАННЯ БПЛА ТА БПНА В УМОВАХ РИЗИКУТА НЕВИЗНАЧЕНОСТІ	173
ДОДАТОК Г. ЛІСТИНГ КОДУ ТРАНСФЕРНОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ	177

ДОДАТОК Д. ФРАГМЕНТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ.....	179
ДОДАТОК Е. ДІАГРАМА КЛАСІВ ДЛЯ НАЛАГОДЖЕННЯ ОБМІНУ ДАНИМИ МІЖ БЕЗПІЛОТНИМИ АПАРАТАМИ	187
ДОДАТОК Ж. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ	188

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

- ІТ – інформаційні технології
- БпА – безпілотні апарати
- UV – unmanned vehicles – безпілотні апарати
- UAV – unmanned aerial vehicle – безпілотний літальний апарат
- UGV – unmanned ground vehicle – безпілотний наземний апарат
- БпЛА – безпілотний літальний апарат
- БпНА – безпілотний наземний апарат
- ПЗ – програмне забезпечення
- GPS – Global Positioning System – система глобального позиціювання
- МБпЛА – мікролітальні безпілотні апарати
- БпЛС – безпілотні літальні системи
- ASR – Automatic Speech Recognition – автоматичне розпізнавання мови
- ШНМ – штучні нейронні мережі
- ЙНМ – ймовірнісна нейронна мережа
- ЗНМ – згорткова нейронна мережа
- ДНМ – динамічна нейронна мережа
- БД – база даних
- ROS – Robot Operating System – операційна система для роботів
- ШІ – штучний інтелект
- API – Application Programming Interface – прикладний програмний інтерфейс
- LiDAR – Light Identification, Detection and Ranging – технологія отримання та обробки інформації про віддалені об’єкти за допомогою оптичних систем
- SLAM – simultaneous localization and mapping – одночасна локалізація і картографування
- IMU – Inertial measurement unit – інерційний вимірювальний пристрій
- TPV – Third-Person View – графічна перспектива з фіксованої відстані позаду до об’єкта

FPV – First-Person View – графічна перспектива «очима» об'єкта

YOLO – you look only once – ПЗ для розпізнавання об'єктів на зображенні

SSD – Single Shot MultiBox Detector – ПЗ для розпізнавання об'єктів на зображенні

MobileNet – ПЗ для розпізнавання об'єктів на зображенні

Labellmg – ПЗ для анотування зображень

SQL – Structured Query Language – мова структурованих запитів

WSL – Windows Subsystem for Linux – запуск виконуваних файлів операційної системи Linux в середовищі Windows

cmd – command line interpreter – консоль операційної системи Microsoft Windows

ip – internet protocol – міжмережевий протокол

HTTP – hyper-text transfer protocol – протокол передачі гіпер-тексту

СУБД – система управління базами даних

JDBC – Java DataBase Connectivity – з'єднання з базами даних на Java

FastAPI – веб-фреймворк для створення API

ВСТУП

Актуальність теми. Успішність виконання багатьох актуальних задач в умовах необхідності економії людських ресурсів, обмеженості фінансових та енергетичних витрат передбачає зростаючу потребу у розробці нових інформаційних технологій, зокрема, стосовно використання безпілотних апаратів. Найбільше це стосується проведення пошуково-рятувальних операцій, а також доставки життєво-необхідних вантажів у важкодоступні місця.

Безпілотні апарати за останні роки набули широких функціональних можливостей і знайшли застосування у багатьох галузях. З огляду перспективності застосування безпілотних апаратів, значно активізувалися дослідження щодо їх ефективного використання. Зростаюча складність задач, що постає перед цим видом техніки, вимагає координації дій безпілотних апаратів для досягнення єдиної мети під час виконання спільної місії.

Під місією розуміємо таке. Місія – швидко, безпечно та в умовах обмежених людських, фінансових та енергетичних ресурсів виконати пошуково-рятувальну операцію та доставити вантажі з пункту відправки до місця призначення.

Питання взаємодії безпілотних апаратів між собою, злагодженість їх дій потребує нових рішень.

Мета роботи. Підвищення ефективності виконання пошуково-рятувальних операцій та доставки вантажів безпілотними апаратами з пункту відправки до місця призначення шляхом розробки нової інформаційної технології керування безпілотними апаратами та їх взаємодії в умовах ризику та невизначеності, а також у злагодженому функціонуванні запропонованого програмно-технічного комплексу.

Програмно-технічний комплекс складається з:

1. БпЛА;
2. БпНА;
3. Центр керування;
4. Оператор;

5. Штучна нейронна мережа, навчена для виконання задачі розпізнавання перешкод;

6. Алгоритм пошуку найкоротшого шляху A-Star;

7. GPS;

8. Розроблене унікальне програмне забезпечення (ПЗ) відповідно до нової комбінованої централізовано-кооперативної математичної моделі керування БПА різних типів (БпЛА та БпНА);

9. Розроблене унікальне ПЗ відповідно до нового комбінованого методу оптимізації маршруту руху БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

Дослідницькі задачі, які необхідно вирішити для досягнення даної мети:

– виконати аналіз існуючих ігрових методів та математичних моделей для керування безпілотними апаратами (БПА);

– розробити комбіновану централізовано-кооперативну математичну модель керування БПА різних типів (літальним та наземним);

– розробити новий ігровий метод взаємодії (кооперації) БпНА та БпЛА, який відрізняється об'єднанням переваг як централізованого так і кооперативного методів керування, що дозволяє здійснити вибір безпілотних апаратів за їх технічними характеристиками в умовах обмеженості ресурсів;

– розрахувати і дослідити ефективність виявлення перешкод для безпілотного літального та наземного апаратів шляхом трансферного навчання нейронної мережі для класифікації зображень за наявності різних умов, у тому числі несприятливих;

– розробити оптимізаційну ігрову математичну модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності;

– розробити новий ігровий метод керування БпЛА та БпНА, який відрізняється одночасним врахуванням ризику та невизначеності, що дозволяє отримати безпечний оптимальний маршрут руху безпілотних апаратів;

- розробити стратегію подолання перешкод для безпілотного наземного апарату з використанням навчання з підкріпленням;
- розробити інформаційну технологію керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Об’єкт дослідження – процес керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Предмет дослідження – моделі, методи та технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Методи дослідження – теорія ігор, теорія ймовірностей, математична статистика, теорія графів, теорія алгоритмів, штучні нейронні мережі, експериментальні методи, метод навчання з підкріпленням, методи об’єктно-орієнтованого та процедурного програмування.

Наукова новизна отриманих результатів:

- розроблено комбіновану централізовано-кооперативну математичну модель керування БПА різних типів (літальним та наземним);
- розроблено новий ігровий метод взаємодії (кооперації) БпНА та БпЛА, який відрізняється об’єднанням переваг як централізованого так і кооперативного методів керування, що дозволяє здійснити вибір безпілотних апаратів за їх технічними характеристиками в умовах обмеженості ресурсів;
- розроблено оптимізаційну ігрову математичну модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності;
- розроблено новий ігровий метод керування БпЛА та БпНА, який відрізняється одночасним врахуванням ризику та невизначеності, що дозволяє отримати безпечний оптимальний маршрут руху безпілотних апаратів.

Характерною особливістю запропонованого дослідження є використання ігрового та ймовірнісного підходів до керування об’єктами із застосуванням нейронних мереж в умовах ризику та невизначеності. Такий підхід до керування динамічними об’єктами забезпечує досягнення найкращого результату при несприятливих умовах, оскільки він поєднує переваги двох зазначених підходів.

Використання ігрового підходу з ймовірнісним є найбільш ефективним для безпілотних апаратів, при виконанні задач пошуку та порятунку людей.

Практичне значення отриманих результатів. На основі одержаних в результаті дослідження алгоритмів розроблено програмне забезпечення і проведено комп'ютерне моделювання різних можливих ситуацій. Запропоновані методи і алгоритми можуть стати основою для керування взаємодією безпілотних апаратів.

Розроблене програмне забезпечення та комп'ютерне моделювання дозволяє розглянути та протестувати різні сценарії, з урахуванням несприятливих умов польоту БпЛА та пересування БпНА, їх взаємодії між собою та з оточуючим середовищем. Це дозволить визначити найбільш оптимальні стратегії керування, які забезпечать найкращі результати в реальних умовах експлуатації. Такий підхід дозволить врахувати несподівані ситуації та забезпечить стабільну роботу безпілотних апаратів у несприятливих умовах. Комп'ютерне моделювання може бути застосовано: для навчання операторів, які керують БпА, а також для навчання штучного інтелекту, що використовуються при управлінні БпА. Системи керування, розроблені на основі цих алгоритмів, можуть стати важливим кроком у напрямку розвитку автономних систем, що працюють у співпраці з людьми.

Наукове дослідження за темою дисертації проводилось у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» у відповідності до Переліку пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок на період до 2026 року.

Інформаційну технологію розроблено в рамках науково-дослідницької роботи: «Інформаційні технології і системи підтримки прийняття рішень. Високопродуктивні комп'ютерні системи та мережі» з державним реєстраційним номером: 0124U002078; дата реєстрації: 27.02.2024.

Пріоритетним напрямом науково-технічної діяльності є «Розвиток сучасних інформаційних, комунікаційних технологій, робототехніки».

Особистий внесок здобувача. Результати, отримані автором особисто у процесі дослідження та представлені до захисту у дисертаційній роботі. У наукових

працях, опублікованих у співавторстві, автору належать: [35] – інтеграція теорії ігор у процес оптимізації вибору станцій зарядки для безпілотних апаратів; [36] – розробка та аналіз алгоритмів пошуку найкоротшого маршруту для БпНА; [37] – розроблення та дослідження методу використання нейронних мереж для розпізнавання дорожніх дефектів за допомогою БпНА; [38] – досліджено методи обміну даними та їх обробки в реляційному представленні для підвищення ефективності роботи інформаційних систем.

Публікації. За результатами дисертаційних досліджень опубліковано 4 наукові публікації, серед яких:

- 1 публікація, яка проіндексована в Scopus базі даних;
- 3 статті у фахових наукових виданнях категорії «Б».

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел (71 найменувань) і 7 додатків. Основний зміст викладений на 142 сторінках друкованого тексту, містить 53 рисунки та 20 таблиць. Загальний обсяг дисертації – 188 сторінок.

1 АНАЛІЗ СУЧАСНИХ РІШЕНЬ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ

1.1 Принципи сучасних інформаційних технологій керування безпілотними апаратами

Інформаційні технології активно розвиваються, удосконалюються, впроваджуються та використовуються в усіх сферах життя вже протягом кількох останніх десятиліть.

Серед багатьох визначень терміну «Інформаційні технології» (ІТ) напевно найбільш вдалим та повним є наступне: інформаційна технологія – це система методів, процесів та способів використання обчислювальної техніки і систем зв'язку для створення, збору, передачі, пошуку, оброблення та поширення інформації з метою ефективної організації діяльності людей [1].

Більш розширене визначення цього терміну надано ЮНЕСКО, а саме: інформаційна технологія – «це комплекс взаємопов'язаних наукових, технологічних, інженерних дисциплін, що вивчають методи ефективної організації праці людей, зайнятих обробкою та зберіганням інформації, обчислювальну техніку, методи організації взаємодії з людьми та виробничим обладнанням, їх практичне застосування, а також пов'язані з цим обробленням соціальні, економічні і культурні проблеми» [1].

Використання інформаційних технологій у різних галузях діяльності людей є не тільки виправданим, але й дуже ефективним кроком розвитку суспільства в цілому.

Інформаційні технології широко застосовуються в науці, промисловості, сільському господарстві, управлінні, освіті, медицині, системах зв'язку, банківській сфері, бізнесі, інших галузях народного господарства (рисунок 1.1).

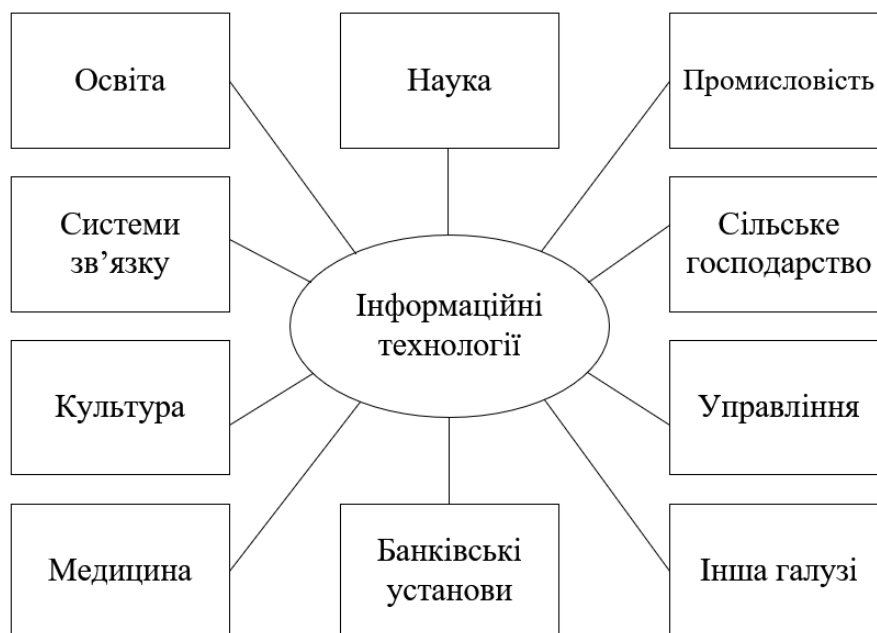


Рисунок 1.1 – Основні напрями застосування інформаційних технологій

Застосування інформаційних технологій дозволяє:

- зекономити ресурси (трудові, фінансові, енергетичні);
- автоматизувати і пришвидшити виробничі (технологічні) процеси;
- замінити некваліфіковану (рутинну) працю на кваліфіковану та творчу;
- підвищити обсяги та швидкість обміну інформацією;
- оперативно та професійно приймати управлінські рішення;
- проводити комп'ютерне моделювання складних систем і процесів в наукових експериментах;
- покращити якість освітніх процесів.

В науці інформаційні технології надають нові можливості досліднику не тільки швидко опрацьовувати великі обсяги інформації, виконати великі та обтяжливі розрахунки, але й отримати результати математичного моделювання складних експериментів, які в реальності провести або неможливо, або навіть небезпечно.

Поява нових викликів сьогодення щодо забезпечення життя і здоров'я людей, які опинилися в екстремально-небезпечних умовах, з одного боку, та постійне удосконалення технічних можливостей безпілотних апаратів (БПА), з іншого боку,

породжують зростаючий попит розробки, подальше впровадження та застосування нових математичних моделей і методів керування БПА.

Збільшення можливостей та вдосконалення функцій сучасних БПА нерозривно пов'язані з рівнем розвитку (станом) інформаційних технологій.

Інформаційні технології керування БПА активно задіяні у таких важливих процесах, а саме: автоматизації керування, обробці даних, забезпеченні безпеки та коректного (правильного) виконання місії.

Розглянемо стан сучасного застосування інформаційних технологій у кожному із перелічених вище процесах.

Автоматизація управління. Один з головних напрямів інформаційних технологій керування БПА. Цей напрям особливо важливий та актуальний для здійснення пошуково-рятувальних операцій, оскільки точність та швидкість виконання місії є найголовнішою метою цих операцій. Зазначимо, що успішність та швидкість виконання окремих завдань місії та місії в цілому збільшується, в той же час потреба в людських ресурсах та безпосередньому втручанні оператора значно зменшується.

Обробка великих масивів інформації. Вхідна інформація щодо навколишнього середовища надходить до БПА з багатьох пристроїв, зокрема: камер та сенсорів, інформація з яких фіксується, аналізується в реальному часі з метою формування рішення знаходження найкращих траєкторій руху при виявленні перешкод.

Інтеграція БПА з зовнішніми системами зв'язку. Дистанційне керування та моніторинг БПА у реальному часі забезпечується їх зв'язком з зовнішніми мережами передачі даних, в тому числі на віддалені сервери. Інформація передана з БПА на зовнішні системи зв'язку підлягає обробці та аналізу.

Системи навігації на основі GPS. Точне позиціонування та навігація БПА забезпечене глобальною системою позиціонування (GPS). Точність керування руху забезпечується використанням GPS та інших інформаційних технологій.

Система запобігання зіткнень. Виявлення перешкод та автоматичне корегування траєкторії руху, яке можливе завдяки використанню сучасних

сенсорів та камер, встановлених на БПА. Ризик непередбачуваних ситуацій та дорожньо-транспортних пригод значно знижується.

Аналіз зображень у реальному часі. БПА можуть обробляти зображення у реальному часі завдяки використанню нейронних мереж та технологій комп'ютерного зору. Визначення координат об'єктів, їх ідентифікація дозволяє приймати відповідні рішення щодо наступних кроків.

Використання бортових датчиків для управління БПЛА, зокрема мікролітальних безпілотних апаратів (МБПЛА) є предметом розгляду статті [2]. Вбудована система для управління БПЛА наведена в роботі [2], може використовуватись навіть при недоступності GPS. Оптимальне управління БПЛА виконується тільки за допомогою бортових обчислювальних пристроїв. З точки зору авторів [2], проведені експерименти з кількома квадрокоптерами продемонстрували надійність та виконання задач навігації в складних умовах (рисунок 1.2). Ця розробка демонструє впровадження сучасних технологій управління та навігації для МБПЛА. Впровадження вбудованої системи для МБПЛА визначає перспективи подальшого розвитку автономних систем керування квадрокоптерами [2]. Однак, деякі проблеми автори залишили не вирішеними, а саме: стабільне позиціонування в одній точці в умовах низького освітлення та працездатність системи в екстремальних умовах.

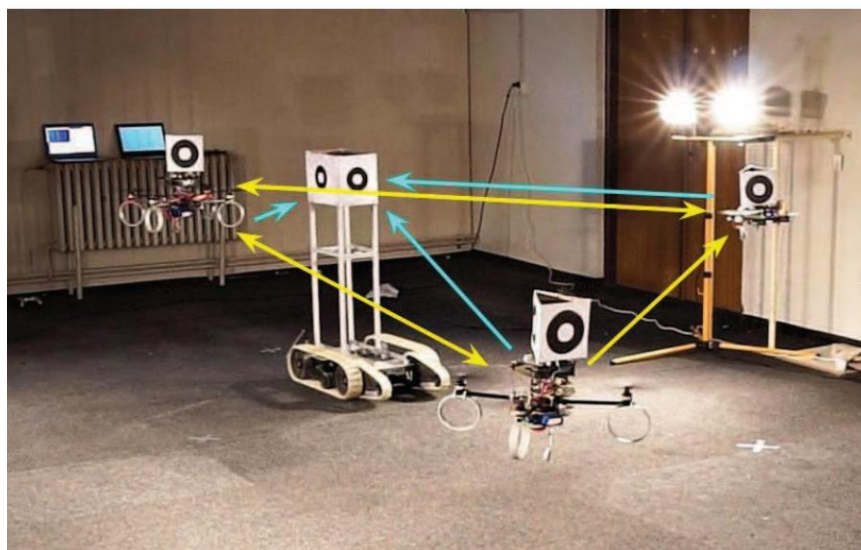


Рисунок 1.2 – Три МБПЛА формують трикутник завдяки розпізнаванню [2]

Історичний огляд безпілотних літальних систем (БПЛС) на які був поширений міжнародний запит надано у статті [3]. БПЛА широко використовуються у спостереженні та передачі інформації. Автор [3] надає огляд підсистем БПЛА, зокрема навігаційної системи, наземної станції, автопілоту, системи планування місії та управління (рисунок 1.3). Приділяється увага сучасним методам проєктування систем керування для БПЛА. Автор [3] розглядає інтелектуальні методи управління, які надають додаткові можливості управління БПЛА. Зростаючі вимоги суспільства потребують подальший розвиток та покращення управління БПЛА. Тенденції розвитку управління БПЛА свідчать про зростання потреби у більш досконаліх та надійних інформаційних технологіях для коректного виконання місії (завдань) в складних умовах.

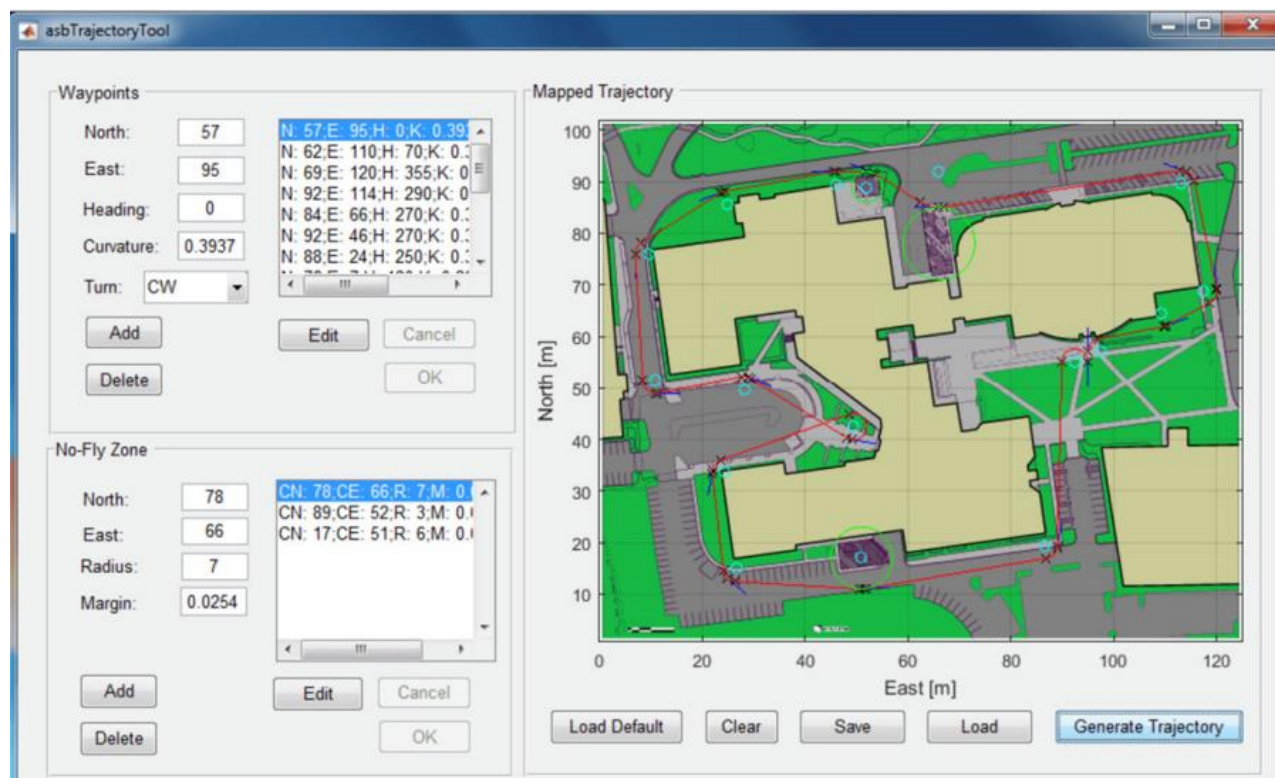


Рисунок 1.3 – Симуляція траєкторії польоту БПЛА [3]

Разом з тим, під час проведення симуляції автор не враховував зовнішні фактори впливу (вітер, погане освітлення). У роботі [3] не розглядалось застосування нейронних мереж у запропонованій системі, хоча їх використання могло покращити фінальний результат дослідження.

Темою розгляду статті [4] є створення єдиної системи управління трафіком безпілотних апаратів. Автори розглядають безпілотні апарати за наступними ознаками: розміром, призначенням та іншими критеріями. Сучасні безпілотні апарати можуть вирішувати цивільні задачі, такі як моніторинг та доставку вантажів. Автори пропонують розглянути впровадження методів інтеграції БПА в цивільний простір, що, на їх думку, дозволить зменшити час на доставку вантажів чи проведення пошуково-рятувальних операцій.

Розробка нелінійної системи управління для безпілотних апаратів за умови недоступності GPS набула актуальності. Тому деякі групи дослідників зайнялись цією тематикою. Система, запропонована у дослідженні [5], може стабілізувати позицію БПЛА за допомогою візуальних об'єктів. Для визначення місця розташування запропоновано використовувати лише датчики, які встановлені на БПЛА. Датчики оцінюють швидкість переміщення у реальному часі. Розрахунки й експерименти в роботі [5] проведені на вбудованому процесорі.

Автоматичне розпізнавання мови (ASR) є одним зі способів керування БПА [6]. Основна мета такої розробки – спрощення взаємодії між людиною-оператором та БПА. Автор [6] звертає увагу на мінімізацію людського втручання при керуванні БПА. Запропонована технологія ASR використовується для виконання спільних місій БПА, наприклад, аграрної обробки полів, перевірці залізничних колій, ліній електропостачання та картографування місцевості (рисунок 1.4). Такий підхід дозволяє одному оператору керувати та спостерігати за місіями кількох БПА у реальному часі.

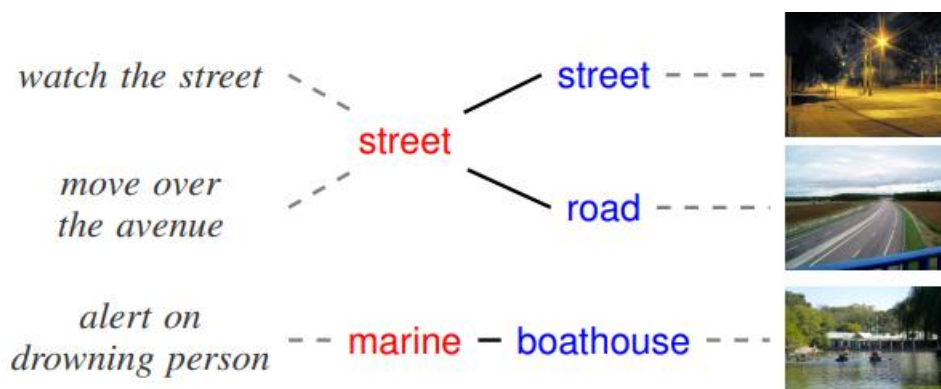


Рисунок 1.4 – Голосове управління та виконання команд для БПА [6]

Як недолік слід зазначити, що керування БПА за допомогою голосових команд має великі обмеження в умовах сторонніх шумів, що негативно впливає на швидкість і точність виконання поставленої задачі.

Безпілотні наземні апарати (БпНА) все частіше використовуються у сільському господарстві. Автори статті [7] пропонують застосування датчиків різних типів для розробки адаптивної навігаційної системи без GPS.

Разом з тим, привертає увагу, що додаткове використання GPS могло б підвищити точність у складних умовах. Крім того, дана система досить залежна від наявного освітлення та поточних погодних умов (рисунок 1.5). Розроблений БпНА використовує такі складні алгоритми, застосування яких може підвищити час обробки інформації.



Рисунок 1.5 – Визначення безпечних та небезпечних зон пересування для БпНА за допомогою вбудованих сенсорів [7]

Використання нечіткої логіки для керування БпЛА розглянуто у роботі [8]. Нечітка логіка дозволяє виконувати точне слідування за заданою траєкторією руху. За інформацією, наданої у дослідженні [8], адаптація до різних умов польоту забезпечує загальну надійність та безпечне проведення місій; проведені

експерименти демонструють ефективність використання нечіткої логіки для керування БпЛА (рисунок 1.6).

Але проблема використання нечітких контролерів в тому, що таку систему неможливо удосконалити за рахунок нейронних мереж, а покращення наявних результатів є несуттєвими [8].

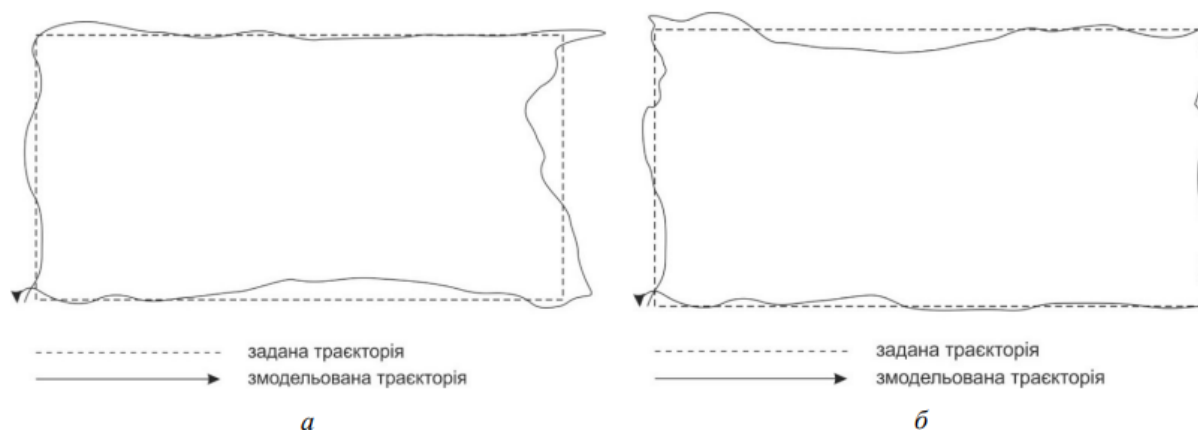


Рисунок 1.6 – Порівняння траєкторії польоту між звичайним контролером (а) та нечітким контролером (б) [8]

1.2 Огляд основних понять ігрових методів

Всі ігрові методи засновані на теорії ігор.

Теорія ігор – розділ прикладної математики, за допомогою якого вчені моделюють поведінку декількох суб'єктів, коли критерій прийняття рішення кожного залежить від рішень, прийнятих іншими [9].

Головними поняттями теорії ігор традиційно вважають:

- суб'єкти, які приймають рішення – гравці (агенти);
- стратегії – можливі послідовності дій, які гравці мають право вибрати;
- винагороди – це виграші, які одержать учасники гри за умови стратегії, яку вони обрали;
- рівновага Неша – стан, коли жоден гравець не може покращити свій виграш, змінивши свою стратегію в односторонньому порядку, якщо інші гравці не змінюють своїх стратегій.

Ігрові методи застосовуються для моделювання поведінки складних систем і відкривають можливості знаходження оптимального рішення, у тому числі, стосовно різних задач з керування БПА.

Використовуючи апарат теорії ігор можна моделювати випадки (ситуації) з кількома гравцями (безпілотниками та/або операторами), які взаємодіють між собою або з природою (навколишнім середовищем) для отримання максимального виграшу.

Розглянемо ряд ігрових методів, які застосовуються для практичного моделювання поведінки складних систем: кооперативні ігрові методи, некооперативні ігрові методи та динамічні ігрові методи.

1.3 Кооперативні ігрові методи та їх застосування у моделюванні складних систем

Кооперативні (коаліційні) ігрові методи розглядають виграш групи гравців, а не кожного окремого гравця.

Кооперативний ігровий метод визначає наскільки ефективно кожна група гравців може працювати разом. Дії кожного індивідуального гравця не є предметом розгляду, важливим є лише спільний виграш між гравцями. Основана увага у цих ігрових методах приділяється розподілу виграшу між гравцями коаліції [9].

Кооперативний ігровий метод визначається як пара $\langle I, v \rangle$, де коаліція $I = \{1, 2, \dots, N\}$ – це скінченна множина гравців, а v – характеристична функція, яка діє на підмножинах I .

Винагорода кожного гравця представлена у вигляді вектора винагород, який визначає, як розподіляється загальний результат гри між гравцями. Під множиною всіх можливих розподілів мається на увазі повний набір таких розподілів для кожного гравця [9].

$$A = \left\{ x = (x_1, \dots, x_N) \in E^n : x_i \geq v(i), \sum_{i=1}^n x_i = v(I) \right\}, \quad (1.1)$$

де A – множина розподілів виграшу між гравцями у кооперативному ігровому методі;

$x = (x_1, \dots, x_N)$ – вектор розподілу винагороди між гравцями. Вектор x складається з компонентів, кожен з яких відповідає частині винагороди для окремого гравця в кооперативному ігровому методі;

E^n – n -вимірний простір, де вектор винагород x , належить цьому простору;

$x_i \geq v(i)$ – це обмеження, яке вказує, що частка винагороди x_i для кожного i -го гравця повинна бути не меншою за мінімальний внесок гравця $v(i)$;

$v(i)$ – індивідуальний внесок або внесок кожного i -го гравця у грі, який визначає мінімальну винагороду, яку гравець має отримати в результаті співпраці;

$\sum_{i=1}^n x_i = v(I)$ – рівність демонструє, що сумарний розподіл винагород між усіма гравцями повинен дорівнювати загальному виграшу коаліції I (усієї групи гравців), позначеному як $v(I)$. Це гарантує, що весь виграш правильно розподіляється між учасниками коаліції; $i = \overline{1, N}$.

Кооперативні ігрові методи бувають двох видів: з передаваною (трансферабельною) корисністю та з непередаваною (нетрасферабельною) корисністю.

Кооперативні ігрові методи з передаваною корисністю – корисність вільно перерозподіляється між гравцями; корисність вимірюється в універсальних одиницях і є загальноприйнятною для всіх гравців [9].

Наведемо класичне означення характеристичної функції.

Характеристична функція v – це функція, яка кожній коаліції $K \subseteq N$ призначає її винагороду. Вона відображає множину всіх можливих підмножин множини гравців у множину дійсних чисел [9]:

$$v: 2^N \rightarrow R \quad (1.2)$$

Характеристична функція визначає, яку загальну винагороду може отримати кожна коаліція гравців $K \subseteq N$ і використовується для розрахунку виграшів у кооперативних ігрових методах в залежності від їхнього об'єднання.

Кооперативна гра вважається несуттєвою, якщо виграш $v(K)$ будь-якої коаліції дорівнює сумі виграшів її окремих гравців. Тоді об'єднання гравців у коаліцію K не дає додаткової винагороди або винагороди порівняно з індивідуальними стратегіями кожного i -го гравця [9].

$$v(K) = \sum_{i \in K} v(i), \quad (1.3)$$

де $v(K)$ – сумарний внесок всіх гравців коаліції K .

Побічний платіж – різниця між загальною винагородою коаліції в цілому і винагородою для кожного i -го гравця.

Окремо розглядаються кооперативні ігрові методи з непередаваною корисністю. Такі методи передбачають, що корисність, де винагорода всієї коаліції K та винагорода окремого i -го гравця не підлягають порівнянню. Такий ігровий метод не має побічних платежів [9].

Основні властивості характеристичної функції наступні: монотонність, суперадитивність, опуклість, індивідуальна та групова раціональності.

1. Монотонність:

Нехай A та B – дві різні коаліції гравців; кількість гравців коаліції A не перевищує кількість гравців коаліції B .

$$\text{Тоді } A \subseteq B \Rightarrow v(A) \leq v(B),$$

де $v(A)$ – виграш коаліції гравців A ;

$v(B)$ – виграш коаліції гравців B .

2. Суперадитивність:

Нехай A та B – дві різні коаліції гравців, які не мають спільних гравців.

$$\text{Тоді } A \cap B = \emptyset \Rightarrow v(A \cup B) \geq v(A) + v(B),$$

де $v(A)$ – виграш коаліції гравців A ;

$v(B)$ – виграш коаліції гравців B .

3. Опуклість:

Нехай A та B – дві різні коаліції гравців, які можуть мати спільних гравців:

Тоді $A \cap B \neq \emptyset \Rightarrow v(A \cup B) + v(A \cap B) \geq v(A) + v(B)$,

де $v(A)$ – виграш коаліції гравців A ;

$v(B)$ – виграш коаліції гравців B ;

$v(A \cup B)$ – виграш об'єднання двох коаліцій гравців A та B ;

$v(A \cap B)$ – виграш гравців, які одночасно входять до коаліції гравців A та B .

4. Умова групової раціональності:

Нехай N – множина гравців.

$x(N)$ – сумарний виграш N гравців, якщо вони грають поодиночі;

S – коаліція гравців.

$v(S)$ – загальний виграш гравців, які входять в коаліцію S (сумарний виграш гравців коаліції S).

Тоді справедливо:

$$x(N) = v(S) \quad (1.4)$$

5. Умова індивідуальної раціональності:

Нехай x_i – виграш i -го гравця, якщо він входить в коаліцію;

$v(\{i\})$ – виграш i -го гравця, який грає самостійно (не входить до коаліції)

Тоді:

$$x_i \geq v(\{i\}) \quad (1.5)$$

1.4 Некооперативні ігрові методи

Теорія некооперативних ігор застосовується для моделювання й аналізу ситуацій, в яких оптимальні рішення кожного учасника (гравця) залежать від його припущень (очікувань) про гру опонентів. Кожний гравець повинен, в даному випадку, намагатися передбачити гру опонентів, використовуючи свої знання правил гри і виходячи із припущень, що опоненти теж раціональні і самі намагаються передбачити кроки своїх опонентів і збільшити свої власні виграші [10].

Некооперативні ігри – це тип моделей у теорії ігор, де передбачається, що кожен гравець приймає рішення самостійно та співпраця між гравцями не передбачена. Будь-які угоди між гравцями, у тому числі створення коаліцій також не передбачаються.

Некооперативна гра має такі основні елементи:

- перелік гравців;
- множина можливих стратегій кожного гравця;
- виграші при кожному дозволеному набору стратегій.

Незважаючи на те, що гравець самостійно формує свою стратегію, його виграш визначається не тільки власною стратегією, але й стратегіями інших учасників (гравців) гри.

1.5 Динамічні ігрові методи

Динамічна гра – гра n осіб у вигляді процесу, що розвивається у часі, протягом якого гравці приймають послідовно часткові рішення, переходячи від одного стану гри до іншого [11].

Динамічна гра передбачає:

- порядок дій гравців;
- потенційні сценарії розвитку гри, які з'являються в процесі проведення гри;
- залежність обсягів виграшів гравців від подій, які відбулися в грі;
- обсяг інформації, яка відома кожному гравцю стосовно дій інших гравців.

Основні етапи динамічної гри:

1. Визначення множини гравців;
2. Моделювання випадкових подій шляхом появи ще одного гравця, а саме «природи»;
3. Визначення порядку ходів гри;
4. Визначення доступності дій кожного гравця на кожному етапі гри.

Розглянемо приклад динамічної гри з використанням безпілотних апаратів (БпА), де перший хід робить оператор БпА. Оператор повинен вибрати стратегію для керування БпА у складній ситуації (рисунок 1.7).

1. Перший крок робить оператор. Він обирає один з двох можливих варіантів: «запустити БпА» або «не запустити БпА»;

2. При виборі варіанту «не запустити БпА», гра завершується. БпА залишається на землі;

3. Якщо в першому кроці оператор обирає варіант «запустити БпА», тоді настає другий крок;

4. Другий крок: БпА потенційно може зіткнутися з перешкодою;

5. Виникають два варіанти подальшого розвитку подій: «БпА обходить перешкоду» або «відбувається зіткнення з перешкодою»;

6. Якщо БпА обійшов перешкоду, то гра завершується успішно;

7. Якщо БпА зіткнувся з перешкодою, тоді оператор намагається налагодити подальший рух БпА. В такому разі можливі два варіанти розвитку подій: позитивний та негативний;

8. У разі позитивного розвитку подій, БпА успішно продовжив подальший рух. Гру завершено успішно;

9. У разі негативного розвитку подій, подальший рух неможливий, БпА знищено та гру завершено з від'ємним виграшем.

Цей процес можна представити у вигляді дерева рішень, де кожна вершина показує момент, коли один із гравців робить свій вибір, а результат залежить від обраних дій обох гравців. Стратегія кожного гравця полягає у виборі дій на кожному кроці, коли настає його черга ухвалювати рішення.

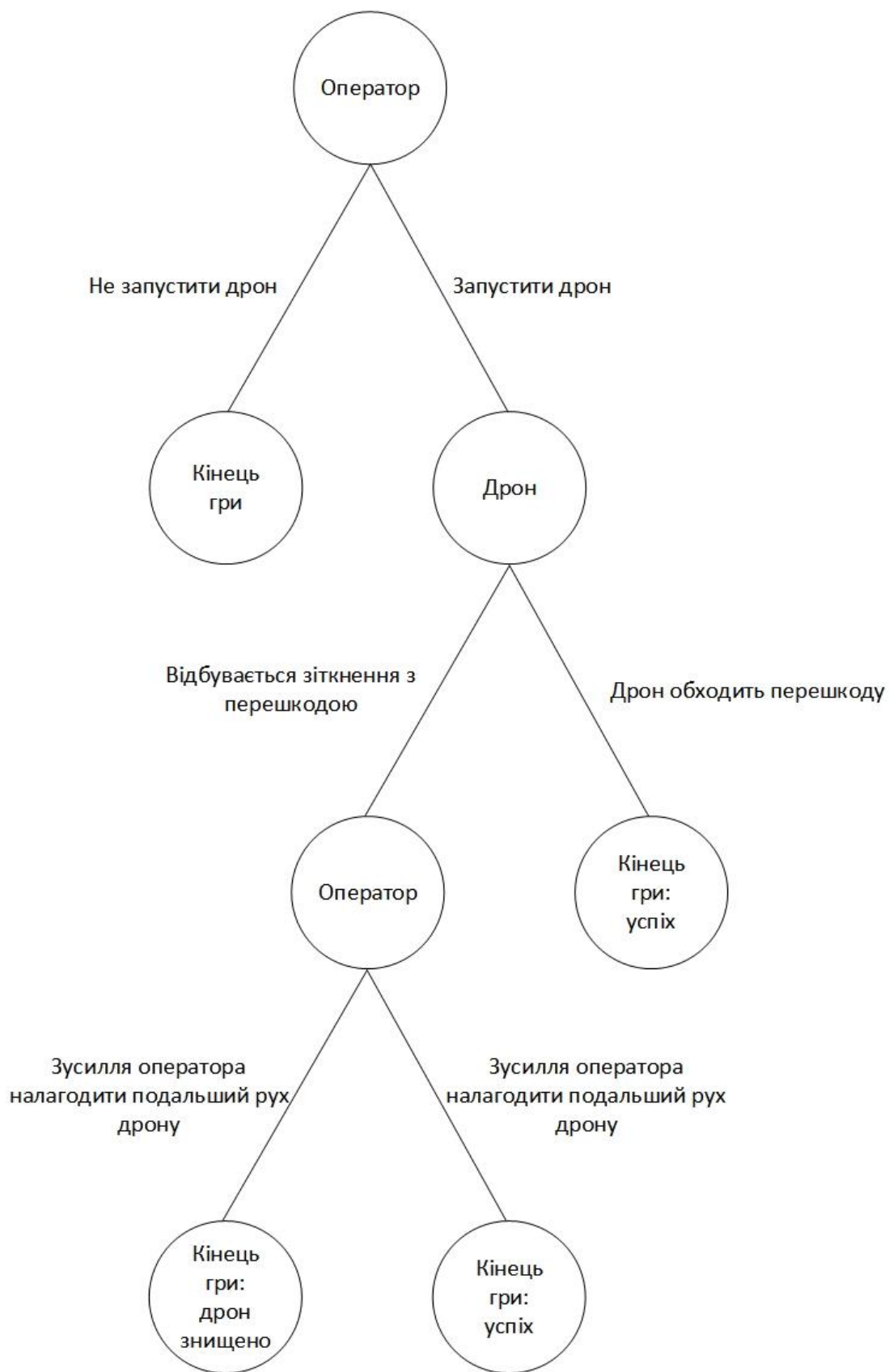


Рисунок 1.7 – Дерево гри

1.6 Нейронні мережі

Штучні нейронні мережі (ШНМ) – це обчислювальні системи, які побудовані за принципами функціонування біологічних нейронних мереж.

ШНМ відносять до класу алгоритмів штучного інтелекту, які виникли у 1980-х роках в результаті досліджень когнітивних та комп'ютерних наук [12].

ШНМ розроблені для алгоритмічного моделювання процесу прийняття рішень людиною, використовуючи приховані структури даних [13].

Структура штучної нейронної мережі (рисунок 1.8) складається з:

- вхідних нейронів;
- прихованих нейронів;
- вихідних нейронів.

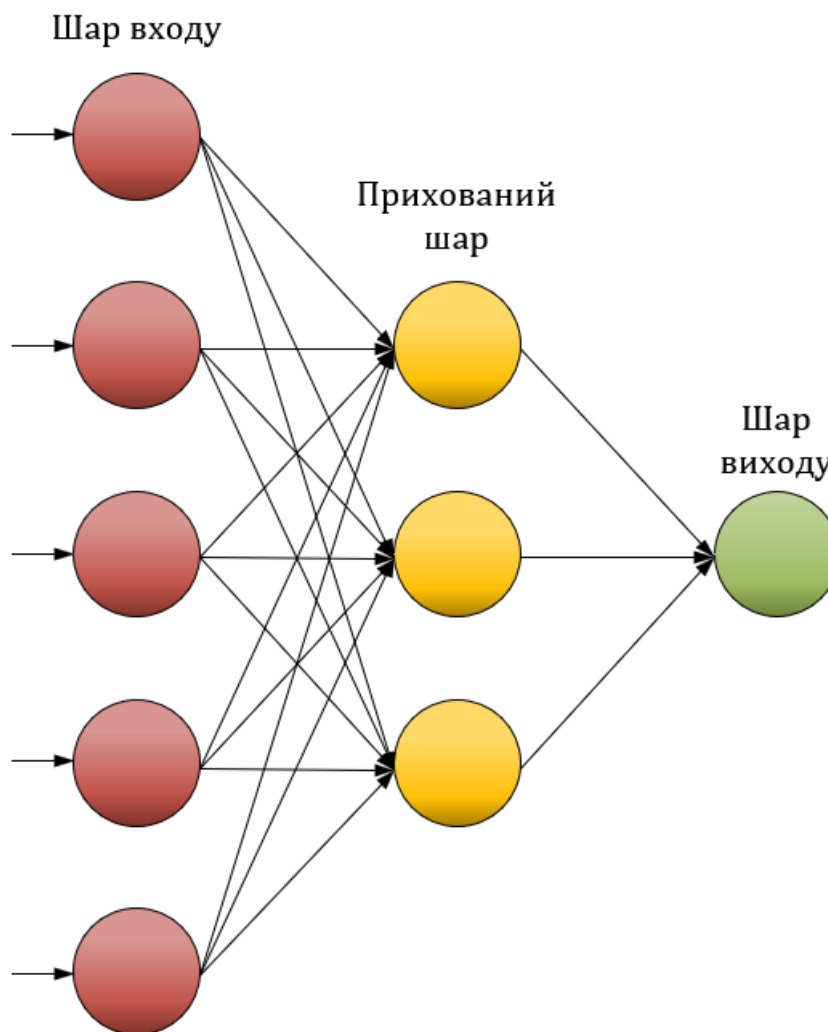


Рисунок 1.8 – Приклад структури штучної нейронної мережі

Вхідні нейрони приймають інформацію із зовнішнього середовища (вхідні сигнали) і передають до прихованого шару. На даному шарі обробка інформації в тому числі обчислення та прийняття рішень не відбуваються.

Приховані нейрони опрацьовують та перероблюють отриману інформацію. Вхідні дані обробляються, змінюються, зокрема, шляхом виконання необхідних обчислень.

Вихідні нейрони відповідають за отримання висновків та остаточного результату.

Структура і складність нейронних мереж надзвичайно різноманітна і визначається в залежності від вирішення проблеми, яка виникає перед дослідником. Нейронні мережі вже зараз застосовуються і мають перспективи більш широкого впровадження в різні аспекти діяльності суспільства.

Штучні нейронні мережі складаються зі штучних нейронів.

Штучний нейрон – це вузол штучної нейронної мережі, нелінійна функція від спільного аргументу (лінійної комбінації вхідних сигналів).

В процесі розвитку ШНМ породило багато різноманітних методик, які знайшли широке використання у багатьох галузях.

Нейронна мережа прямого поширення

Нейронна мережа прямого поширення – це тип нейронної мережі, в якій дані передаються з одного шару в наступний в одному напрямку. Така мережа не має зворотних зв'язків. Перцептрон Розенблата є основою нейронної мережі прямого поширення.

Приклад нейронної мережі прямого поширення наведено на рисунку 1.8.

Основна перевага нейронної мережі прямого поширення полягає в тому, що вона проста та ефективна. Її також відносно легко навчити, що робить її хорошим вибором для багатьох завдань. Однак нейронні мережі прямого зв'язку не так добре обробляють послідовні дані, як рекурентні нейронні мережі [14].

Рекурентна нейронна мережа

Рекурентна нейронна мережа – це тип нейронної мережі, яка має цикли зворотного зв'язку. Це означає, що інформація може переміщатися у двох

напрямок між шарами мережі. Цикли дозволяють рекурентним нейронним мережам обробляти послідовні дані, такі як розпізнавання мови та машинний переклад. Приклад рекурентної нейронної мережі наведено на рисунку 1.9.

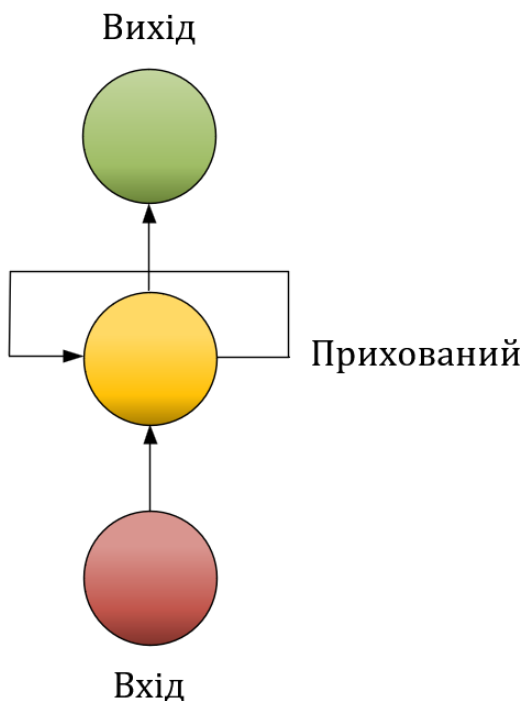


Рисунок 1.9 – Приклад рекурентної нейронної мережі

Архітектура рекурентної нейронної мережі є набором шарів, які схожі до нейронної мережі прямого зв'язку. Однак шари рекурентної нейронної мережі з'єднані в цикл, так що вихідні дані кожного шару можуть повертатися на вхід того самого шару. Це дозволяє мережі вивчати часові зв'язки між точками даних.

Проста рекурентна нейронна мережа має три шари: вхідний, повторюваний прихований і вихідний, як показано на рисунку 1.9. Вхідний рівень має N вхідних вузлів. Вхідними даними цього рівня є послідовність векторів за час t , наприклад $\{\dots, x_{t-1}, x_t, x_{t+1}, \dots\}$, де $x_t = (x_1, x_2, \dots, x_N)$. Вхідні вузли в повністю з'єднаній рекурентної нейронної мережі поєднані з прихованими вузлами в прихованому шарі. З'єднання визначаються ваговою матрицею W . Прихований рівень M має $h_t = (h_1, h_2, \dots, h_M)$ прихованих вузлів, які з'єднані один з одним у часі повторюваними зв'язками [15].

Ймовірнісна нейронна мережа

Ймовірнісна нейронна мережа (ЙНМ) – це тип нейронної мережі прямого поширення, яка широко використовується в задачах класифікації та розпізнавання образів [16].

Архітектура ЙНМ складається з чотирьох шарів (рисунк 1.10):

- вхідний шар;
- шар патерну;
- шар підсумку;
- вихідний шар.

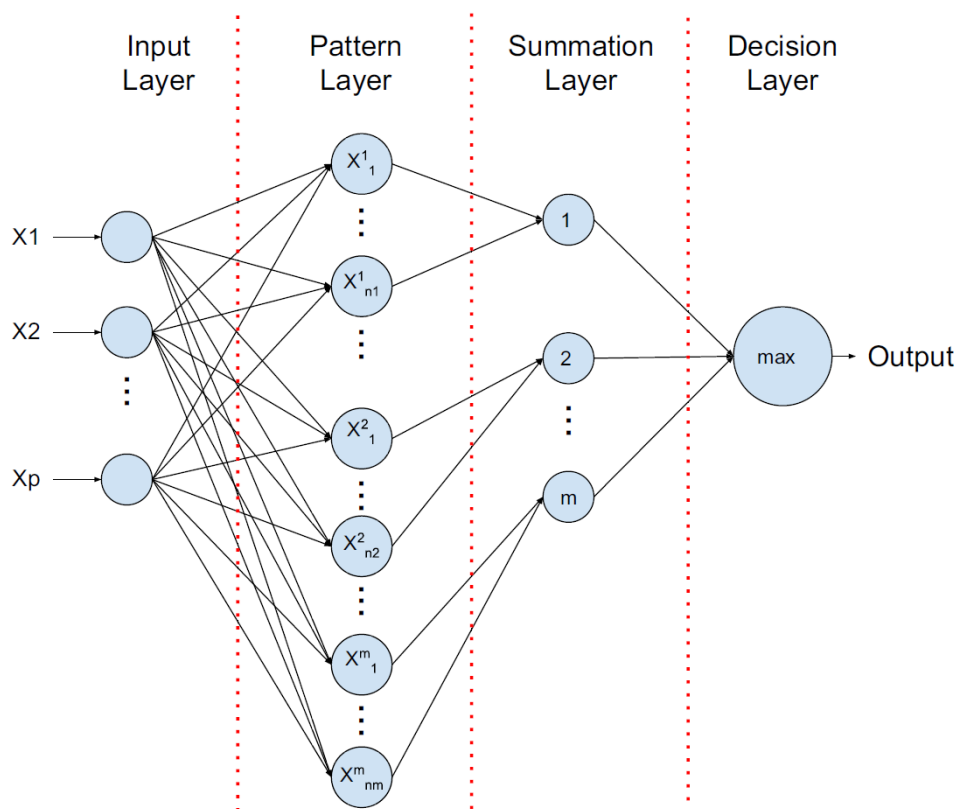


Рисунок 1.10 – Архітектура ймовірнісної нейронної мережі [16]

Вхідний шар приймає початкові дані, які необхідно класифікувати. Шар патерну обчислює ймовірності для кожного зразку даних, використовуючи ймовірнісну функцію розподілу. Шар підсумку підсумовує ймовірності для кожного класу. Вихідний шар визначає клас з найбільшим показником ймовірності та надає фінальний результат класифікації.

Згорткова нейронна мережа

Згорткова нейронна мережа (ЗНМ) – це тип нейронної мережі прямого поширення, який застосовується у різноманітних задачах. ЗНМ часто використовується для класифікації зображень, виявлення об'єктів, обробки відеоматеріалу, розпізнавання мови жестів та розпізнавання голосових команд [17].

Архітектура ЗНМ складається з чотирьох шарів (рисунок 1.11):

- шар згортки;
- шар об'єднання;
- повністю з'єднаний шар;
- нелінійний шар.

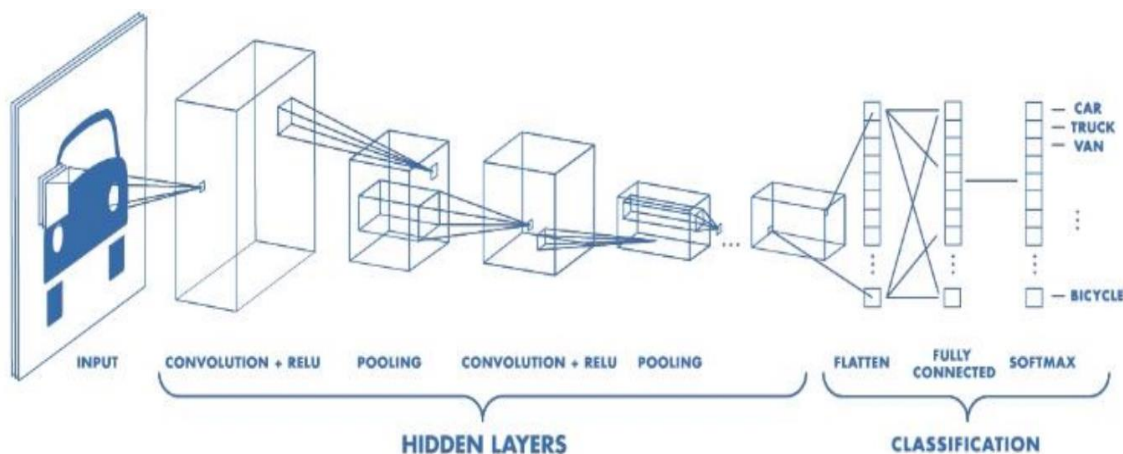


Рисунок 1.11 – Архітектура згорткової нейронної мережі [17]

Перед тим як пояснити кожен шар архітектури ЗНМ, розглянемо додаткові поняття, які необхідні для розуміння роботи ЗНМ.

Ядро – це матриця чисел, яка використовується для обчислення згортки на вхідному зображенні (чи іншого об'єкту).

Згортка – це математична операція, яка обчислює нові значення для кожної області зображення (чи іншого об'єкту), створюючи нову матрицю. Ця матриця демонструє вихідну карту ознак, які відносяться до зображення (чи іншого об'єкту).

Шар згортки використовує фільтри ядра для обчислення згортки вхідного зображення шляхом вилучення основних ознак.

Шар об'єднання поєднує два послідовних шарів згортки.

Повністю з'єднаний шар схожий на нейронну мережу прямого зв'язку, як показано на рисунку 1.8. Цей рівень знаходиться на нижньому шарі мережі. Він отримує вхідні дані від шару об'єднання або шару згортки, зведений перед відправленням на наступний рівень.

Нелінійний шар є фінальним компонентом ЗНМ. Він забезпечує здатність нейронної мережі виявляти складні закономірності вхідних даних. Нелінійна функція визначає вихідні дані нейронної мережі, наприклад «так» або «ні». Найбільш поширеними та популярними нелінійними функціями ЗНМ є Sigmoid, Tanh, ReLU, Leaky ReLU, Noisy ReLU та Parametric Linear Units [17]. Організація та функція зорової кори значною мірою впливають на архітектуру ЗНМ, оскільки вона розроблена так, щоб нагадувати нейронні зв'язки в мозку людини. Серед популярних архітектур ЗНМ є LeNet, AlexNet і VGGNet [17].

Динамічні нейронні мережі

Динамічна нейронна мережа (ДНМ) – це тип нейронної мережі, яка постійно змінює свою структуру або функції в залежності від вхідних параметрів або зовнішніх умов [18].

Динамічні нейронні мережі можна розділити на три основні групи, на основі яких проводяться обчислення на вхідних вибірках (рисунк 1.12):

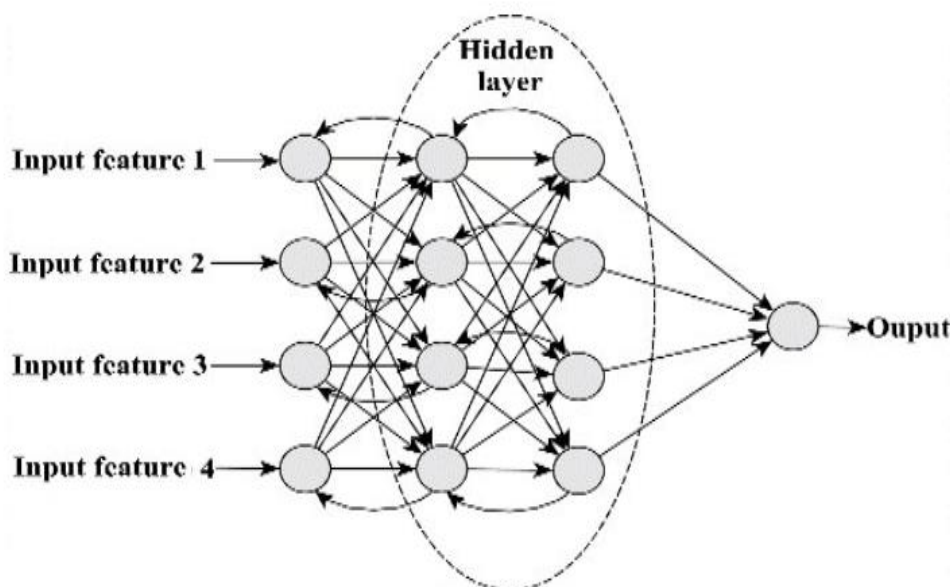


Рисунок 1.12 – Архітектура динамічної нейронної мережі [18]

Вибіркові динамічні нейронні мережі належать до моделей глибокого навчання, які динамічно розподіляють обчислювальні ресурси на основі кожного окремого входу. Зокрема, ці мережі обробляють кожен зразок як єдине ціле і не вникають у внутрішню структуру даних окремих зразків.

Просторові динамічні нейронні мережі – це моделі, які враховують просторову структуру в межах вхідних зразків, переважно застосованих до візуальних даних, таких як зображення. Звичайні алгоритми глибокого навчання рівномірно розподіляють обчислення між просторовими областями, що призводить до надлишкової обробки в багатьох завданнях зору, таких як виявлення об'єктів, де область інтересу становить лише невелику частку вхідних даних. Завдяки адаптивному зосередженню на інформаційних регіонах, які є найбільш відповідними для завдання, просторово-динамічні мережі можуть значно підвищити обчислювальну ефективність [18].

Тимчасові динамічні нейронні мережі можуть нерівномірно розподіляти обчислення за часовим виміром для послідовних даних, таких як відео. У разі поточних даних, таких як відео, зазвичай існує висока кореляція між найближчими кадрами. Отже, динамічний фокус на конкретних ключових кадрах є вирішальною характеристикою для моделей глибокого навчання для зменшення надлишкових обчислень. Крім того, часові та просторові адаптивні обчислення можуть бути реалізовані одночасно для досягнення вищої ефективності [18].

Практичне застосування

Нейронні мережі набули великого значення для розв'язку складних багатокритеріальних задач у найрізноманітніших областях, а саме: у вирішенні технічних проблем, навчанні та науці, медицині, управлінні виробничими (технологічними) процесами, державному управлінні.

Проектування, впровадження та використання нейронних мереж вирішують вагомі технічні проблеми. Зокрема, вибір найкращого маршруту руху транспорту та оптимізація транспортних потоків [19]; моделювання та аналіз складних технічних систем, функціонування яких визначається великою кількістю

різноманітних параметрів [20]; автоматизація технологічних процесів у промисловості; моделювання складних фізичних процесів.

В галузі освіти нейронні мережі активно використовуються, оскільки забезпечують раціональне використання людських ресурсів та надають можливість трансформувати педагогічний процес для проведення індивідуального навчання. Зокрема, визначення оцінки якості освітніх програм у вищих навчальних закладах з урахуванням багатьох факторів [21–22].

Нейронні мережі є ефективним сучасним інструментом в галузі наукових досліджень. Завдяки їх застосуванню підвищується точність наукових результатів, заощаджуються людські та фінансові ресурси [23–24].

Нейронні мережі активно використовуються у медицині для підвищення точності діагностування хвороб, прогнозування спалахів захворювань, покращення якості медичних зображень, впровадження роботизованої хірургії [25–26].

Головною метою застосування нейронних мереж в управлінні виробничими (технологічними) процесами є отримання кінцевого продукту високої якості за умови високої ефективності всіх виробничих операцій та оптимізації витрат [27–28].

Підводячи підсумок, наведеної вище інформації щодо застосування нейронних мереж, робимо висновок, що повний огляд потенційних можливостей сучасних нейронних мереж наразі є неможливим, оскільки регулярно з'являються нові розробки в цій області наукових досліджень, які безумовно кращі за попередні. Існуючі нейронні мережі постійно удосконалюються, а ті що з'являються вперше мають набагато більше можливостей ніж ті, що проєктувалися раніше.

1.7 Аналіз поточного стану інформаційних технологій керування безпілотними апаратами на базі ігрового підходу та нейронних мереж

Перейдемо до аналізу поточного стану інформаційних технологій керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Поєднання ігрового підходу з можливостями нейронних мереж запропоновано використовувати у задачах перехоплення роїв БпЛА [29].

Перед авторами [29] постає виконання двох задач, а саме: визначення цілей для БпЛА, які займаються перехопленням, та БпЛА, які ухиляються від переслідування (рисунок 1.13).

Для зменшення часу обчислення та збільшення швидкості переслідування та перехоплення автори пропонують застосування нейронних мереж. Вся система обчислення базується на пошуку рівноваги Неша [9].

За результатами проведених експериментів виявлено такі обмеження у застосуванні: при збільшенні розмірності рою суттєво зростає складність обчислень, під час перехоплення не розглянуто варіант зі зміною поведінки рою, який захищається. Застосування нейронних мереж має суттєве значення лише для засобів ухилення, а не для всього комплексу управління в цілому.

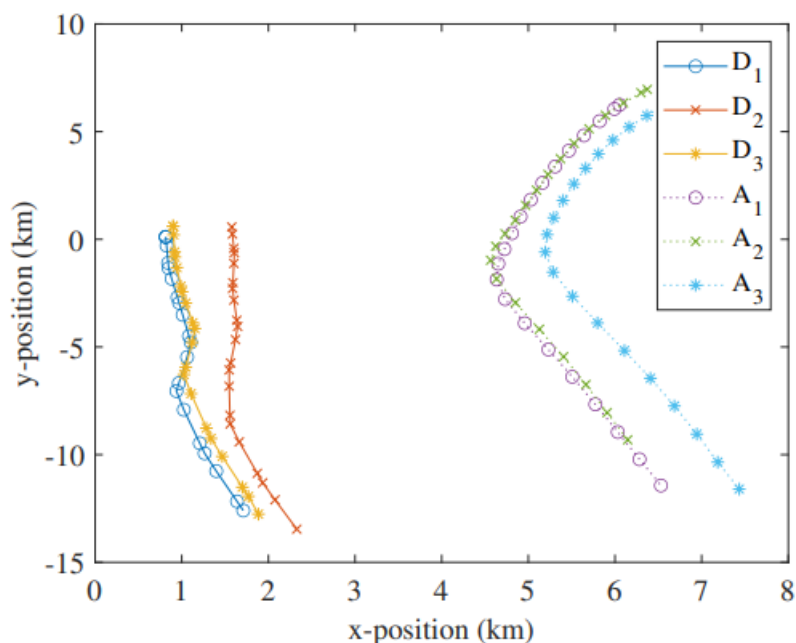


Рисунок 1.13 – Рій БпЛА (зліва) захищається від атакуючого рою (справа) [29]

Нейронні мережі раціонально використовувати для керування безпілотними апаратами.

Розробку використання нейронних мереж для керування безпілотними апаратами представлено у дослідженні [30].

Автори [30] навчили нейронну мережу адаптуватися до змін умов польоту БПЛА у реальному часі (рисунок 1.14).

Для проведення експериментів запропоновано використовувати зворотний зв'язок від сенсорів БПЛА. Представлений алгоритм роботи системи керування дозволяє знизити ризик зіткнення завдяки нейронній мережі [30].

Разом з тим, в результаті проведення експериментів виявлено такі проблеми керування БПЛА: автори [30] виконували адаптацію виключно швидкості пересування БПЛА з врахуванням наявності перешкод; розглянуто тільки горизонтальний рух БПЛА, а вертикальний – залишився поза предметом дослідження; для функціонування системи керування БПЛА використано тільки один сенсор, що може негативно впливати на загальні показники.



Рисунок 1.14 – Керування швидкістю БПЛА за допомогою нейронних мереж [30]

Використання глибокого навчання з підкріпленням дозволяє керувати траєкторіями руху БПЛА.

За твердженнями авторів [31] розроблена модель досить точно відстежувала траєкторію руху та зменшила споживання енергії на відміну від застосування стандартних контролерів.

Але представлений у дослідженні [31] БПЛА не використовував додаткові датчики, які б могли збільшити точність керування. Додатково слід зазначити, що

не були проведені додаткові розрахунки енерговитрат при суттєвих відхиленнях від траєкторії руху.

Особливе значення має навчання нейронної мережі для розпізнавання об'єктів на зображенні у реальному часі.

Для цього автори статті [32] анують велику кількість зображень. На рисунку 1.15 наведено процес анування зображення для тестування та тренування нейронної мережі для розпізнавання дерев у лісовій місцевості, що є важливим елементом забезпечення безпечності польоту.

Окремо використано глибоке навчання нейронної мережі для імітації поведінки автопілота [32].

Проаналізувавши дослідження [32] можна виділити наступні недоліки: нейронні мережі продемонстрували нижчу точність розпізнавання при зміні умов освітлення; інтеграція всіх представлених технологій у єдину систему керування БПЛА відсутня.



Рисунок 1.15 – Зображення для тренування нейронної мережі [32]

Планування оптимальної траєкторії для великої кількості БПЛА можна розробити за допомогою ігрових методів та машинного навчання (рисунок 1.16).

Проведені дослідження демонструють ігровий метод, який за висновками авторів [33], дозволяє знизити енергоспоживання, уникнути зіткнень між БПЛА та адаптуватися до змін вітру.

Запропонований авторами [33] ігровий метод середнього поля ефективно працює з великою кількістю БПЛА, але для обмеженої кількості БПЛА внесок кожного суттєво зростає, що в свою чергу має негативний вплив на точність розрахунків.

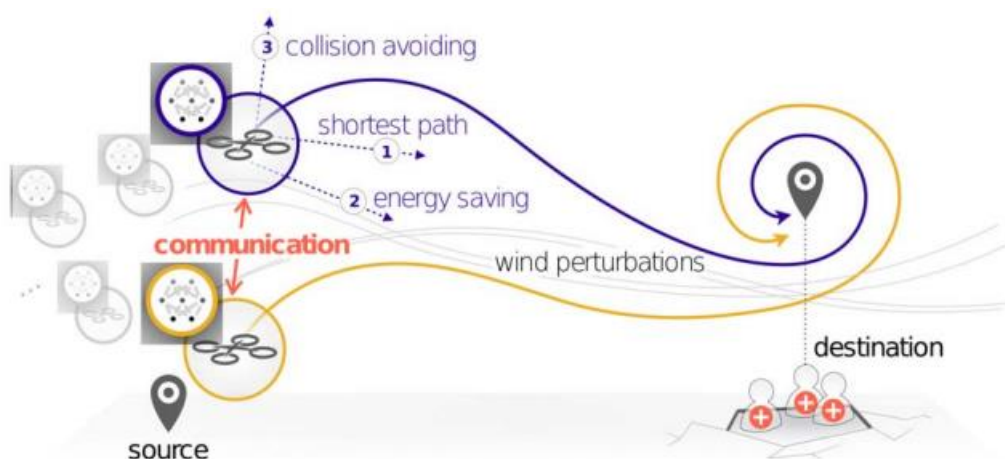


Рисунок 1.16 – Пересування великої кількості БПЛА з початкової точки до кінцевої [33]

Застосування кооперативного ігрового методу є важливою задачею для оптимізації індивідуальних і командних витрат у багатоагентних системах.

Напівдецентралізована стратегія управління запропонована у науковому дослідженні [34].

Така стратегія дозволяє врахувати локальну та індивідуальну інформацію про гравців (агентів). Використання кооперативного ігрового методу дозволяє зменшити загальні витрати команди.

В якості недоліків, слід окремо зазначити, що метод, представлений у статті [34], ефективний для невеликих команд, але обчислювальна складність зростає при збільшенні кількості гравців у команді. Крім того, запропонована система не враховує зміни у структурі взаємодії між гравцями, а також ігнорує зміни у зовнішньому середовищі.

1.8 Висновки до розділу 1

В першому розділі дослідження висвітлено стислий огляд сучасного стану розвитку інформаційних технологій, їх всебічне застосування в різних галузях: виробництві, сільському господарстві, науці, освіті, зв'язку та управлінні.

Особливу увагу звернено на актуальність інформаційних технологій, їх значимість у сучасному світі.

Наведено огляд інформаційних технологій, новітніх наукових досліджень, напрямів, їх практичної реалізації на даний час.

Виконано аналіз позитивних та проблемних (негативних) аспектів практичної реалізації розглянутих наукових досягнень.

Проведено аналіз актуального стану інформаційних технологій керування безпілотними апаратами із застосуванням ігрового підходу та нейронних мереж.

По-перше, надано огляд сучасних прикладних ігрових методів, побудованих на базі апарату теорії ігор, з використанням яких вже отримана низка перспективних результатів та проаналізовані останні наукові та практичні дослідження в цій галузі.

По-друге, розглянуто різні типи нейронних мереж, їх застосування у різноманітних сферах життя. Також розглянуто використання нейронних мереж у поєднанні з навчанням з підкріпленням, що дозволяє приймати оптимальні рішення керування у системі.

Використання нейронних мереж у поєднанні з ігровими методами має перспективу розвитку для підвищення безпеки виконання місії безпілотними апаратами.

Запит суспільства для вирішення складних задач із застосуванням безпілотних апаратів визначає попит на подальший розвиток та вдосконалення інформаційних технологій, у тому числі при розробці нових підходів, методів та моделей керування безпілотними апаратами.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ

2.1 Обґрунтування використання кооперативного ігрового методу у керуванні безпілотними апаратами

Для проведення наукового дослідження процесів керування безпілотними апаратами (БПА) обрано ігровий метод, який найкраще підходить для виконання задачі оптимізації та координації дій.

З усіх існуючих ігрових методів в процесі дисертаційного дослідження обрано кооперативний ігровий метод, оскільки цей метод дозволяє вирішити проблему взаємодії між декількома БПА (гравцями), зокрема, з урахуванням:

- площі покриття;
- швидкості пересування;
- корисного навантаження в умовах обмеженості людських та фінансових ресурсів й енергоспоживання;
- необхідності планування оптимального маршруту руху кількох БПА в складних умовах (виявлення великих статичних перешкод із застосуванням нейронних мереж в умовах ризику та невизначеності, а також появи інших можливих перешкод);
- оптимізації прокладання маршруту в процесі вибору зарядних станцій для БПА.

Результати досліджень викладено у статтях [35–38].

В процесі проведеного дослідження отримано висновок, що саме застосування теорії ігор у поєднанні з методом централізованого управління дозволить отримати найкращі результати для оптимізації вибору зарядних станцій і планування часу зарядки для кількох БПА.

Більш детально зупинимось на прикладі, який наведено в статті [35].

Будемо притримуватися наступних позначень:

- а) D1 – ідентифікатор першого БПА;
- б) D2 – ідентифікатор другого БПА;

- в) $D3$ – ідентифікатор третього БпА;
- г) $S1$ – ідентифікатор першої зарядної станції;
- д) $S2$ – ідентифікатор другої зарядної станції;
- е) $S3$ – ідентифікатор третьої зарядної станції.

Отже, у дослідженні [35] розглянуто випадок, де є три БпА ($D1, D2, D3$) і три зарядні станції ($S1, S2, S3$).

Всі три БпА мають потребу у заряджанні власного акумулятору в залежності від статусу зарядженості батареї та місій, які вони мають виконати.

Всі три зарядні станції мають різну потужність та струм навантаження.

В дослідженні зроблено наступне припущення:

- три БпА є гравцями (учасниками) у грі і взаємодіють між собою;
- винагородою у такій грі прийнято вважати: отримання заряду від станції та успішне виконання місії при максимальній економії часу;
- мета гри полягає в тому, щоб кожен БпА (гравець) міг потенційно отримати максимальну винагороду.

Розглянемо приклад 1.

В процесі дослідження змодельовано наступну ситуацію.

Вважаємо, що БпА $D1$ та БпА $D2$ необхідно підзарядити. Найближчою зарядною стацією до БпА $D1$ та БпА $D2$ є станція $S1$.

Випадок, коли обидва БпА одночасно спрямовують свої маршрути до єдиної зарядної станції $S1$, не буде оптимальним, оскільки БпА $D2$ втрачає час на виконання місії, чекаючи на завершення процесу підзарядки БпА $D1$.

Завдяки використанню ігрового методу, обрання зарядних станцій відбувається автоматично для БпА $D1$ та БпА $D2$ (тобто перехід на інші зарядні станції $S2$ або $S3$ зменшить час очікування для підзарядки).

Перейдемо до прикладу 2.

Іншим сценарієм, який змодельовано у дослідженні є такий.

БпА $D1$ потребує суттєвої підзарядки, але не критичної і може зачекати.

БпА $D2$ має більший заряд, але його пріоритет у виконанні місії вищий ніж у БпА $D1$.

В такому випадку БпА $D1$ пропускає БпА $D2$ уперед.

Отже, таку гру можна вважати кооперативною, тому що БпА співпрацюють для досягнення максимально найкращого спільного результату.

Розглянуті вище змодельовані ситуації ілюструють можливості застосування ігрових методів для успішного завершення поставленої місії.

Розглянемо приклад 3.

В процесі цього дослідження змодельовано іншу ситуацію.

Будемо притримуватися наступних позначень:

- а) $D1$ – ідентифікатор першого БпА;
- б) $D2$ – ідентифікатор другого БпА;
- в) $D3$ – ідентифікатор третього БпА;
- г) $S1$ – ідентифікатор першої зарядної станції;
- д) $S2$ – ідентифікатор другої зарядної станції;
- е) $S3$ – ідентифікатор третьої зарядної станції.

Кожен БпА ($D1, D2, D3$) має можливість обрати одну з трьох можливих зарядних станцій ($S1, S2, S3$).

Вважаємо, що винагорода, яку отримує кожний з БпА окремо – це кількість часу, який зекономить БпА, щоб дістатися однієї із можливих зарядних станцій.

Матриця винагород надана у таблиці 2.1.

Таблиця 2.1 – Матриця винагород

	$S1$	$S2$	$S3$
$D1$	2	1	3
$D2$	1	3	2
$D3$	3	2	1

Матриця винагород складається з БпА, зарядних станцій та часу, який БпА можуть зекономити при підключенні до фіксованої зарядної станції.

Наприклад, якщо БпА $D1$ переходить до станції $S1$, БпА $D2$ переходить до станції $S2$, а БпА $D3$ переходить до станції $S3$, тоді БпА $D1$ економить 2 одиниці часу, БпА $D2$ економить 3 одиниці часу, а БпА $D3$ економить 1 одиницю часу.

У прикладі 3 використано положення про матрицю винагород і рівноваги Неша.

За допомогою використання матриці винагород A показано винагороду, яку кожний i -й БпА отримує від вибору j -ї зарядної станції. Винагорода a_{ij} визначається як кількість часу, збереженого i -м БпА за умови вибору j -ї станції.

При цьому $i = \overline{1,3}$; $j = \overline{1,3}$.

Рівновага Неша – набір стратегій (дій) у грі з двома чи більше гравцями, при якому вибір оптимальної стратегії кожним гравцем відбувається з урахуванням наступних дій суперників.

У даному прикладі рівновага Неша – набір стратегій (вибір зарядних станцій), за яких жоден БпА не може збільшити свою винагороду шляхом односторонньої зміни своєї стратегії, не враховуючи стратегії інших безпілотників [35].

Рівновага Неша обчислена наступним чином:

1. Визначено винагороду для кожного БпА на підставі вибору зарядної станції;
2. Визначено, чи збільшує одностороння зміна стратегії винагороду для будь-якого БпА;
3. Якщо при зміні стратегії жоден БпА не може збільшити свою винагороду, то поточний набір стратегій є рівновагою Неша.

Сценарій 1: $D1 \rightarrow S1, D2 \rightarrow S2, D3 \rightarrow S3$

Винагороди: $D1 = 2, D2 = 3, D3 = 1$

Загальна винагорода всіх трьох БпА (гравців) = 6.

Жоден БпА не може збільшити свою вигоду, змінивши лише станцію.

Тоді $(S1, S2, S3)$ є рівновагою Неша.

Сценарій 2: $D1 \rightarrow S2, D2 \rightarrow S3, D3 \rightarrow S1$

Винагороди: $D1 = 1$, $D2 = 2$, $D3 = 3$

Загальна винагорода всіх трьох БпА (гравців) = 6.

Жоден БпА не може збільшити свою вигоду, лише змінивши станцію.

Тоді (S2, S3, S1) також є рівновагою Неша.

В обох випадках розподіл БпА між зарядними станціями досягає стану, коли будь-яка одностороння зміна стратегії БпА не призведе до збільшення його винагороди. Ця характеристика визначає рівновагу Неша для кожного сценарію.

Проведений аналіз правильно визначає рівновагу Неша на основі заданої матриці винагороди, демонструючи застосування ігрового методу для оптимізації вибору зарядних станцій для БпА [35].

2.2 Централізований і децентралізований метод керування безпілотними апаратами

Для керування безпілотними апаратами застосовуються як централізовані, так і децентралізовані методи керування [39–40]. Вибір того чи іншого методу здійснюється на підставі в залежності від задач, які треба вирішити.

Надамо означення централізованому методу керування БпА.

Централізований метод керування кількома БпА передбачає, що рішення про виконання усіх завдань (місій) всіма БпА (гравцями) приймається виключно одним оператором або з єдиного центру керування. Процес керування системою БпА відбувається за допомогою багаторівневої структури. Головною особливістю цього методу є збереження контролю за кожним БпА (гравцем).

В свою чергу, децентралізований метод керування кількома БпА – це відсутність централізованого управління; всі завдання та місії в цілому виконуються групою БпА (групою гравців) під управлінням кількох центрів керування.

Розглянемо більш детально основні положення централізованого та децентралізованого методів керування БпА.

Нижче наведено порівняльний аналіз централізованого та децентралізованого методів керування БПА.

Суть

Централізований метод:

- а) один оператор керує всіма БПА в мережі;
- б) оператор відповідає за весь комплекс завдань, у тому числі навігацію, керування місіями, та моніторинг стану систем;
- в) контроль також виконується оператором.

Децентралізований метод:

- а) БПА діють практично автономно та взаємодія відбувається за допомогою мережі;
- б) наявність декількох пунктів керування (операторів);
- в) оператори мають узгоджувати дії між собою, що передбачає появу попиту у розробці додаткових інструментів мінімізації людського фактору в процесі прийняття рішень на рівні «оператор-оператор»;
- г) права кожного оператора обмежені;
- д) кожен окремий оператор має права лише на рівні загального управління місіями та завданнями. Локальне виконання місії (місій) передбачає застосування автоматизації.

Переваги

Централізований метод:

- **Контроль:** Весь контроль за виконанням місії (місій) покладається на одного оператора.
- **Простота:** Відсутність запитів щодо розробки складних алгоритмів автономії.
- **Безпека:** Оператор приймає всі рішення одноосібно.

Децентралізований метод:

– **Масштабованість:** Збільшення кількості БпЛА підвищує навантаження на оператора, але ця проблема вирішується шляхом утворення чи введення в дію додаткових пунктів керування та залученням нових операторів;

– **Стійкість до відмов:** Втрата зв'язку з одним оператором чи БпЛА не зупиняє виконання місії (місій).

– **Менше навантаження на оператора:** Зменшення когнітивного навантаження на окремого оператора виключно у разі відсутності аварійних ситуацій (всі оператори і технічне забезпечення знаходяться у робочому стані).

Недоліки

Централізований метод:

– Успішність виконання місії (місій) повністю залежить від людського фактору.

– **Обмеження кількості БпЛА:** один оператор фізично не здатний ефективно керувати великою кількістю БпЛА.

– **Вразливість:** втрата працездатності оператора чи відмова обладнання пункту керування зупиняє всю систему і призводить до зриву виконання місії (місій).

– **Високий рівень навантаження:** оператор може бути перевантажений великою кількістю різноманітних завдань та відволікаючими подіями, що призводять до помилок.

– **Досвід оператора:** кваліфікація оператора має прямий вплив на виконання місії (місій).

Децентралізований метод:

– **Складність проєктування:** Потреба у створенні нових складних алгоритмів децентралізованого керування та відповідного програмного забезпечення.

– **Менш передбачувана поведінка:** Обмежені права оператора можуть призвести до складних та непередбачених системою ситуацій у тому числі навіть до зриву місії (місій).

– **Витрати:** Розробка автономних систем потребує значних людських, фінансових ресурсів та часу.

На основі результатів порівняльного аналізу розглянутих методів зробимо висновок.

Централізований метод ефективно застосовується у випадках, де потрібен моніторинг і безпосереднє втручання оператора. Централізований метод ефективно застосовується для обмеженої кількості БпА. Децентралізований метод складніший в реалізації та потребує значно більше ресурсів, у тому числі фінансових для вирішення деяких типів задач, які вимагають великої кількості БпА.

2.3 Комбінована централізовано-кооперативна математична ігрова модель керування БпЛА та БпНА

В процесі проведення наукового дослідження отримана комбінована централізовано-кооперативна математична ігрова модель керування БпЛА та БпНА, яка вирішує проблему наведену нижче.

Для виконання пошуково-рятувальної місії необхідно задіяти кілька (два) БпА різних типів: БпЛА та БпНА. Вантаж для пошуково-рятувальної місії знаходиться на БпНА.

БпЛА виконує функцію супроводження БпНА.

Місія вважається виконаною, коли необхідний вантаж буде неушкодженим доставлено до місця призначення в умовах оптимального вибору методу керування БпНА із врахуванням обмеженості людських, енергетичних та фінансових ресурсів.

Для вирішення цієї проблеми обрано централізований метод керування, який, як відомо за означенням, передбачає керування БпА з одного центру управління (насамперед одним оператором). Разом з тим, розроблена математична модель є кооперативною, оскільки в процесі моделювання розглянуто кілька (два) БпА, які виконують разом спільну місію, тобто дії одного пов'язані з діями іншого.

Крім того, в процесі побудови зазначеної моделі використано підходи, термінологію й апарат ігрового методу. Іншими словами запропонований ігровий підхід розв'язання задачі, засновано на теорії ігор (або теорії дослідження операцій).

Вважаємо, що:

- БпА – гравці;
- обрання фіксованого способу дії i -го БпА – обрання стратегії i -м гравцем, де $i = \overline{1, N}$; N – кількість гравців.

Введено припущення, що $N = 2$, тобто в подальшому розглядаємо двох гравців (БпЛА та БпНА).

Під загальною функцією виграшу $F_{\text{загальна}}$ маємо на увазі функцію, яка визначає головну ціль гри, а саме: найуспішніше вирішення поставленої місії від її початку і до кінця.

Результатом моделювання із застосуванням комбінованої централізовано-кооперативної математичної ігрової моделі керування БпА (за припущенням БпЛА та БпНА) є отримання максимуму функції виграшу.

В процесі побудови цієї ігрової математичної моделі введені наступні позначення:

$F_{\text{загальна}}$ – загальна функція виграшу коаліції гравців (двох гравців БпЛА та БпНА), яка визначає головну ціль гри та значення якої треба максимізувати.

Головне практичне значення застосування функції виграшу $F_{\text{загальна}}$ – виконати місію з максимальним покриттям території у реальному часі з найменшими витратами часу та ресурсів (людських, фінансових та енергетичних).

$$F_{\text{загальна}} = \sum_{j=1}^3 a_j F_j - \sum_{j=4}^5 a_j F_j \rightarrow \max, \quad (2.1)$$

де F_j – складові загальної функції виграшу;

При побудові моделі введено таке припущення: обрано п'ять основних найважливіших характеристик експлуатації БпА: покриття території, швидкість

пересування, корисне навантаження, енергоспоживання, витрати фінансових ресурсів.

Для успішного виконання поставленої задачі потрібна карта місцевості, де буде відбуватися доставка вантажу у важкодоступні місця або пошуково-рятувальна операція. Запропоновано в реальних умовах використовувати електронні картографічні ресурси, зокрема, OpenStreetMap.

Для побудови карти місцевості з подальшим отриманням результатів моделювання використовуються можливості симулятора.

Для кожної характеристики введено такі пріоритети:

a_j – j -ий пріоритет; $j = \overline{1,5}$;

$$\sum_{j=1}^5 a_j = 1, \text{ де } 0 \leq a_j \leq 1; j = \overline{1,5}, \quad (2.2)$$

де a_1 – пріоритет покриття; a_2 – пріоритет швидкості пересування БпА; a_3 – пріоритет корисного навантаження БпА; a_4 – пріоритет енергоспоживання; a_5 – пріоритет фінансових ресурсів.

Вважаємо, що всі функції f_i – нормалізовані.

p_i – пріоритет кожного i -го об'єкту керування (БпЛА та БпНА), де

$$0 \leq p_i \leq 1; i = \overline{1, N}; N = 2;$$

$$\sum_{i=1}^N p_i = 1; \quad (2.3)$$

F_1 – цільова функція, яка визначає максимальну площу покриття території коаліцією гравців:

$$F_1 = \sum_{i=1}^N p_i * f_{i \text{ покриття}}(x_i(t)), \quad (2.4)$$

де $f_{i \text{ покриття}}(x_i(t))$ – функція покриття i -го об'єкту керування (гравця) в момент часу t ;

$x_i(t)$ – стан i -го об'єкту керування в момент часу t , де $0 \leq t \leq T$; $t = \overline{0, T}$;

$t = 0$ – момент початку місії;

T – момент закінчення місії.

F_2 – цільова функція, яка визначає максимальну швидкість коаліції гравців:

$$F_2 = \sum_{i=1}^N p_i * f_{i \text{ швидкість}}(x_i(t)), \quad (2.5)$$

де $f_{i \text{ швидкість}}(x_i(t))$ – функція швидкості i -го об'єкту керування (гравця) в момент часу t .

F_3 – цільова функція, яка визначає максимальне навантаження коаліції гравців:

$$F_3 = \sum_{i=1}^N p_i * f_{i \text{ навантаження}}(x_i(t)), \quad (2.6)$$

де $f_{i \text{ навантаження}}(x_i(t))$ – функція навантаження i -го об'єкту керування (гравця) в момент часу t .

F_4 – цільова функція, яка визначає мінімальну витрату енергії коаліцією гравців:

$$F_4 = \sum_{i=1}^N p_i * f_{i \text{ енерг.}}(u_i(t)), \quad (2.7)$$

$$f_{i \text{ енерг.}}(u_i(t)) = (\|u_i(t)\|)^2, \quad (2.8)$$

де $f_{i \text{ енерг.}}(u_i(t))$ – функція енергоспоживання i -го об'єкту керування (гравця) в момент часу t ;

$u_i(t)$ – керуюча функція для i -го об'єкту керування (гравця) в момент часу t , де $t = \overline{0, T}$.

F_5 – цільова функція, яка визначає мінімальні фінансові витрати коаліцією гравців:

$$F_5 = \sum_{i=1}^N p_i * f_{i \text{ фінанс.}}(u_i(t)), \quad (2.9)$$

$$f_{i \text{ фінанс.}}(u_i(t)) = (u_i(t))^2, \quad (2.10)$$

де $f_{i \text{ фінанс.}}(u_i(t))$ – функція фінансових витрат i -го об'єкту керування (гравця) в момент часу t .

Вводимо означення ідеальної функції виграшу $F_{\text{ідеальна}}$.

$F_{\text{ідеальна}}$ – максимально можливе значення функції виграшу з нульовим енергоспоживанням та нульовими фінансовими витратами (ідеальна функція виграшу).

$$F_{\text{ідеальна}} = \sum_{j=1}^3 a_j F_j; \quad (2.11)$$

$$\text{Ефективність} = \frac{F_{\text{загальна}}}{F_{\text{ідеальна}}} * 100\% \quad (2.12)$$

На рисунку 2.1 продемонстровано залежність цільової функції від типу моделі.

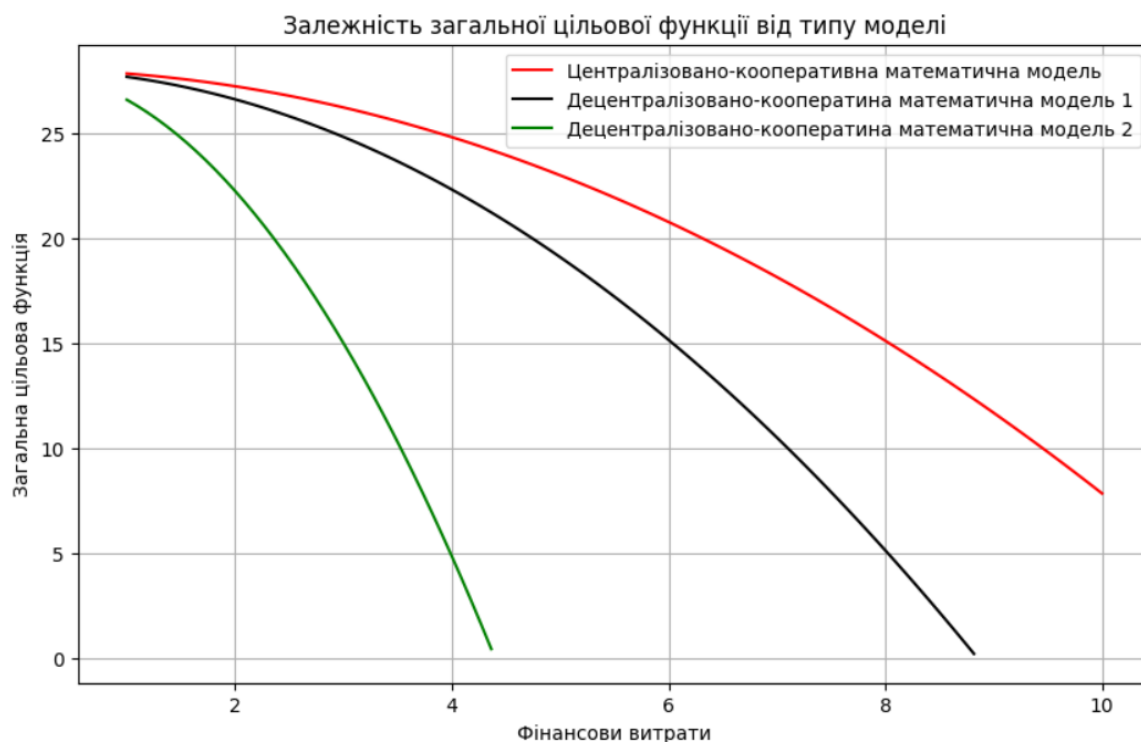


Рисунок 2.1 – Залежність функції виграшу від типу моделі

Наступні розрахунки наведено у додатку Б:

1. Комбінована централізовано-кооперативна (один центр керування):

$$\text{Ефективність} = \frac{F_{\text{загальна}}}{F_{\text{ідеальна}}} * 100\% = 71,12\%;$$

2. Децентралізовано-кооперативна 1 (два центри керування):

$$\text{Ефективність} = \frac{F_{\text{загальна}}^I}{F_{\text{ідеальна}}} * 100\% = 65,06\%;$$

3. Децентралізовано-кооперативна 2 (п'ять центрів керування):

$$\text{Ефективність} = \frac{F_{\text{загальна}}^{II}}{F_{\text{ідеальна}}} * 100\% = 22,66\%.$$

Зазначимо, що $F_{\text{загальна}}^I$ – загальна функція виграшу при застосуванні децентралізовано-кооперативної моделі 1; $F_{\text{загальна}}^{II}$ – загальна функція виграшу при застосуванні децентралізовано-кооперативної моделі 2.

Різниця в ефективності між централізовано-кооперативною та децентралізовано-кооперативними моделями становить 6,06% для першого випадку та значні 48,46% для другого, що вказує на суттєве зменшення значення загальної цільової функції через зростання витрат.

На основі отриманої моделі розроблено централізовано-кооперативний ігровий метод керування БпЛА та БпНА, який об'єднує переваги як централізованого методу, так і кооперативного методу:

1. Один оператор керує обома БпЛА та БпНА в мережі;
2. Оператор відповідає за весь комплекс завдань, у тому числі навігацію, керування місіями та моніторинг стану систем;
3. Оператор одноосібно виконує контроль за виконання місії;
4. Кооперативний ігровий метод застосовується для вирішення завдань оптимізації та координації дій між БпЛА та БпНА.

Переваги комбінованої централізовано-кооперативної математичної моделі керування БпЛА та БпНА у порівнянні з децентралізованою кооперативною математичною моделлю керування БпЛА та БпНА наступні:

- економія людських ресурсів, оскільки кількість операторів не більше одного;
- економія енергоспоживання, оскільки тільки один центр керування;
- економія фінансових ресурсів, оскільки місія виконується при мінімальних витратах.

2.4 Огляд механізмів прийняття рішень для пошуку оптимального шляху. Результати проведення порівняльного аналізу стандартних алгоритмів прийняття рішень

Розглянемо алгоритми пошуку оптимального (найкоротшого) шляху для керування безпілотними наземними апаратами (БпНА).

Алгоритм, який використовується для визначення оптимального шляху, повинен задовольняти певним (наперед заданим) значенням обраних параметрів. Зокрема, значення розрахункового часу при використанні обраного алгоритму повинно бути найменшим у порівнянні із значеннями цього параметру при застосуванні інших алгоритмів визначення оптимального шляху.

Отримано результати проведення аналізу чотирьох алгоритмів, які найчастіше застосовуються для пошуку оптимального (найкоротшого) шляху [36].

У роботі надано порівняльний аналіз алгоритмів пошуку найкоротшого шляху: алгоритм Дейкстри, A-Star, Bi-Directional A-Star і Rapidly-exploring random tree (RRT). Кожен з них має різні переваги та обмеження щодо продуктивності, вимог до пам'яті та точності.

В проведеному науковому дослідженні [36] порівняння цих чотирьох алгоритмів виконано за наступними характеристиками: повнота, оптимальність і часова складність.

Визначення основних характеристик алгоритмів A-Star, BiA-Star Дейкстри, RRT

Надамо означення обраних характеристик.

Повнота алгоритму означає, що його використання завжди дозволить знайти шлях (якщо цей шлях існує) [36].

Під оптимальністю алгоритму мається на увазі пошук найкоротшого шляху, за умови існування такого шляху.

Відповідно до введеного означення оптимальності алгоритми A-Star, BiA-Star та Дейкстри є оптимальними. В той же час, алгоритм RRT не є оптимальним, оскільки шлях, який він знаходить, може не бути найкоротшим [36].

Часова складність – характеристика алгоритму, яка вказує на кількість операцій, в залежності від розміру вхідних даних.

Аналіз часової складності алгоритмів вказує на те, що алгоритми A-Star, BiA-Star і Дейкстри мають однакову часову складність $O(k^n)$, де k – коефіцієнт розгалуження, а n – глибина рішення. При цьому, під коефіцієнтом розгалуження розуміємо середню кількість ребер, які виходять з кожної вершини графу; під глибиною рішення розуміємо мінімальну кількість кроків, які необхідно пройти для знаходження оптимального шляху від початку шляху до його завершення.

У деяких випадках BiA-Star може бути швидшим, ніж A-Star та алгоритм Дейкстри, оскільки алгоритм BiA-Star виконує пошук із початкового та цільового вузлів одночасно, що може зменшити кількість вузлів, які необхідно пройти [36].

Алгоритми A-Star, BiA-Star та Дейкстри найчастіше використовуються для пошуку шляху у місцевості, де перешкоди статичні. RRT застосовується за наявності динамічних перешкод, тобто умов, які часто змінюються і шлях необхідно знаходити знову.

Аналіз алгоритмів A-Star, BiA-Star Дейкстри, RRT

Порівняльний аналіз алгоритмів A-Star, BiA-Star Дейкстри, RRT надано у таблиці 2.2.

Таблиця 2.2 – Порівняння A-Star, BiA-Star, Дейкстри та RRT алгоритмів [36]

	Алгоритм			
	A-Star	BiA-Star	Дейкстри	RRT
Повнота	Так	Так	Так	Так

Продовження таблиці 2.2

	Алгоритм			
	A-Star	BiA-Star	Дейкстри	RRT
Оптимальність	Так	Так	Так	Ні
Часова складність	$O(k^n)$	$O(k^{n/2})$	$O(k^n)$	Може змінюватися

A-Star – це алгоритм пошуку шляху, який використовує евристичну функцію для пошуку кінцевого вузла [36]. Евристична функція оцінює відстань між заданою вершиною та кінцевою вершиною. Алгоритм використовує цю інформацію, щоб визначити пріоритет вузлів, які знаходяться ближче до цілі. За допомогою алгоритму знаходиться вузол, який має найнижчу загальну довжину шляху. Загальна довжина шляху визначається як сума двох значень, а саме: довжини шляху від початку шляху до фіксованої точки n^* і значенню розрахункової функції, яка знаходить найменший шлях від фіксованої точки n^* до кінцевої точки N (точки закінчення місії).

BiA-Star – це двонаправлена версія A-Star, яка одночасно виконує пошук і з початкового вузла, і з кінцевого вузла. Алгоритм завершується, коли шлях знайдено. Алгоритм BiA-Star потенційно може бути швидшим за A-Star, оскільки він проходить з початкового і кінцевого вузлів одночасно, що зменшує загальну кількість вузлів, які необхідно пройти [36].

Алгоритм Дейкстри – це алгоритм пошуку шляху, який знаходить найкоротший шлях між початковим вузлом і всіма іншими наявними вузлами. Він працює, починаючи з початкового вузла та завдяки повторним діям шукає вузол із найменшим ребром. Алгоритм зупиняється, коли він досягає кінцевого вузла або коли всі доступні вузли в просторі пошуку пройдено [36].

RRT – алгоритм, який використовується для планування руху в динамічних середовищах, але він в повному обсязі не враховує повноту й оптимальність. Разом з тим, передбачена можливість застосування цього алгоритму у складних і

динамічних середовищах. Він, як правило, використовується в робототехніці та автономних системах, оскільки має здатність враховувати динамічні перешкоди [36].

Оцінка продуктивності алгоритмів A-Star, BiA-Star Дейкстри, RRT у статичному середовищі

Для визначення продуктивності зазначених алгоритмів пошуку шляху в статичному середовищі, проведено експерименти, метою яких було порівняння ефективності алгоритмів пошуку оптимальних шляхів з урахуванням статичних перешкод [36].

Експерименти проводилися з кожним із чотирьох алгоритмів пошуку оптимальних шляхів. Для проведення експериментів обрана мова програмування Python, оскільки Python надає необхідні інструменти та функціональні можливості тестування алгоритмів і аналізу їх продуктивності.

Кожний алгоритм застосовувався для вирішення однакового завдання – знайти оптимальний шлях від початкового вузла до кінцевого вузла із врахуванням наявності статичних перешкод. Для всіх чотирьох експериментів надані однакові початковий і кінцевий вузли. Початковий та кінцевий вузли представлені у вигляді матриці-лабіринту на наступному рисунку 2.2.

```
maze = [[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 1, 0, 0, 1, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

start = (0, 0)
end = (9, 8)
```

Рисунок 2.2 – Матриця-лабіринт [36]

Матрицю-лабіринт представлено у вигляді матриці, де перешкоди позначені значенням «1», а зони без перешкод значенням «0». Ця матриця-лабіринт стала основою для алгоритмів навігації та пошуку оптимальних шляхів. Експерименти

мали на меті отримати цінну інформацію про сильні та слабкі сторони кожного алгоритму в статичному середовищі.

Розрахунковий час проходження матриці-лабіринту вирішено використовувати для демонстрації можливостей цих алгоритмів.

В результаті визначено один алгоритм, який найбільше підходить для використання у реальних програмних додатках, що включають планування шляху з урахуванням статичних перешкод [36].

Зазначимо, що при проведенні експериментів враховувалася наявність виключно статичних перешкод. Динамічні перешкоди при проведенні експериментів не розглядалися.

Експеримент, проведений у [36] надав інформацію про розрахунковий час виконання (продуктивність) алгоритмів A-Star, BiA-Star, Дейкстри та RRT за наявності статичних перешкод.

Результати експериментів представлені у таблиці 2.3.

Таблиця 2.3 – Порівняння A-Star, BiA-Star, Дейкстри та RRT алгоритмів [36]

Алгоритм	Тип перешкоди	Розрахунковий час (сек)
A-Star	Статична	20.26
BiA-Star	Статична	30.46
Dijkstra's	Статична	49.96
RRT	Статична	130.34

За результатами експериментів, найкращим із чотирьох алгоритмів є A-Star, оскільки він має найменший розрахунковий час виконання. Отже, саме цей стандартний алгоритм обрано в якості модуля для програмної реалізації оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

2.5 Оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності

Оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності побудована для отримання оптимального маршруту руху БпА з урахуванням ризику та невизначеності в появі різного виду фізичних перешкод та небажаних подій, наявність яких необхідно було врахувати в процесі моделювання.

Врахування таких перешкод та подій безпосередньо пов'язано із якістю побудованої моделі. Основні поняття, запропоновані при побудові моделі.

Місія – доставка вантажу із пункту відправлення до пункту призначення.

Успішне виконання місії – виконання завдання в повному обсязі, а саме: доставка вантажу в неушкодженному стані з місця відправлення до місця призначення з найменшими витратами.

Невизначеність – поява/відсутність перешкод (подій), поява яких потенційно загрожує виконанню місії.

Побудова математичної моделі виконана з використанням основних положень та підходів теорії ігор (теорія досліджень операцій) та теорії графів.

В процесі побудови математичної моделі до врахування невизначеності явища або події підходимо з різних сторін, в залежності від характеру того чи іншого небажаного явища (перешкоди) або події.

Для побудови термінологічного та понятійного апарату моделі частково застосовано означення невизначеності [41].

В подальшому викладенні побудованої в результаті проведеного наукового дослідження математичної моделі враховано таке.

Обмеженість та/або неточність вхідних даних (вхідної інформації) дає підстави розглядати два різних види ситуацій, які необхідно враховувати в процесі знаходження рішення:

1. Рішення приймаються в умовах ризику;
2. Рішення приймаються в умовах невизначеності.

Якщо рішення приймаються в умовах ризику, то появу небажаного явища, події чи перешкоди можна моделювати, використовуючи апарат теорії ймовірностей, тобто із знаходженням та використанням функції розподілу ймовірностей випадкової величини.

Якщо рішення доводиться приймати в умовах невизначеності, процес знаходження функції розподілу ймовірностей або потребує значних зусиль, або знаходження цієї функції взагалі неможливо [41].

За умови невизначеності функція розподілу може бути або невідома, або її знайти неможливо.

В запропонованій моделі рішення приймалися за умови наявності ризику, тобто вирішальним критерієм при формуванні рішення визнана поява найбільш ймовірної небажаної події, наприклад, весняного бездоріжжя при майбутньому формуванні маршруту. Використовувалися статистичні метеорологічні спостереження за визначений перебіг часу [42].

Ризик – поява небажаного явища/події чи перешкоди, які можна моделювати, використовуючи апарат теорії ймовірностей.

Крім того, в цій моделі рішення приймалися і в умовах невизначеності, наприклад, раптової появи ями або іншої великої перешкоди на шляху маршруту руху. Стосовно такої непередбачуваної ситуації застосовувався інший підхід – апарат теорії ігор та теорії графів [43–44], в поєднанні з можливостями нейронної мережі та навчання з підкріпленням. Більш детально застосування нейронної мережі для подолання фактору невизначеності розглядається у третьому розділі дисертації.

Вважаємо, що для моделювання карта місцевості отримана за допомогою симулятора. Граф будується засобами, які доступні при застосуванні симулятора.

При побудові запропонованої моделі використано такі основні поняття теорії графів:

– граф G – фігура на площині, яка складається з непорожньої скінченної множини V точок (вузлів) і скінченної множини E ліній (ребер), що з'єднують деякі пари вершин [43];

- ребро графа – лінія, яка з'єднує фіксовану пару вузлів;
- вузол графа – точка на графі, яка представляє об'єкт;
- орієнтований граф – граф, у якому кожне ребро має напрямок.

Визначається множиною вузлів та множиною впорядкованих пар ребер. Кожне ребро представлено парою вузлів (A, B), де існує ребро від вузла A до вузла B;

– зважений граф – граф, кожному ребру якого поставлено у відповідність деяке значення [44];

– елементарний шлях – шлях, в якому будь-яка вершина не зустрічається двічі; при кожному кроці зростають можливості вибору нових вершин. Повторне проходження однієї й тієї ж вершини не передбачено [44].

Перейдемо до безпосереднього викладення оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

Загальна функція оптимізації шляху (функція виграшу):

$$Q_{\text{заг}} = \min \sum_{n=1}^N q(n), \quad n = \overline{1, N}, \quad (2.13)$$

де $Q_{\text{заг}} \rightarrow \min$ – функція визначає оптимальний (мінімально короткий) шлях для успішного виконання місії;

n – n -ий вузол;

N – останній вузол, при якому місія виконана (точка прибуття);

$q(n)$ – загальна довжина шляху через вузол n .

Перший етап задачі: подолання ризиків

При побудові моделі використано орієнтований зважений граф G .

Координати всіх вузлів, у тому числі координати початкового пункту вузла (пункту відправлення) і координати кінцевого вузла (пункту призначення) визначаються за допомогою GPS.

I – множина, простір всіх можливих ребер графу, за якими потенційно може рухатися БпНА;

i – i -е ребро графа, за яким потенційно може рухатися БпНА; $i \in I$; $i = \overline{1; I}$;

ξ_i – випадкова дискретна величина, значення якої відображає стан i -го ребра графу.

За припущенням вважаємо:

ξ_i – може приймати тільки два значення, а саме:

$\xi_i = \{x_j\}$, де $j = \overline{1; 2}$;

x_j – можливі значення випадкової величини ξ_i .

Вважаємо, що:

$$\xi_i = \begin{cases} x_1, & \text{якщо } i \text{ – ребро (ділянка шляху) непрохідне;} \\ x_2, & \text{якщо } i \text{ – ребро (ділянка шляху) прохідне.} \end{cases} \quad (2.14)$$

Вважаємо: $x_1 = 0$; $x_2 = 1$;

Тоді

$$\xi_i = \begin{cases} 0, & \text{якщо } i \text{ – ребро (ділянка шляху) непрохідне;} \\ 1, & \text{якщо } i \text{ – ребро (ділянка шляху) прохідне.} \end{cases} \quad (2.15)$$

p_i – ймовірність того, що i -е ребро може бути непрохідним через бездоріжжя;
 $0 \leq p_i \leq 1$, де $i \in I$.

K – кількість проведених спостережень (експериментів);

K^0 – кількість проведених спостережень (експериментів), коли i -е ребро було непрохідним.

k – проведене спостереження (експеримент) за k -й період часу; $k \in K$; $k = \overline{1; K}$; k – дискретна величина;

C – константа, задається оператором в якості попередньої умови до початку виконання місії; $C = const$; $0 \leq C \leq 1$; в залежності від складності задачі значення C може змінюватися оператором, як в більшу сторону, так і в меншу.

Метеорологічні дані:

t_k – температура повітря за k -й період часу (k -е спостереження);

d_k – опади (їх інтенсивність) за k -й період часу (k -е спостереження).

В результаті проведеного наукового дослідження отримано $F(x)$ – розрахункову функцію розподілу ймовірностей випадкової величини ξ_i для кожного фіксованого i -го ребра, де $i \in I, i = \overline{1, I}$.

$$F(x) = \begin{cases} 0, \text{ якщо } \sum_{k=1}^K x_k = 0; \\ p_k(x), \text{ якщо } 0 < \sum_{k=1}^K x_k < K; \\ 1, \text{ якщо } \sum_{k=1}^K x_k = K; \end{cases} \quad (2.16)$$

При цьому, $p_k = \frac{K^0}{K}$ – ймовірність того, що i -е ребро непрохідне.

Прокоментуємо значення $F(x)$.

$F(x) = 0$, тоді i -е ребро вважається непрохідним.

$F(x) = 1$, тоді i -е ребро вважається прохідним.

$F(x) = p_k(x)$ – це ймовірність того, що i -е ребро може бути непрохідним.

$P\{\xi_i = 0\}$ – ймовірність події, коли випадкова величина $\xi_i = 0$, тобто ймовірність того, що $\xi_i = x_1 = 0 \Rightarrow i$ -е ребро непрохідне.

Вводимо умову:

– якщо $P\{\xi_i = 0\} \geq C$, то i -е ребро непрохідне, тобто ймовірність, що i -е ребро непрохідне повинна дорівнювати або перевищувати наперед задане число $C = \text{const}$; i -е ребро виключається з подальшого формування маршруту;

– якщо $P\{\xi_i = 0\} < C$, то i -е ребро вважається прохідним і включається до подальшого формування маршруту.

Результатом застосування першої частини оптимізаційної ігрової математичної моделі керування БпЛА та БпНА є отримання такого оновленого переліку ребер графу з якого виключено проблемні ділянки руху (ребра), які з великою ймовірністю (ризиком) можуть бути непрохідними.

Продемонструємо на прикладі як визначається ризик i -го ребра відповідно до наведеної першої частини моделі. Таблиця 2.4 містить дані метеорологічних спостережень станом на 12 березня за фіксовану вибірку років для певної місцевості, де знаходиться i -е ребро графа.

Таблиця 2.4 – Стан i -го ребра в залежності від метеорологічних даних на 12 березня

Роки	k	t	Опади	Стан дороги	ξ_i	P	C
1	2	3	4	5	6	7	8
2015	1	3	+++	-	0	0,6	0,3
2016	2	2	+++	-	0		
2017	3	3	++	+	1		
2018	4	0	+++	-	0		
2019	5	7	+	+	1		
2020	6	4	+	+	1		
2021	7	4	++	+	1		
2022	8	2	+++	-	0		
2023	9	2	+++	-	0		
2024	10	3	+++	-	0		

Пояснення до таблиці 2.4:

– перший стовпчик – вибірка років, за яку взяті спостереження станом на 12 березня;

– другий стовпчик – номер спостереження (експерименту);

– третій стовпчик – показник температури за 12 березня по кожному року;

– четвертий стовпчик – відображає інтенсивність опадів станом на 12 березня по кожному року, а саме: +++ означає дуже сильні опади, ++ означає помірні опади, + незначні опади або їх відсутність;

– п'ятий стовпчик – заповнюється експертом в залежності від показників першого та другого стовпчиків та особливостей їх впливу на стан i -го ребра маршруту руху; - означає, що i -е ребро непрохідне; + означає, що i -е ребро прохідне;

– шостий стовпчик – 0 означає, що i -е ребро непрохідне; 1 означає, що i -е ребро прохідне;

– сьомий стовпчик – обчислюємо P – ймовірність події, що i -е ребро непрохідне, де $P_i = P\{\xi_i = 0\} = \frac{K^0}{K}$, де K^0 – кількість років спостережень, коли i -е ребро непрохідне; K – кількість всіх років спостережень;

– восьмий стовпчик – містить єдине наперед задане число C , де C – константа; $C = const$; $0 \leq C \leq 1$.

Розглянемо таблицю 2.4 для визначення можливості використання i -го ребра маршруту руху.

Так, у шостому стовпчику таблиці 2.4 отримано результат $P\{\xi_i = 0\} = 0,6$, при цьому $P\{\xi_i = 0\} \geq C$, де $C = 0,3$. Робимо висновок, що існує велика ймовірність (ризик) того, що i -е ребро непрохідне. Отже, при подальшому формуванні маршруту i -е ребро з розгляду виключається.

Наведений алгоритм перевірки i -го ребра (прохідне/непрохідне) в моделі передбачено застосовувати до кожного з ребер графу.

На першому етапі отримується оновлений перелік ребер, які доцільно розглядати як прохідні.

Другий етап задачі складається з:

– розгляду оновленого переліку ребер після виконання першого етапу моделі;

– використання нейронної мережі по розпізнаванню великих статичних перешкод за допомогою літального БпА (детальний опис способів навчання та налаштування нейронних мереж у рамках інформаційної технології керування БпА наведено у розділі 3).

Третій етап задачі: знаходження оптимального шляху з урахуванням наявності великих статичних перешкод та оновленого переліку ребер графу (виключено ребра, які з великою ймовірністю можуть бути непрохідними). Отримано оновлений варіант графу. Застосовано алгоритм A-Star.

$Q_{\text{заг}} \rightarrow \min$ – довжина шляху повинна бути найкоротшою для виконання місії.

$q(n)$ – загальна довжина шляху через вузол n .

$$q(n) = g(n) + h(n), \quad (2.17)$$

де $g(n)$ – функція, значення якої дорівнюють довжині шляху від початкового вузла до n -го вузла;

$h(n)$ – розрахункова функція, яка знаходить шлях від вузла n до кінцевого вузла N .

Четвертий етап задачі: виявлення фактору невизначеності.

Загальну блок-схему роботи алгоритму керування обома БпА надано на рисунках 2.5–2.8 у підрозділі 2.6 «Опис архітектури інформаційної системи керування з використанням нейронних мереж та ігрових підходів». Алгоритм навчання з підкріпленням наведено у розділі 3 «Експериментальна перевірка ігрового підходу та нейронних мереж у керуванні безпілотними апаратами».

Під фактором невизначеності розуміємо раптову появу великої ями або іншої суттєвої статичної перешкоди, яку не вдалося розпізнати нейронною мережею БпЛА.

В результаті застосування зазначених вище алгоритмів передбачено:

– БпНА долає перешкоду без додаткових дій (втручання) центру керування та проведення повторних розрахунків (використання камер та сенсорів разом з навчанням з підкріпленням потенційно забезпечує подолання перешкоди) (перехід до шостого етапу);

– у разі неподолання перешкоди БпНА фіксує поточний маршрут як неможливий, записує поточні координати і передає ці дані до центру керування та БпЛА (перехід до п'ятого етапу задачі).

П'ятий етап задачі: подолання фактору невизначеності.

Виконується виключно тоді, коли в процесі руху БпНА на якомусь з ребер з'являється фактор невизначеності (раптова перешкода), яку БпНА не подолав.

Задача набуває циклічного характеру (заново виконуються перший, другий, третій етапи), а саме:

- у першому етапі отримується оновлений перелік ребер, які доцільно розглядати як прохідні. З цього переліку виключено ребро з виявленою раптовою перешкодою;

- використовується інформація, отримана в результаті виконання другого етапу задачі;

- отримано оновлений варіант графу, застосовано алгоритм A-Star.

За замовчуванням вважаємо, що від початку та до кінця місії фактор невизначеності більше одного разу з'являтися не буде.

На цьому етапі отримано оновлений граф з оптимальним маршрутом, в якому відсутнє ребро, де виникла невизначена (раптова) перешкода. Повторно визначено оптимальний маршрут від поточної точки знаходження БпНА до точки прибуття.

Тобто отримано остаточний маршрут руху для наземного БпА.

Шостий етап: прибуття у пункт призначення. Успішне завершення місії.

Ефективність отриманої моделі

Для оцінки ефективності побудованої моделі запропоновано врахувати п'ять найважливіших критеріїв: успішне виконання місії; довжина маршруту; економія часу; здатність подолання можливих ризиків в процесі виконання місії; здатність подолання невизначеності при виконанні місії.

Запропоновано розглядати наведені вище п'ять критеріїв для:

- ідеальної моделі, коли місія виконана повністю з найменшими витратами часу, найкоротшою довжиною маршруту, а також з урахуванням можливих ризиків та невизначеності (далі – модель №1);

- реальної оптимізаційної моделі, коли успішне виконання місії проблематичне, оскільки довжина маршруту знайдена оптимальна, але не враховані можливі ризики та невизначеність, які можуть з'явитися в процесі виконання місії [45] (далі – модель №2);

– реальної оптимізаційної моделі із врахуванням ризиків, появу яких можна формалізувати за допомогою апарату теорії ймовірностей. Разом з тим, інші небажані події, які розглядаються як «невизначеність» ігноруються. Успішне виконання місії – проблематичне [45] (далі – модель №3);

– реальної оптимізаційної моделі, отриманої в результаті виконаного наукового дослідження. Застосування цієї моделі дозволяє отримати оптимальний маршрут успішного виконання місії з урахуванням ризиків і невизначеності щодо появи небажаних подій (далі – модель №4).

При побудові реальної моделі, отриманої в науковому дослідженні, було прагнення отримати модель, щоб значення її критеріїв були найближчими до значень цих критеріїв для ідеальної моделі.

В таблиці 2.5 наведено значення кожного із п'яти визначених критеріїв успішності (якості) застосування моделі у відсотках до значення відповідного критерію ідеальної моделі (моделі №1).

За замовчуванням вважаємо, що всі п'ять критеріїв визначених вище і наведених у другому стовпчику таблиці 2.5, для ідеальної моделі виконуються на 100% (модель №1). Розглянуті та опрацьовані оптимізаційні моделі шляху [46].

Таблиця 2.5 – Значення критеріїв моделей

Критерії	Моделі оптимізаційні			
	Модель №1 (%)	Модель №2 (%)	Модель №3 (%)	Модель №4 (%)
1	2	3	4	5
Успішне виконання	100	54	75	95
Довжина маршруту	100	140	240	170
Час виконання	100	375	252	155
Подолання ризиків	100	0	100	100
Подолання ризиків та невизначеності	100	0	0	100

Результати оптимізації, отриманої відповідно моделі №2 наведеної у [45], свідчать, що успішне виконання місії в реальних умовах ризику та невизначеності дорівнює 54% по відношенню до ідеального. Ризики та невизначеність не враховуються, тому довжина маршруту збільшиться на 40%, а час виконання місії збільшується на 275% в порівнянні із значеннями аналогічних критеріїв моделі №1. Подолання ризиків та невизначеності нульові. Інформація щодо моделі №2 наведена у третьому стовпчику таблиці 2.5.

Результати оптимізації, отриманої відповідно моделі №3 наведеної у [45], значно кращі ніж у моделі №2, а саме: успішне виконання місії становить 75% від ідеального, враховано 100% ризиків, довжина маршруту на 140% більша за ідеальну, час виконання місії становить 252% від ідеального. Значне покращення результатів моделі №3 у порівнянні з результатами моделі №2, отримано завдяки врахуванню ризиків при виконанні місії [45].

Результати оптимізації моделі, отриманої в результаті наукового дослідження, тобто моделі №4, такі. Успішність виконання місії становить 95%. Довжина маршруту зменшується в порівнянні з відповідним показником моделі №3. Час виконання місії зменшується у порівнянні з моделями №2 та №3.

Модель №4 враховує не тільки ризики, але й невизначеності шляху при виконанні місії.

Результати обчислення оптимізації шляху надано у додатку В. Разом з тим, врахування ризиків і невизначеності забезпечує повне успішне виконання місії.

Переводимо значення критеріїв в бали для подальшого аналізу ефективності моделі (таблиця 2.6).

Вважаємо, що 100 балів «успішності виконання місії» та 10 балів по кожному іншому критерію має виключно ідеальна модель.

Таблиця 2.6 – Аналіз ефективності моделей

Критерії		Моделі оптимізаційні			
		Модель №1 (бали)	Модель №2 (бали)	Модель №3 (бали)	Модель №4 (бали)
	1	2	3	4	5
1	Успішне виконання	100	54	75	95
2	Оцінка довжини маршруту	10	-4	-14	-7
3	Оцінка часу виконання	10	-27,5	-15,2	-5,5
4	Подолання ризиків	10	0	10	10
5	Подолання невизначеності	10	0	0	10
6	Всього балів	140	22,5	55,8	102,5

Далі наведено пояснення до змісту таблиці 2.6.

Критерій «Успішне виконання місії» за припущенням для ідеальної моделі (модель №1) дорівнює 100 балів. Отже, з таблиці 2.5 витікає, що цей критерій для моделі №2 в балах становить 54, для моделі №3 становить 75, для отриманої моделі №4 становить 95 (балів).

Критерій «Оцінка довжини маршруту» за припущенням у випадку моделі №1 дорівнює 10 балів. Тоді при переведенні в бали довжина маршруту для моделі №2 становить 14 балів. Отже, різниця між ідеальною довжиною маршруту і реальним маршрутом, отриманим моделлю №2 становить $\Delta = 10 - 14 = -4$ бали. Іншими словами, отриманий маршрут у порівнянні з ідеальним, якщо брати за модулем, відрізняється на 4 бали.

Цей алгоритм застосовується для обчислення зазначеного критерію для моделей №2-№4.

Критерій «Оцінка часу виконання місії» за припущенням у випадку моделі №1 дорівнює 10 балам. Тоді при переведенні в бали час виконання місії для моделі

№2 становить 37,5 балів. Отже, різниця між ідеальним часом виконання місії і реальним часом виконання, отриманим моделлю №2 становить $\Delta_1 = 10 - 37,5 = -27,5$ бали. Іншими словами, отриманий час виконання місії у порівнянні з ідеальним, якщо брати за модулем, відрізняється на 27,5 бали.

Цей алгоритм застосовується для обчислення зазначеного критерію для моделей №2-№4.

Критерій «Подолання ризиків» за припущенням для моделі №1 становить 10 балів. Тоді для моделі №2 цей критерій дорівнює 0 балів, оскільки ця модель не враховує появу ризиків. Моделі №3 та №4 враховують появу ризиків. Отже, зазначений критерій для моделей №3 та №4 становить відповідно 10 балів.

Критерій «Подолання невизначеності» за припущенням для моделі №1 становить 10 балів. Тоді для моделей №2 та №3 зазначений критерій дорівнює 0 балів, оскільки ці моделі не враховують появу невизначеності. Модель №4 враховує появу невизначеності. Отже, зазначений критерій для моделі №4 становить 10 балів.

Критерій «Всього балів» розраховується для кожної моделі окремо. Значення цього критерію для кожної моделі дорівнює сумі балів всіх зазначених вище критеріїв цієї моделі.

Переходимо до визначення ефективності кожної із чотирьох моделей.

Ефективність моделі розраховуємо за наступною формулою:

$$E = \frac{E_1}{E_2} * 100\%, \quad (2.18)$$

$$E_1 = \sum_{i=1}^5 e_i, \quad (2.19)$$

де e_i – значення i -го критерію для реальної моделі, $i = \overline{1,5}$, e_i береться з таблиці 2.6.

$$E_2 = \sum_{i=1}^5 e_i^*, \quad (2.20)$$

де e_i^* – значення i -го критерію для ідеальної моделі, $i = \overline{1,5}$, e_i^* береться з таблиці 2.6.

Результати розрахунків ефективності кожної моделі наведено у таблиці 2.7.

Таблиця 2.7 – Ефективність моделі

Модель №1 (%)	Модель №2 (%)	Модель №3 (%)	Модель №4 (%)
1	2	3	4
100	16,1	39,9	73,2

Ефективність ідеальної моделі (модель №1) взята за 100%.

Ефективність моделей, що розглядалися у порівнянні з ідеальною така:

- ефективність моделі №2 становить 16,1% (ризики та невизначеність не враховуються);
- ефективність моделі №3 становить 39,9% (ризики враховуються);
- ефективність моделі №4 становить 73,2% (ризики та невизначеність враховуються).

Отже, найбільшу ефективність з розглянутих реальних моделей має модель, отримана в результаті проведеного наукового дослідження «Оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності».

2.6 Опис архітектури інформаційної системи керування з використанням нейронних мереж та ігрових підходів

У сучасних системах керування безпілотними апаратами використання централізованого методу для керування кількома, зокрема, двома БПА є ефективним підходом для координації групових дій.

Централізований метод виконує ключову роль у прийнятті рішень, забезпечуючи узгодженість дій всіх БпА.

Централізовано-кооперативний ігровий метод дозволяє оцінити стратегічну взаємодію між БпА, а саме: БпЛА та БпНА та зовнішнім середовищем.

В результаті проведення наукового дослідження отримана оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності, яка дозволяє отримати оптимальний маршрут руху БпА з урахуванням невизначеності в появі раптового виду фізичних перешкод та небажаних подій, наявність яких необхідно було врахувати в процесі моделювання (підрозділ 2.5).

Інтеграція нейронних мереж у функціонал обох БпА надає можливість пришвидшити виконання поставленої спільної місії та уникнути аварійних і раптових непередбачуваних ситуацій.

Поєднання цих технологій формує основу для створення автономних та ефективних систем керування.

Далі буде представлено спільний алгоритм роботи безпілотного літального апарату (БпЛА) та безпілотного наземного апарату (БпНА), який враховує результати використання нейронної мережі для обробки даних щодо наявності великих статичних перешкод.

Підготовча частина загального алгоритму. Збір, підготовка та опрацювання вхідної інформації

Через центр керування (оператором) до початку запуску системи необхідно виконати:

- збір вхідної інформації;
- підготовку вхідної інформації;
- опрацювання вхідної інформації.

Вхідна інформація використана в подальших розрахунках на різних кроках прийняття рішення основної задачі: створення інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Вхідна інформація отримана з БПА щодо навколишнього середовища:

- відкритість або закритість території;
- час доби, що впливає на використання відповідного обладнання для успішного завершення місії;
- погодні умови.

Підготовча частина загального алгоритму (рисунок 2.3) складається з:

- обстеження зовнішнього середовища обома БПА з метою обробки інформації про умови виконання місії;
- здійсненні початкових технічних налаштувань обладнання БПА для конкретної місцевості.

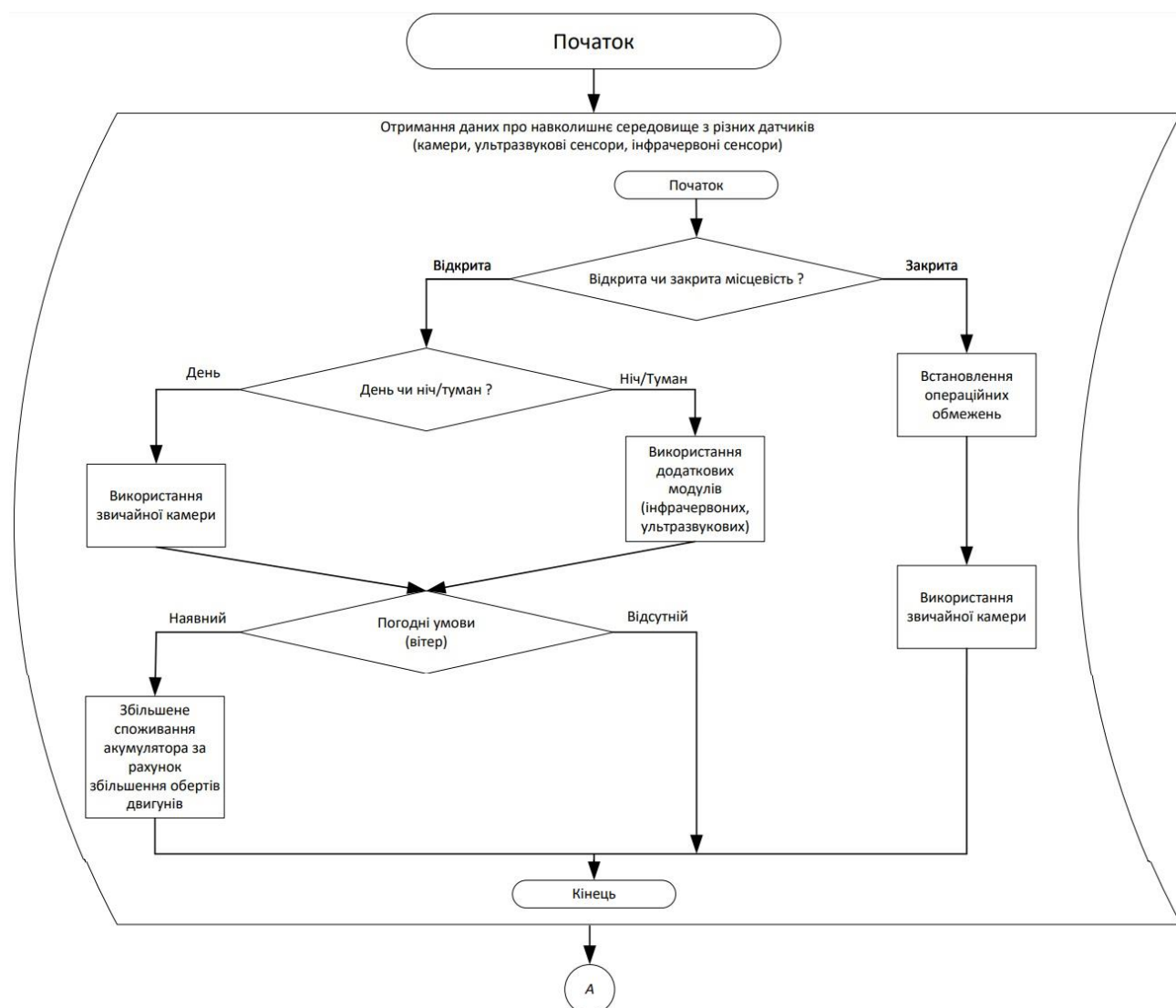


Рисунок 2.3 – Підготовча частина загального алгоритму. Отримання даних із зовнішнього середовища та збір вхідної інформації

Загальний алгоритм роботи обох БпА передбачає декілька кроків для забезпечення автономної та ефективної взаємодії з навколишнім середовищем.

Підготовча частина передбачає використання різних датчиків для збору даних про навколишнє середовище, а саме:

- камер;
- ультразвукових сенсорів;
- інфрачервоних сенсорів.

Якщо місцевість відрита, а час доби денний, використовується звичайна камера.

Якщо нічний час або наявний туман використовуються інфрачервоні та ультразвукові модулі.

Додатково враховуються погодні умови, зокрема наявність або відсутність вітру, що може впливати на споживання енергії та швидкість роботи.

Зокрема, у випадку закритої місцевості встановлюються операційні обмеження, які включають наступні:

- швидкості БпНА і БпЛА;
- висоту польоту БпЛА;
- покриття території.

Вхідна інформація містить (рисунок 2.4):

- 1) координати початкової точки А (пункт відправлення);
- 2) координати кінцевої точки В (пункт призначення);
- 3) технічні характеристики БпА, які включають:
 - покриття території;
 - швидкість пересування;
 - корисне навантаження;
 - енергоспоживання.

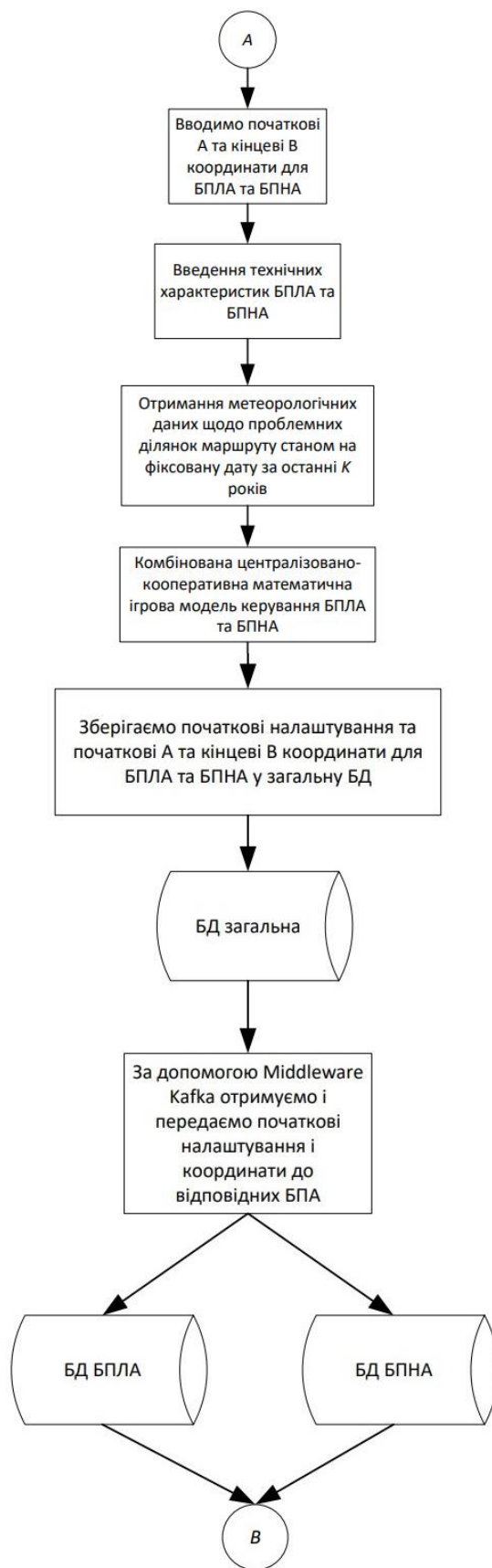


Рисунок 2.4 – Продовження підготовчої частини загального алгоритму. Збір та накопичення вхідної інформації

Зазначені вхідні дані необхідні при використанні комбінованої централізовано-кооперативної математичної ігрової моделі керування БпЛА та БпНА, застосування якої дозволяє розрахувати максимальну функцію успішності (виграшу), яку отримає коаліція БпА в процесі виконання місії.

Вхідні дані, які містять метеорологічні спостереження та висновки експертів за кожен фіксований день року за останні K років, використовуються при застосуванні оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності, а саме:

- температура повітря;
- наявність/відсутність опадів та їх інтенсивність;
- висновок експерта, в залежності від температури та інтенсивності опадів щодо стану проблемних ділянок шляху.

Виконання алгоритму проведено відповідно до комбінованої централізовано-кооперативної математичної ігрової моделі.

Разом з тим, початкові налаштування та координати точок А та В зберігаємо і передаємо у загальну базу даних (БД).

Початкові налаштування та координати отримуємо із загальної БД і відправляємо до відповідних баз даних БпЛА та БпНА, застосовуючи проміжне програмне забезпечення Middleware Kafka.

Комбінована централізовано-кооперативна математична ігрова модель керування БпЛА та БпНА міститься у продовженні підготовчої частини загального алгоритму.

Перша частина загального алгоритму

У першій частині загального алгоритму (рисунок 2.5) наведено процес взаємодії між безпілотним літальним апаратом (БпЛА) та безпілотним наземним апаратом (БпНА).

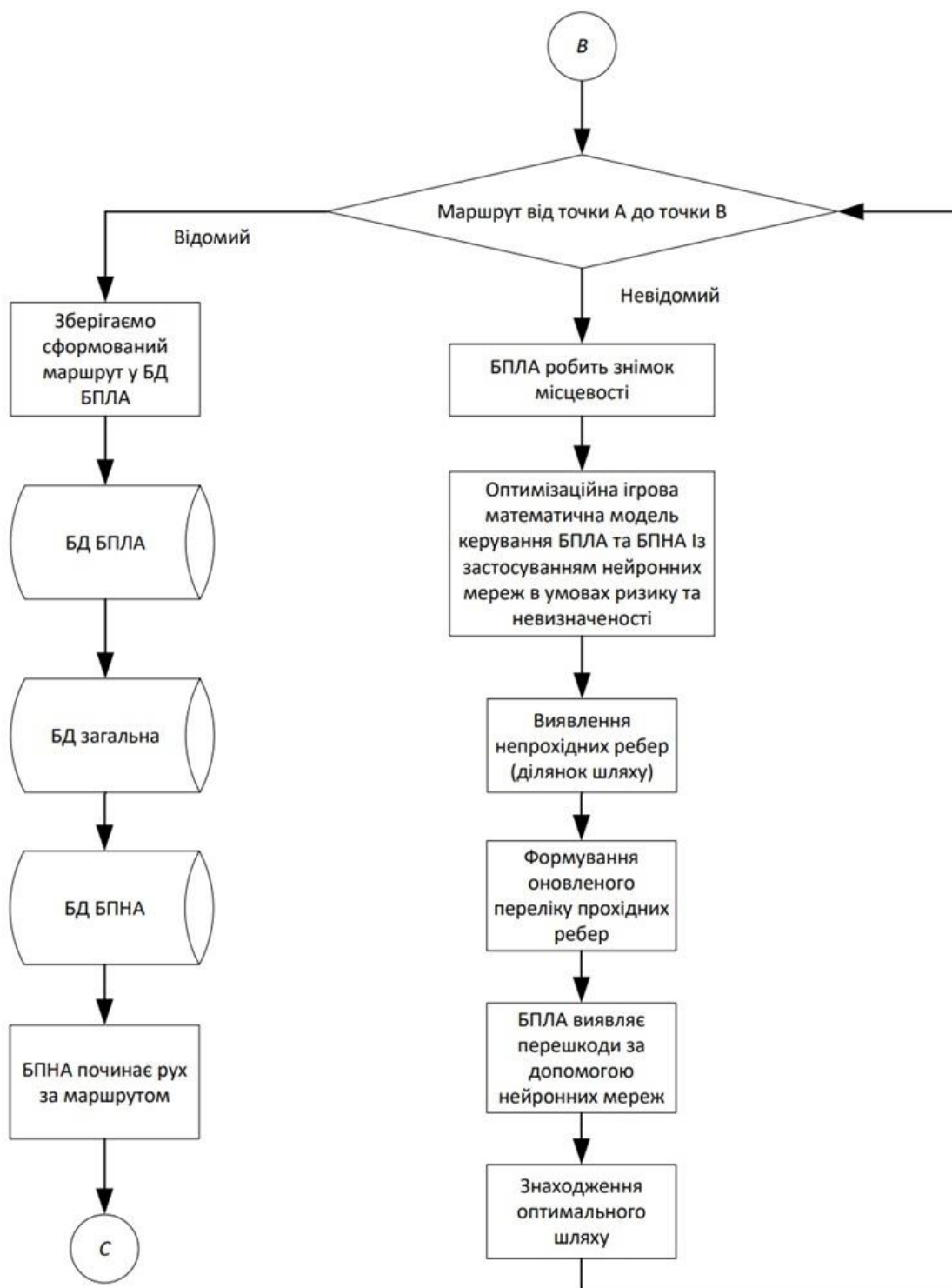


Рисунок 2.5 – Перша частина загального алгоритму. Планування маршруту від початкової точки А до кінцевої точки В

Зазначимо, що для виконання спільної місії недостатньо мати лише координати початкової точки А та кінцевої точки В. Необхідно також побудувати сам маршрут руху для БпНА, який буде успішним для виконання місії.

Відбувається перевірка наявності прокладеного маршруту від початкової точки А до кінцевої точки В, а саме: якщо маршрут невідомий, БпЛА виконує знімок місцевості з висоти за допомогою встановленої камери та сенсорів.

Оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності побудована для отримання оптимального маршруту руху БпА з урахуванням невизначеності в появі різного виду фізичних перешкод та небажаних подій, наявність яких необхідно було врахувати в процесі моделювання.

Завдяки цій моделі визначаються непрохідні ребра та формується оновлений перелік прохідних ребер.

Отримане зображення з БпЛА обробляється за допомогою навченої нейронної мережі, яка здатна виявляти перешкоди.

Знайдено оптимальний шлях з урахуванням наявності великих статичних перешкод та оновленого переліку ребер графу (виключено ребра, які з великою ймовірністю можуть бути непрохідними). Отримано оновлений варіант графу. Застосовано алгоритм A-Star.

Наступним кроком проводиться повторна перевірка наявності прокладеного маршруту руху від початкової точки А до кінцевої точки В.

Сформований маршрут зберігається у БД БпЛА.

Отриманий перелік координат передається до загальної БД за допомогою проміжного програмного забезпечення Middleware Kafka.

Перелік координат від загальної БД передається в БД БпНА шляхом використання проміжного програмного забезпечення Middleware Kafka.

Рух прокладеним маршрутом розпочинається після отримання переліку координат БпНА.

Друга частина загального алгоритму побудована для прийняття рішень в умовах невизначеності, а саме при появі раптових перешкод, невиявлених за допомогою БпЛА (рисунок 2.6).

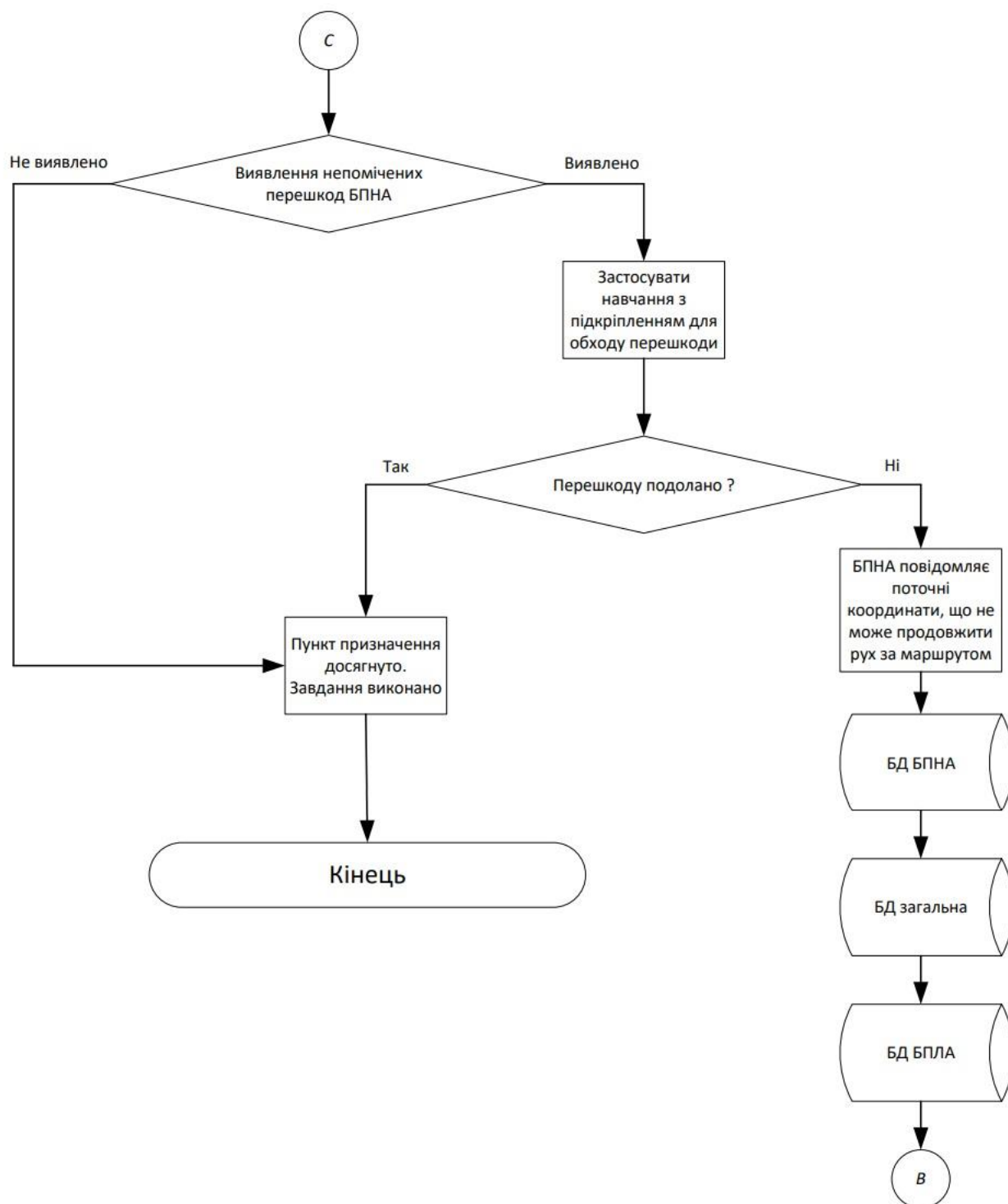


Рисунок 2.6 – Друга частина загального алгоритму. Виявлення раптових перешкод за допомогою БПНА

Маються на увазі такі перешкоди, які не були заздалегідь ідентифіковані за участю БПЛА. Наприклад, в процесі виконання місії несподівано з'явилась велика яма на дорозі [37].

Перевірка виявлення раптової перешкоди за допомогою власної камери та сенсорів проводиться БпНА.

Якщо раптової перешкоди не виявлено, пункт призначення успішно досягнуто та місія виконана.

Якщо раптову перешкоду виявлено, БпНА застосовує навчання з підкріпленням, щоб подолати раптову перешкоду.

Якщо раптову перешкоду подолано, пункт призначення успішно досягнуто та місія виконана.

Якщо ж раптову перешкоду подолати неможливо і навчання з підкріпленням не допомогло, поточні координати та маршрут руху, виявлений як неможливий, фіксуються в БД БпНА.

Повідомлення про поточні координати і подальшу неможливість продовжити рух за маршрутом передаються до загальної БД за допомогою проміжного програмного забезпечення Middleware Kafka.

Повідомлення від загальної БД до БД БпЛА надходить шляхом використання проміжного програмного забезпечення Middleware Kafka.

Після отримання повідомлення БД БпЛА від загальної БД, алгоритм прокладання маршруту повторно виконується БпЛА.

Всі кроки, отриманого в результаті наукового дослідження загального алгоритму, логічно пов'язані і забезпечують стабільну взаємодію роботи між БпЛА та БпНА.

У третьому розділі наведено результати проведених експериментів у симуляторі, які продемонстрували роботу нейронних мереж та навчання з підкріпленням для моделювання різних ситуацій при виконанні місії.

2.7 Висновки до розділу 2

У другому розділі виконаного наукового дослідження здійснено огляд, порівняння, аналіз існуючих методів, надано обґрунтування обрання апарату дослідження, який застосовано в процесі проєктування інформаційної технології

керування безпілотними апаратами (БПА); отримано нові математичні моделі та визначена архітектура інформаційної системи.

Для обрання способів побудови інформаційної технології системи керування БПА з використанням нейронних мереж та ігрових підходів надано обґрунтування доцільності (перспективності) використання кооперативного ігрового методу у керуванні БПА.

Так, саме математичні моделі типу кооперативних ігрових моделей спроможні адекватно відтворити керування групою (коаліцією) безпілотних апаратів (наземних і літальних), надати змогу розглядати основні елементи зазначеної математичної моделі в якості гравців (в цьому дослідженні – БПА), враховувати технічні обмеження та способи взаємодії БПА між собою в умовах коаліції.

Проведено також огляд і порівняльний аналіз централізованого і децентралізованого методів керування БПА, окремо виявлено переваги та недоліки кожного з методів.

Зроблено висновок, що централізований метод керування коаліцією БПА у порівнянні з децентралізованим має такі переваги:

- контроль: весь контроль за виконанням місії (місій) покладається на одного оператора;
- простота: відсутність запитів щодо розробки складних алгоритмів автономії;
- безпека: оператор приймає всі рішення одноосібно.

Враховуючи переваги централізованого методу над децентралізованим, в науковому дослідженні обрано централізований метод керування БПА.

Комбінована централізовано-кооперативна математична ігрова модель керування БПА побудована в процесі наукового дослідження.

В процесі побудови цієї моделі використано підходи і апарат ігрового методу. В якості загальної функції виграшу обрано функцію, яка визначає головну ціль

гри – успішне виконання місії від її початку і до кінця. Результатом моделювання є знаходження максимуму загальної функції виграшу.

$$F_{\text{загальна}} = \sum_{j=1}^3 a_j F_j - \sum_{j=4}^5 a_j F_j \rightarrow \max,$$

де F_j – складові загальної функції виграшу; $j = \overline{1,5}$;

a_j – пріоритет F_j , тобто j -ї складової функції виграшу; $j = \overline{1,5}$, де $j = 1$ – покриття, $j = 2$ – швидкість пересування, $j = 3$ – корисне навантаження, $j = 4$ – енергоспоживання, $j = 5$ – фінансові ресурси.

Експериментальне порівняння $F_{\text{загальна}}$ результатів оптимізації функцій успішності, отриманої комбінованої централізовано-кооперативної математичної ігрової моделі, з однієї сторони з відповідними результатами функцій успішності ідеальної та децентралізованої моделі, з іншої сторони, дає змогу зробити висновок про значно більшу ефективність функції виграшу отриманої моделі. Так, ефективність комбінованої централізовано-кооперативної математичної ігрової моделі у порівнянні з ідеальною становить 71,12%. Ефективність децентралізовано-кооперативної моделі становить від 22,66% до 65% у порівнянні з ідеальною.

Під ідеальною розглядалася модель, де за замовчуванням енергетичне споживання та фінансові витрати дорівнюють нулю.

Практичне застосування комбінованої централізовано-кооперативної математичної ігрової моделі дозволить при виконанні місії двома БПА (наземним та літальним):

- 1) обрати ігровий підхід;
- 2) обрати централізовано-кооперативний метод керування БПА;
- 3) з метою найефективнішого виконання місії визначитися з вибором найбільш підходящих технічних характеристик коаліції БПЛА та БПНА.

Розроблена оптимізаційна ігрова математична модель керування БПЛА та БПНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

В науковому дослідженні детально розглянуто підхід до вирішення проблеми обмеженості та/або неточності вхідних даних, відповідно до якого розглядаються два різних види ситуацій, які зазвичай необхідно дослідити, а саме: 1 – рішення приймаються в умовах ризику; 2 – рішення приймаються в умовах невизначеності.

В залежності від характеру (типу) невизначеності залежить вибір апарату дослідження.

Якщо рішення приймаються в умовах ризику, то появу небажаної перешкоди моделюють із застосуванням апарату теорії ймовірностей.

Якщо рішення приймається в умовах невизначеності, функцію розподілу ймовірностей знайти практично неможливо.

В науковому дослідженні розглядаються обидві ситуації, які виникають при моделюванні керування БпЛА та БпНА: 1 – ризик появи небажаного явища, зокрема весняного/осіннього бездоріжжя; 2 – невизначеність – поява раптової перешкоди (ями) в процесі виконання місії.

Побудова оптимального маршруту з урахуванням ризику відбувається при застосуванні першої частини моделі, ще до початку місії. Із переліку всіх можливих ребер графу (ділянок шляху) вилючаються ті ребра, які мають ймовірність непрохідності більшу, або рівну деякій наперед заданій константі C , де $0 \leq C \leq 1; C = const$.

Функція розподілу ймовірностей для кожного i -го ребра визначається експериментальним шляхом, з використанням метеорологічних даних за визначений період (наприклад, за 10 останніх років). Під метеорологічними даними розуміємо температуру повітря та інтенсивність опадів.

Результат застосування першої частини оптимізаційної ігрової математичної моделі – отримання шляху з виключеними проблемними ребрами (ділянками).

Друга частина моделі вирішує проблему подолання раптової перешкоди (невизначеності) на шляху в процесі виконання місії. Запропоновано вирішувати цю проблему за допомогою можливостей нейронних мереж та навчання з підкріпленням.

Додаткова перевірка щодо виявлення раптової перешкоди за допомогою камери та сенсорів проводиться БпНА.

Якщо раптової перешкоди не виявлено, пункт призначення успішно досягнуто та місія виконана.

Якщо раптову перешкоду виявлено, БпНА застосовує навчання з підкріпленням для подолання раптової перешкоди.

У разі, коли раптову перешкоду подолати неможливо і навчання з підкріпленням не допомогло, поточні координати та маршрут руху, виявлений як неможливий, фіксуються в базі даних БпНА та передаються до бази даних БпЛА. У цьому разі алгоритм прокладання маршруту виконується БпЛА повторно.

Проведено аналіз ефективності оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

За замовчуванням вважалося, що ефективність ідеальної моделі становила 100%. За проведеними розрахунками отримано, що ефективність моделі, яка не враховувала ризику та невизначеність, становила всього 16,1%; ефективність моделі, яка враховувала тільки ризики, становила 39,9%; ефективність моделі, яка враховувала і ризики, і невизначеність, становила 73,2%.

Отже, найбільшу ефективність з розглянутих реальних моделей має модель, отримана в результаті наукового дослідження, яка враховувала і ризики, і невизначеність.

3 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ У КЕРУВАННІ БЕЗПЛОТНИМИ АПАРАТАМИ

3.1 Аналіз можливостей сучасних симуляторів. Вибір симулятора для проведення експериментів наукового дослідження

При виконанні експериментальної перевірки отриманих результатів наукового дослідження проведені математичні моделювання, які передбачають використання симулятора, який здатний підтримувати керування різними типами безпілотних апаратів (БпЛА та БпНА).

Розглянуто найпоширеніші симулятори (Ardupilot, PX4, ROS, Gazebo, Microsoft AirSim), а також проаналізовані їхні переваги та недоліки.

Ardupilot: універсальний симулятор з відкритим кодом, який підтримує управління як літальними, так і наземними безпілотними апаратами. Ця система використовується для розробки, тестування та керування БпА, роботами та іншими автономними пристроями у реальному часі [47].

PX4: симулятор з відкритим кодом, що підтримує автономне управління БпА, використовується в процесі створення як простих, так і складних автономних БпА [47–49].

ROS: програмна платформа для розробки роботизованих систем. Модульна архітектура симулятора дозволяє проводити експерименти як з наземними, так і з літальними БпА [50].

Gazebo: симулятор, який використовується для створення реалістичних тривимірних середовищ, а також є ефективним для тестування віртуальних безпілотних апаратів [51].

Microsoft AirSim: найбільш функціональний симулятор, дозволяє проводити експерименти у різноманітних середовищах. Цей симулятор підтримує достатньо багато додаткових сучасних бібліотек, що є однією з позитивних його властивостей [52]. Основною перевагою Microsoft AirSim є реалістична графіка, імітація

фізичних явищ та можливість додавати власні об'єкти, у тому числі перешкоди та додаткові типи БпА.

Результати більш детального огляду запропонованих симуляторів наведені нижче.

ArduPilot

Використання ArduPilot дозволяє створити віртуальні образи БпА різних типів. Особливістю цього симулятора є наявність різноманітного набору програмних інструментів.

ArduPilot – це проєкт з відкритим вихідним кодом, який представляє собою вдосконалену, повнофункціональну та надійну програмну систему автопілота. Цей симулятор здатний керувати практично будь-якою транспортною системою: літаками, планерами, мультикоптерами, гелікоптерами, підводними човнами та наземними транспортними засобами. Типи транспортних засобів, які підтримуються симулятором ArduPilot, можуть доповнюватися (рисунок 3.1).

Крім того, ArduPilot ефективно використовується для проведення симуляцій великих роїв БпЛА та БпНА.

ArduPilot застосовується для проведення тестування та проєктування нових розробок великими установами та корпораціями, такими як NASA, Intel та Boeing, а також університетами [47].

Завдяки своїм можливостям симулятор ArduPilot набув великий попит у застосуванні в різних галузях, насамперед, дослідницьких проєктах, промисловості та сільському господарстві.

Окремо розглянуто переваги та недоліки симулятора ArduPilot.

Переваги симулятора ArduPilot:

- наявність різноманітних апаратних платформ (літальних, наземних, підводних апаратів). Передбачено право обрати таку апаратну платформу, яка найкраще моделює конкретну ситуацію;

– підтримка налаштувань під індивідуальні запити користувачів. Потенційні можливості застосування симулятора ArduPilot за необхідності збільшуються завдяки розробці та використанню додаткових модулів;

– наявність користувальницької документації та підтримка з боку розробників;

– професійне обговорення на форумах, в яких беруть участь користувачі та розробники.

Недоліки симулятора ArduPilot:

– складні налаштування симулятора;

– журнал виконаних дій потребує доопрацювання;

– проблема розробки, сумісності та інтеграції нестандартних програмних модулів сповільнює поширення цього симулятора.

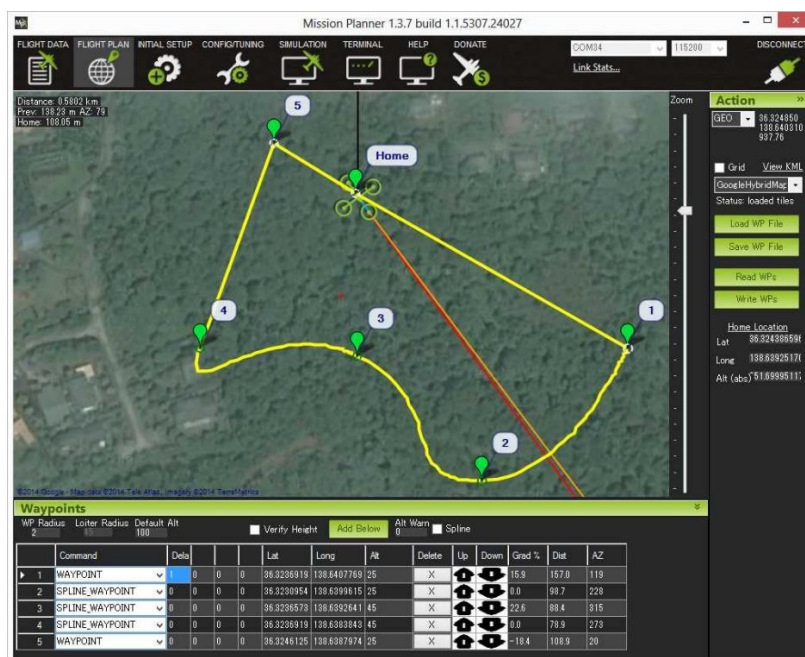


Рисунок 3.1 – Вигляд роботи вікна симулятора Ardupilot [47]

PX4 Autopilot

PX4 Autopilot – симулятор, який використовується в багатьох проєктах різного напрямлення (рисунок 3.2).

Головною позитивною рисою цього симулятора є відкритість спеціалізованого програмного забезпечення. Зазначимо, що такий тип програмного

забезпечення надає можливість запрограмувати контролери БПА, а також здійснювати симуляції до етапу їх практичного впровадження [48].

Цей симулятор активно застосовується в проєктах різного напрямку.

Переваги симулятора PX4 Autopilot:

- надійність в роботі та стабільність програмного забезпечення;
- передбачено взаємодію з наземними станціями, що, в свою чергу, спрощує управління БПА;
- оперує широким набором сенсорів та апаратних модулів, що розширює можливості експлуатації різних типів БПА;
- передбачено підтримку різних безпечних режимів повернення;
- висока безпека та надійність програмного коду – одна з головних переваг симулятора, що досягається завдяки постійним тестуванням апаратного та програмного забезпечення.

Недоліки симулятора PX4 Autopilot:

- при моделюванні літальних та наземних апаратів з'являються проблеми, які не притаманні іншим симуляторам, зокрема при симуляції розряду батареї [49];
- ефективне використання системи потребує визначення технічних характеристик апаратів, які необхідно спроектувати.

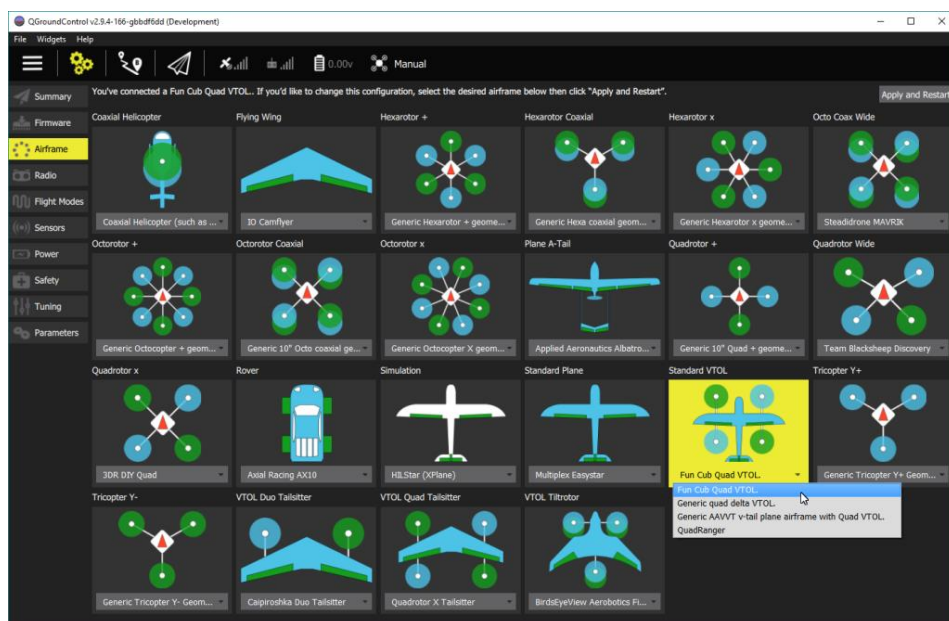


Рисунок 3.2 – Вигляд роботи вікна симулятора PX4 Autopilot [48]

Robot Operating System (ROS)

Robot Operating System (ROS) – це набір бібліотек спеціалізованого програмного забезпечення (рисунок 3.3).

Симулятор Robot Operating System (ROS) відриває можливості написання програм для роботів, зокрема, драйверів для актуальних алгоритмів.

Симулятор Robot Operating System має відкритий код.

ROS ефективно застосовуються в процесах проєктування систем автономної навігації, а також для визначення оптимальних маршрутів при наявності небезпеки [50].

Переваги симулятора Robot Operating System:

- модульна структура бібліотеки дозволяє поєднувати прості компоненти для створення складних систем;
- вирішення різноманітних задач, від оптимального планування маршруту до обробки вхідних даних, які стосуються характеристики місцевості;
- надання відкритого доступу до програмних бібліотек, у тому числі тих, які підтримуються сторонніми розробниками.

Недоліки симулятора Robot Operating System:

- складність у встановленні, налаштуванні та експлуатації симулятора;
- для проведення складних симуляцій з використанням ROS передбачається наявність високопродуктивної апаратної бази.

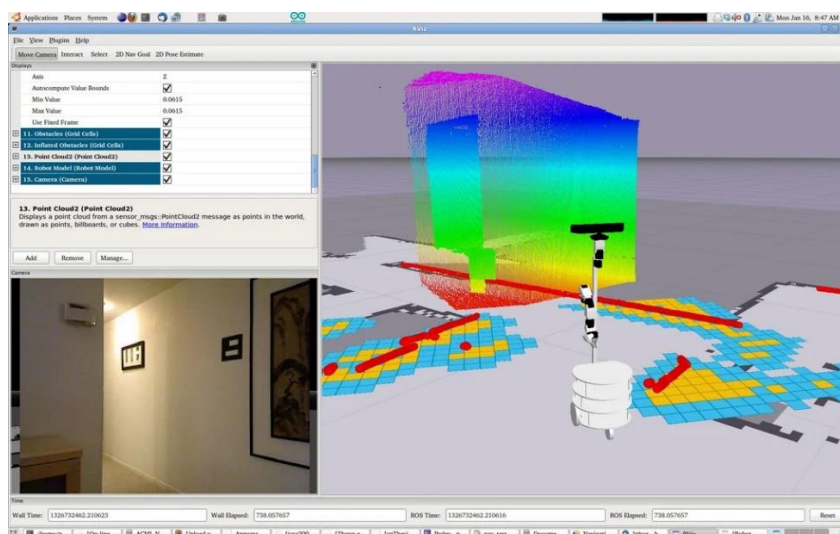


Рисунок 3.3 – Вигляд роботи вікна симулятора ROS (Robot Operating System) [50]

Gazebo

Gazebo – це тривимірний динамічний симулятор.

Gazebo імітує поведінку роботів як у внутрішніх, так і в зовнішніх середовищах (рисунок 3.4). Симулятор відкриває можливості моделювати фізичні процеси достатньо точно завдяки наборам програмних датчиків та інтерфейсів для користувачів. Використання бібліотеки Gazebo Physics обмежене областю фізичних процесів [51].

Переваги симулятора Gazebo:

- моделювання фізичних процесів відбувається достатньо точно, включаючи наступні фізичні явища: тертя, освітлення та сенсорні дані;
- забезпечує підтримку різних типів роботів, у тому числі, літальних та наземних БПА.

Недоліки симулятора Gazebo:

- практичне використання симулятора потребує наявності комп'ютера з високою продуктивністю, оскільки це необхідно для отримання графічних зображень з високою якістю та проведення великої кількості розрахунків;
- значні витрати часу та наявність вузькоспеціальних знань для успішного моделювання складних систем.

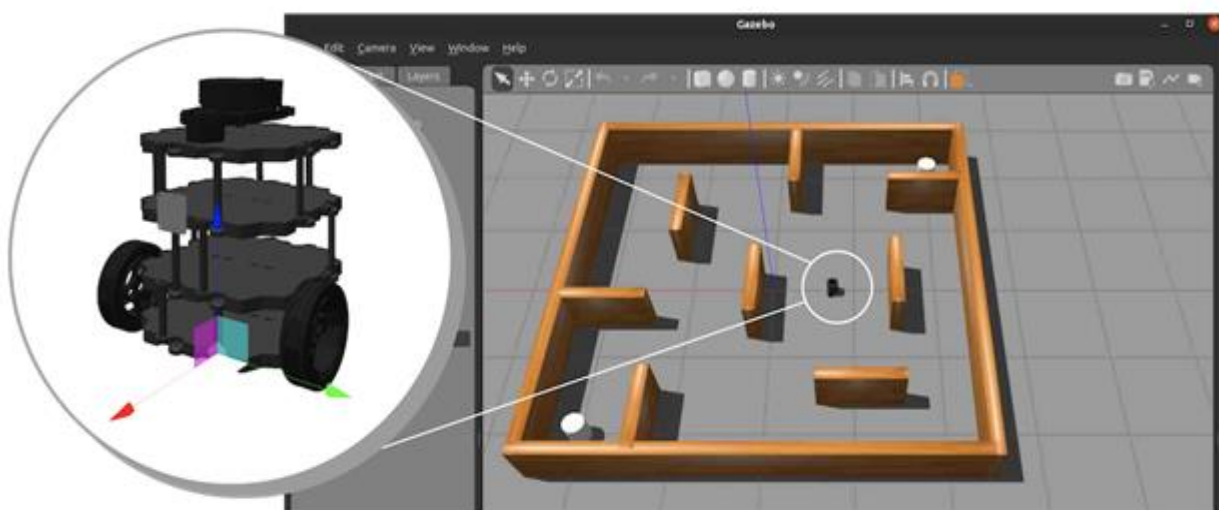


Рисунок 3.4 – Вигляд роботи вікна симулятора Gazebo [51]

Microsoft AirSim

Microsoft AirSim симулятор був створений у 2017 році дослідницьким підрозділом Microsoft [52].

В першу чергу, даний симулятор використовувався для проведення досліджень із застосуванням штучного інтелекту (ШІ), зокрема моделюванні руху.

Однією з позитивних рис симулятора є створення (використання) реалістичного віртуального середовища з метою перевірки алгоритмів автономного керування. Тестувальне середовище включає симуляцію польотів БПЛА, пересування БПНА, інтеграцію сенсорів та можливість використання нейронних мереж.

Наразі сторонні розробники здійснюють підтримку програмного забезпечення даного симулятора, оскільки офіційна підтримка Microsoft AirSim припинена у 2023 році. Користувачі продовжують використання Microsoft AirSim у своїх проєктах та змінюють його під власні задачі.

Microsoft AirSim використовується для тренування нейронних мереж, щоб розпізнавати перешкоди, а також у навчанні з підкріпленням для коректного слідування за визначеним маршрутом.

Крім того, у Microsoft AirSim передбачено відкритий API, завдяки якому дозволено виконувати зовнішній код на мові програмування Python.

Перейдемо до розгляду можливостей симулятора Microsoft AirSim більш детально.

Microsoft AirSim працює на базі ігрового рушія Unreal Engine.

Ігровий рушій Unreal Engine дозволяє створювати різноманітні віртуальні середовища, наближені до реальних та фізичні процеси.

Серед основних можливостей симулятора є:

- віртуальна реалізація фізичних процесів польоту чи пересування БПА;
- можливість застосування різних програмних бібліотек, які дозволяють моделювати поведінку, у тому числі, LiDAR сенсору, камери, GPS, ультразвукових та інфрачервоних сенсорів;

- наявність спеціалізованих внутрішніх підпрограм, які дозволяють збирати дані для навчання нейронних мереж;
- передбачена можливість для симуляції погодних умов, різноманітних перешкод та інших можливих ситуацій.

Microsoft AirSim використовується для моделювання ігрових методів, які передбачають участь декількох БпА (гравців). Кооперативний ігровий метод дозволяє моделювати ситуації, де БпА (гравці) можуть мати індивідуальну чи спільну цільову функцію для досягнення загальної мети.

Остаточний вибір програмного забезпечення для симуляції залежить від особливостей і цілей проекту.

Симулятор Microsoft AirSim використовується для моделювання поведінки БпА у міському середовищі з перешкодами, такими як будівлі, транспорт, несприятливі погодні умови (дощ, туман, вітер). Він ефективно застосовується для тестування автономної навігації та обробки зображень з використанням нейронної мережі (рисунок 3.5).

Переваги симулятора Microsoft AirSim:

- містить програмні модулі та дозволяє розширити вибір віртуальних середовищ, які якнайкраще імітують реальність;
- наявність можливості використовувати різноманітні віртуальні середовища;
- передбачено навчання нейронних мереж за допомогою внутрішніх підпрограм для збору даних;
- використовується для комп'ютерного моделювання поведінкою БпЛА, БпНА.

Недоліки симулятора Microsoft AirSim:

- потрібен високопродуктивний комп'ютер для роботи з деталізованими середовищами;
- офіційна підтримка призупинена у 2023 році.

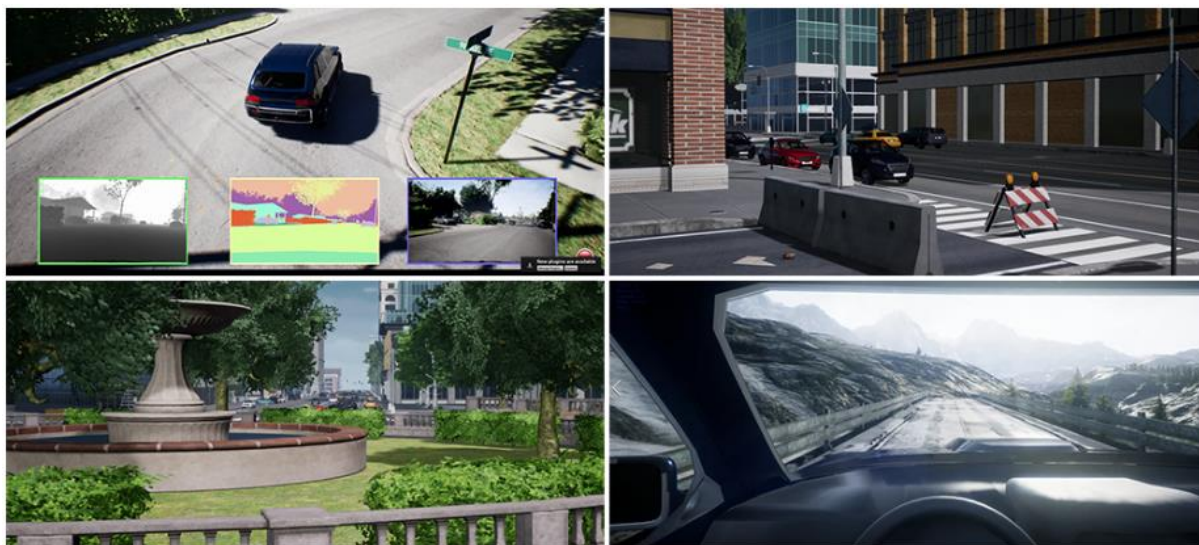


Рисунок 3.5 – Вигляд роботи можливостей симулятора Microsoft AirSim [52]

3.2 Впровадження симуляції

Симуляція ігрового методу управління БПА з використанням нейронних мереж є комплексним завданням.

Впровадження симуляції здійснено з використанням симулятора Microsoft AirSim, який завантажено та встановлено Microsoft AirSim з офіційного репозиторію GitHub. Обрано версію Unreal Engine 4.27 для збірки та запуску Microsoft AirSim. Зазначимо, що перехід на оновлену версію Unreal Engine 5 є недоцільним, оскільки з використанням нової версії залучається багато ресурсів, які можна задіяти для навчання нейронних мереж при розпізнаванні зображень.

Перейдемо до розгляду архітектури роботи симулятора Microsoft AirSim (рисунок 3.6).

Microsoft AirSim підтримує два типи стандартних БПА: наземний та літальний. Наземним БПА будемо вважати автомобіль, а літальним – квадрокоптер.

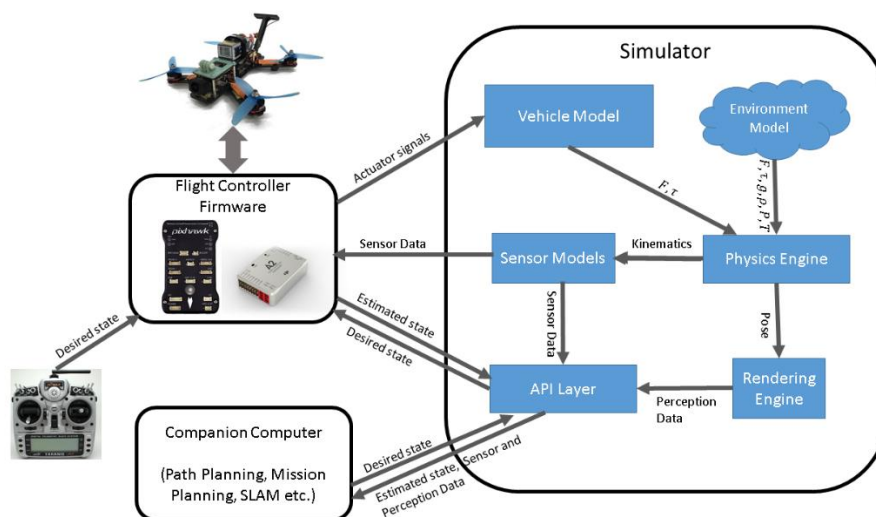


Рисунок 3.6 – Архітектура роботи симулятора Microsoft AirSim для реального БпА [52]

На рисунку 3.6 зображено архітектуру роботи симулятора Microsoft AirSim, що використовується для моделювання поведінки безпілотних апаратів (БпА). Ця архітектура представляє основні компоненти Microsoft AirSim та їх взаємодію у процесі симуляції.

Надамо огляд елементів архітектури роботи симулятора Microsoft AirSim.

Flight Controller Firmware:

Центральний контролер відповідає за керування БпА. Він представляє собою окремий пристрій, який містить спеціалізовану прошивку для керування БпА. Прикладом такого контролера є Pixhawk, який забезпечує виконання команд, контроль стабільності БпА та обробку сигналів.

Companion Computer:

Додатковий обчислювальний мікрокомп'ютер відповідає за виконання складних обчислень, зокрема, планування маршруту, локалізацію і картографування середовища (SLAM), а також обробку даних з сенсорів.

Simulator

Основна частина Microsoft AirSim, яка відповідає за симуляцію фізичних явищ, навколишнього середовища та роботу сенсорів. Simulator складається з кількох ключових модулів:

1. **Vehicle Model:** модель безпілотного апарату (БпЛА чи БпНА) відтворює фізичні характеристики цих моделей, а саме: масу, інерцію, коефіцієнти лінійного та кутового опору, коефіцієнти тертя.

2. **Environment Model:** модель зовнішнього середовища, яка враховує вплив таких факторів, як гравітація, вітер та інші природні явища.

3. **Physics Engine:** здійснює розрахунки, які стосуються фізичних процесів, зокрема, кінематики, динаміки руху та тертя.

4. **Rendering Engine:** здійснює графічне відображення середовищ.

5. **Sensor Models:** моделювання роботи різноманітних сенсорів, таких як камери, GPS, LiDAR тощо, для збору даних у симуляції.

6. **API Layer:** відповідає за посередництво між симулятором і зовнішнім програмним забезпеченням; забезпечує відправлення команд для керування БпА через API, отримання даних з сенсорів і аналіз стану апаратів.

7. **Remote Controller:** фізичний або віртуальний пульт управління, який дозволяє безпосередньо керувати БпА, коли це необхідно.

Розглянемо взаємодію між компонентами Microsoft AirSim:

1. **Actuator signals:** сигнали управління, які передають команди на фізичну або віртуальну модель БпА через прошивку контролера.

2. **Sensor Data:** отримані дані від сенсорів (камер, LiDAR, інфрачервоні та ультразвукові), передаються до контролера чи мікрокомп'ютера для подальшого аналізу.

3. **Desired state:** бажаний стан, який передається від центру керування до симулятора.

4. **Estimated state:** фактичний стан, який передається від симулятора до центру керування.

Внесення змін до архітектури роботи симулятора Microsoft AirSim

Проведення експериментів наукового дослідження із застосуванням ігрового підходу та нейронних мереж у керуванні БпА вимагало внести зміни до архітектури симулятора Microsoft AirSim.

З метою модифікації архітектури, внесено декілька змін, які показані на рисунку 3.7.

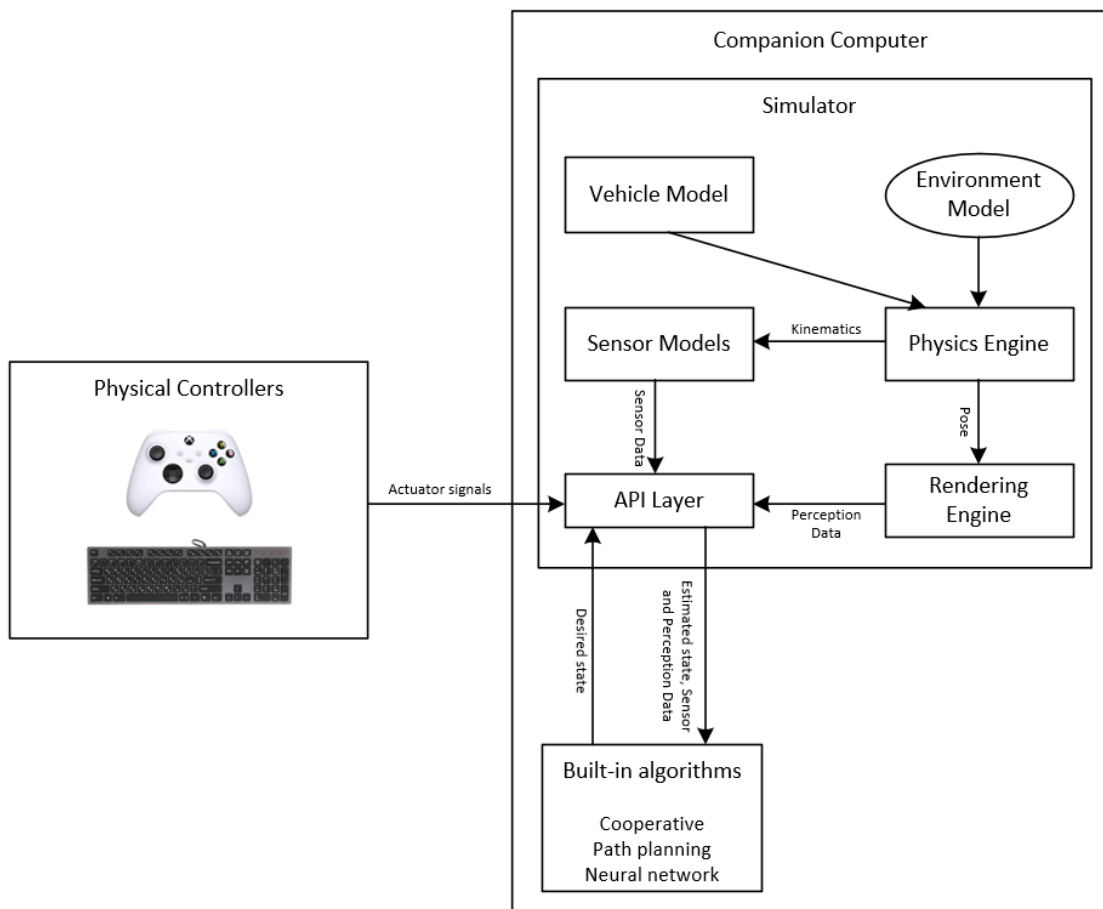


Рисунок 3.7 – Модифікована архітектура роботи симулятора Microsoft AirSim для віртуального БПА

Запропонована в науковому дослідженні модифікована архітектура роботи симулятора Microsoft AirSim включає наступні зміни:

1. Прибрано центральний контролер (Flight Controller Firmware) та окремий обчислювальний мікрокомп'ютер (Companion Computer), які не використовувались при проведенні експериментів.

2. Додано нові фізичні контролери (Physical Controllers):

а) додано блок з фізичними пристроями керування, такими як джойстик та клавіатура (на вибір), які необхідні для проведення експериментів;

б) фізичні пристрої підключені до симулятора через Actuator signals, забезпечують безпосередній контроль дій апарата за допомогою API Layer.

3. Додано блок вбудованих алгоритмів (Built-in algorithms):

а) інтегровано окремий модуль для ігрових математичних моделей, які включають:

- централізовано-кооперативне управління (cooperative control);
- планування маршруту в умовах ризику та невизначеності (path planning);
- нейронні мережі (neural network).

б) блок вбудованих алгоритмів робить розрахунки і передає їх результати на симулятор до API Layer. Симулятор повертає результати розрахунків оператору і відображає їх на Companion Computer.

4. Взаємодія з API Layer:

а) API Layer працює як посередник між симулятором і вбудованими алгоритмами. Така взаємодія дозволяє автоматизувати процес управління без потреби втручання оператора за допомогою фізичних контролерів.

5. Збережено симуляційний модуль:

а) блоки Vehicle Model, Environment Model, Sensor Models, Physics Engine та Rendering Engine залишилися без змін, але вони отримують і обробляють команди не тільки від фізичних контролерів, але й від блоку вбудованих алгоритмів.

3.3 Моделювання та впровадження симуляційного середовища

Для проведення експериментів наукового дослідження запропоновано використання двох типів БпА (наземного та літального).

Перший тип БпА – наземний, має вигляд автомобіля, далі БпНА. В бібліотеці для Python БпНА представлений у вигляді програмного модуля CarClient для використання наземних транспортних засобів у віртуальному симуляційному середовищі. Цей модуль дозволяє передавати команди до автомобіля, а також застосовувати сенсори, камери, LiDAR, GPS та інші. CarClient надає можливість

точного управління, дозволяючи тестувати алгоритми автономного керування, у тому числі планування маршруту, розпізнавання об'єктів та обхід перешкод.

Завдяки Microsoft AirSim API оператор має можливість напряду керувати БпНА із застосуванням спеціалізованої програми на Python або C++, виконуючи команди для прискорення, гальмування чи поворотів.

Завдяки реалістичному ігровому рушію, CarClient (рисунок 3.8) дозволяє оцінювати вплив дорожніх умов на поведінку автомобіля, у тому числі тертя чи нахилу поверхні.



Рисунок 3.8 – CarClient в Microsoft AirSim

Другий тип БпА – літальний. MultiRotorClient (з Microsoft AirSim API) – це програмний модуль, спеціально створений для моделювання роботи багатороторних безпілотних літальних апаратів (далі БпЛА), таких як квадрокоптери. Особливістю цього типу БпА є здатність зупинятись в одній точці та виконувати різноманітні маневри. Зазначене робить його зручним при проведенні досліджень і експериментів у сфері автономного керування. MultiRotorClient надає можливість взаємодіяти з віртуальним БпЛА через API для виконання польотних завдань, таких як зліт, приземлення, навігація за маршрутом та виявлення перешкод.

Програмний модуль MultiRotorClient забезпечує доступ до даних сенсорів, таких як камери, LiDAR, GPS і IMU, що дозволяє розробляти та тестувати алгоритми для виявлення об'єктів, побудови карт місцевості та орієнтації в

просторі. Завдяки високій точності моделювання польоту, MultiRotorClient можна використовувати для імітації різних сценаріїв, у тому числі для польотів у міських умовах, серед густої рослинності чи при несприятливих погодних умовах.

MultiRotorClient (рисунок 3.9) відкриває можливості для реалізації кооперативних ігрових методів, моделюючи роботу двох чи більше БПА, які взаємодіють один з одним у реальному часі. Цей програмний модуль використовується для інтеграції з ігровими методами і дозволяє тестувати сценарії, де кілька БПА виконують завдання як коаліція, наприклад, спільний моніторинг або пошук об'єктів. Крім того, модуль підходить для моделювання екстрених ситуацій, наприклад, пошуково-рятувальних операцій, де БПА повинні діяти автономно в умовах обмеженої видимості чи втрати зв'язку.



Рисунок 3.9 – MultiRotorClient в Microsoft AirSim

Додатково Microsoft AirSim має підтримку переключення з TPV (Third-Person View) до FPV (First-Person View), що дозволяє краще оцінити роботу симулятора.

Ігровий рушій Unreal Engine надає змогу використовувати різноманітні середовища для їх подальшого застосування в дослідженнях та експериментах у симуляторі Microsoft AirSim.

Міське середовище в Microsoft AirSim містить висотні будівлі, дороги, пішохідні зони, світлофори та інші автономні транспортні засоби (рисунок 3.10). Міська забудова використовується для тестування алгоритмів руху БПА в умовах

великої кількості об'єктів. Наявна можливість обрання різних погодних умов (дощ, туман, ніч), що дозволяє моделювати ситуації з обмеженою дальністю видимості. У цьому середовищі також легко тестувати кооперативні ігрові методи, де кілька БпА працюють разом у складних умовах.



Рисунок 3.10 – Демонстрація міського середовища у симуляторі Microsoft AirSim

Гірська місцевість у Microsoft AirSim має складні рельєфи з крутими схилами, ущелинами та бездоріжжям (рисунок 3.11). Така місцевість підходить для тестування БпА, які виконують завдання пошуку та порятунку у важкодоступних місцях, наприклад, у горах. Рельєфність місцевості створює труднощі для планування маршруту та навігації, оскільки потрібно враховувати висоту, вітер та ризик зіткнення. У таких умовах особливо важливо використовувати сенсори, такі як LiDAR і камери, для картографування місцевості в реальному часі. Також ця місцевість дозволяє моделювати ситуації, коли БпА кооперуються для забезпечення точності й ефективності у спільних пошуково-рятувальних місіях.



Рисунок 3.11 – Демонстрація гірської місцевості у симуляторі Microsoft AirSim

Африканська савана у симуляторі Microsoft AirSim представлена відкритими рівнинами, густими лісами, водоймами та дорогами [53]. Ця дика місцевість містить детальні тривимірні моделі флори і фауни, які використовуються для тестування алгоритмів БПА (рисунок 3.12). Крім того, віртуальна місцевість африканської савани надає можливість тестувати алгоритми оптимізації маршрутів у масштабних місіях. У кооперативних сценаріях кілька БПА можуть працювати разом, щоб одночасно охопити великі території.



Рисунок 3.12 – Демонстрація африканської савани у симуляторі Microsoft AirSim

Лісова місцевість у Microsoft AirSim складається з густої рослинності, дерев різної висоти, кущів та іншої фауни (рисунок 3.13). Віртуальний ліс надає умови тестування БПА, які використовуються для моніторингу дикої природи, оцінки стану лісів чи виявлення стихійних лих. Сенсори, такі як камери та LiDAR, використовуються для навігації між деревами, картографуванні території та глибокого аналізу навколишнього середовища. Насиченість перешкодами робить це середовище викликом для розробки алгоритмів маршруту руху та точного позиціонування. Лісова місцевість підходить для навчання нейронних мереж, які аналізують дані в умовах низької освітленості чи несприятливих погодних умов. У кооперативних ігрових методах кілька БПА можуть використовуватися для пошуково-рятувальних операцій та доставки вантажів у важкодоступні місця.



Рисунок 3.13 – Демонстрація лісової місцевості у симуляторі Microsoft AirSim

3.4 Актуальні способи навчання та налаштування нейронних мереж у рамках інформаційних технологій

Огляд методів розпізнавання об'єктів

На сьогоднішній день існує багато методів розпізнавання об'єктів. Швидкий розвиток графічних прискорювачів дозволив ефективно застосовувати згорткові нейронні мережі (ЗНМ) [17].

На основі ЗНМ існує декілька технологій для розпізнавання об'єктів, які вважаються найактуальнішими, які представлені нижче.

Перша технологія, яку розглянемо, називається YOLO («ти тільки дивишся раз»). Ця згорткова нейронна мережа виконує обробку зображення лише за один прохід (відсутність повторної обробки зображення), що дозволяє досягти високої швидкості виявлення об'єктів в реальному часі [37, 54].

Вхідне зображення проходить через декілька шарів згортки, які визначають ключові ознаки (контури, текстури, колір).

Кожен пройдений шар зменшує роздільну здатність зображення, але збільшує кількість фільтрів для кращого розпізнавання ознак об'єкта.

Ознаки об'єкта передаються і зберігаються у блоці, який обчислює ймовірності приналежності цих об'єктів до певного класу. Цей блок також проводить обчислення координат обмежувальних рамок (bounding boxes).

В процесі виконання наукового дослідження використана технологія Single Shot MultiBox Detector (SSD).

Друга технологія – SSD використовується для розпізнавання об'єктів на зображенні.

Ця технологія заснована на згортковій нейронній мережі прямого зв'язку. Подібно до технології YOLO, розпізнавання об'єктів відбуваються за один прохід (повторна обробка зображення відсутня) із застосуванням нейронної мережі. Але SSD додатково враховує багат шарові ознаки об'єктів [37, 54].

Зазначимо, що багат шарові ознаки означають, що нейронна мережа аналізує зображення на різних рівнях деталізації (шарів ознак), кожен з яких має різну роздільну здатність і спроможність розпізнавати об'єкти різних розмірів [55].

Формула $L(x, c, l, g)$ у статті [55] визначає функцію втрат для моделі Single Shot MultiBox Detector (SSD). Вона використовується для навчання нейронної мережі і складається з двох основних частин:

$$L(x, c, l, g) = \frac{1}{N} \left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g) \right), \quad (3.1)$$

де $L(x, c, l, g)$ – загальна функція втрат;

N – кількість об'єктів виявлених нейронною мережею, якщо $N = 0$, втрати прирівнюються до нуля [55];

x – індикатор відповідності отриманого зображення об'єкту реальному об'єкту;

c – набір наперед визначених ймовірностей, відповідно до яких, об'єкт належить певному класу;

l – отримані координати рамок в результаті роботи нейронної мережі;

g – реальні (фізичні) координати рамок, які містять об'єкт. Вони представляють реальні значення розташування об'єкта (анотації);

$L_{conf}(x, c)$ – функція втрат характеризує помилку між отриманими та фактичними класами об'єктів [55];

$L_{loc}(x, l, g)$ – функція локалізації характеризує помилку між отриманими та фактичними координатами рамок виявленого об'єкта [55];

α – ваговий коефіцієнт, що визначає баланс між значеннями функції втрат та функції локалізації (часто дорівнює 1).

Особливістю використання методу SSD є застосування карти ознак.

При цьому SSD використовує кілька карт ознак різних масштабів для виявлення як малих, так і великих об'єктів. Нейронна мережа будує одночасно декілька рамок та оцінює ймовірності знаходження фіксованого об'єкту всередині цих рамок [55].

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1} * (k - 1), k \in [1, m], \quad (3.2)$$

де s_k – розмір k -ї рамки;

s_{min} – мінімальний масштаб рамок для виявлення малих об'єктів;

s_{max} – максимальний масштаб рамок для виявлення великих об'єктів;

m – кількість карт ознак (карта ознак – оброблена версія вхідного зображення);

k – індекс поточної карти ознак.

Кarti ознак з більшими значеннями k відповідають за об'єкти великого розміру, у той час як k з меншими значеннями – виявляють об'єкти малого розміру.

Такий підхід дозволяє SSD виявляти об'єкти різних розмірів у межах одного зображення [55].

Для порівняння розглянемо наступну (третю) технологію, а саме: MobileNet.

MobileNet також можна використовувати для виявлення об'єктів [37, 54].

Ця технологія відрізняється високою швидкістю операцій і може бути застосована без наявності графічного процесора.

Разом з тим, MobileNet має меншу точність ніж YOLO або SSD. Саме тому застосування MobileNet передбачає використання іншої архітектури в якості класифікатора.

MobileNet, як і попередні розглянуті технології, побудована на згортковій нейронній мережі, але розпізнання об'єктів виконується з більшою швидкістю завдяки глибинним роздільним згорткам (Depthwise Separable Convolutions) [56].

MobileNet застосовує глибинні згортки до кожного каналу зображення (червоний, зелений, синій). Потім точкові згортки об'єднують ці канали. Варто уточнити, що глибинні та точкові згортки використовуються замість стандартних згорток [37, 54].

Комбінація глибинних та точкових згорток дозволяє MobileNet знизити складність розрахунків приблизно у 8-9 разів у порівнянні зі стандартною згорткою [56].

Трансферне навчання та його переваги

При проведенні наукового дослідження обрано комплексне використання обох технологій: MobileNet та SSD, оскільки такий комбінований підхід дозволив поєднати швидкість MobileNet та точність розпізнавання об'єктів SSD, а також трансферне навчання.

Трансферне навчання – це дослідницька задача у машинному навчанні, зосереджена на збереженні знань, отриманих під час вирішення однієї задачі, та подальшого їх застосування для вирішення інших, але пов'язаних між собою задачами [57].

Трансферне навчання застосовано в комбінації з MobileNet та SSD для покращення ефективності розпізнавання об'єктів, зокрема, при обмеженому обсязі навчальних даних.

3.5 Комплексне застосування трансферного навчання для вирішення проблеми розпізнавання великих статичних та раптових перешкод

Застосування процесу трансферного навчання у проведеному науковому дослідженні

Безпілотні апарати (БпА) ефективно використовуються для збору інформації про об'єкти на місцевості та їх фізичні характеристики. У роботі [37] розглянуто та досліджено ефективність розпізнавання вибоїн за допомогою згорткової нейронної мережі з використанням безпілотних наземних транспортних засобів. Цей спосіб дозволяє уникнути аварійних ситуацій на дорогах.

З використанням безпілотних наземних апаратів проводять дослідження в несприятливих умовах і важкодоступних місцях, а нейронна мережа з розпізнаванням об'єктів надає можливість швидко виявити перешкоди.

Процес трансферного навчання почався з навчання базової моделі, тобто з попереднього тренування. В дисертаційному дослідженні використано конфігурацію `ssd_mobilenet_v1_pets.config` [37] для класифікації зображень.

Завдяки цій конфігурації проходить вивчення загальних ознак об'єктів, а саме: базові форми та текстури.

Отже, для проведення наукового дослідження обрано трансферне навчання, оскільки цей метод дозволив: зекономити час навчання нейронної мережі, швидко виявити великі статичні перешкоди та невизначеності (раптові перешкоди на шляху БпА).

Таким чином, перевагою трансферного навчання є те, що воно проходить швидко і потребує менше даних. Саме з цієї причини обрано трансферне навчання [37] у проведенні наукового дисертаційного дослідження. У додатку Г наведено лістинг коду з трансферним навчанням.

В ході проведення наукового дослідження експериментально підтверджено, що точність розпізнавання раптових перешкод (наприклад, вибоїни, ями) на шляху БпНА залежить від якості зображень.

Результати розпізнавання раптових перешкод за допомогою трансферного навчання

Результати тестування розпізнавання раптових перешкод, які демонструють ефективність використання трансферного навчання, наведені у вигляді таблиці [37].

У рамках експерименту [37] зібрано близько 200 зображень раптових перешкод (вибоїн та ям). Окремо відібрано 50 зображень для тестування і 150 зображень для тренування.

Всі зображення взяті у денний час, відстань до поверхні землі в межах 1 метра [37].

За допомогою програми Labellmg анотовано (оброблено з метою розпізнавання) всі 200 зображень [37].

Для створення технології розпізнавання раптових перешкод (вибоїн та ям), використано попередньо натреновану модель та застосовано трансферне навчання для вивчення нового об'єкта (раптової перешкоди).

Результати експериментів наведено на рисунку 3.14.

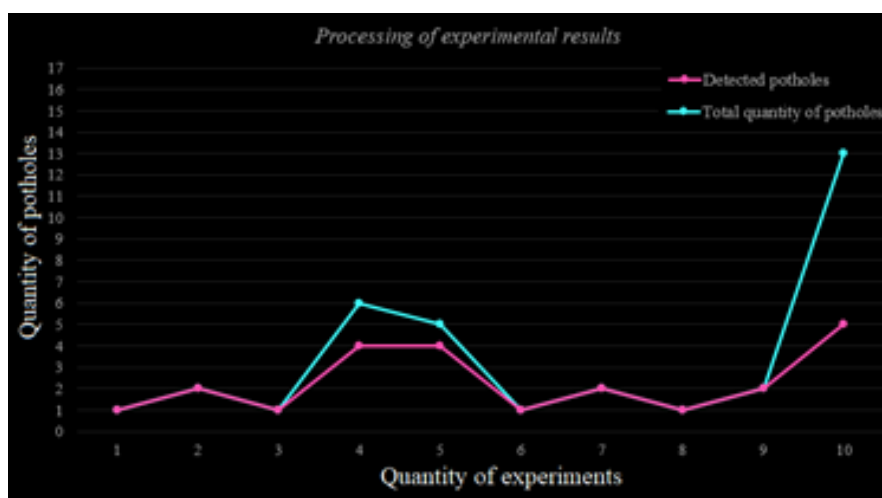


Рисунок 3.14 – Залежність точності розпізнавання від кількості ям [37]

Всі дисертаційні дослідження, які пов'язані з навчанням нейронної мережі, виконано із застосуванням симулятора Microsoft AirSim. Згорткова нейронна мережа ефективно розпізнала нові об'єкти, а саме – великі статичні блоки. Для проведення експериментів зібрані та опрацьовані такі зображення:

- близько 100 великих статичних блоків (перешкод) з камери наземного БпА;
- близько 100 великих статичних блоків (перешкод) з камери літального БпА;
- перешкод, зібраних у денний час;
- перешкод, зібраних у нічний час за допомогою інфрачервоного бачення;
- перешкод, зібраних в умовах туману;
- перешкод, зібраних з різних ракурсів.

Після проведеного трансферного навчання нейронної мережі [37], отримано наступні результати виявлення великих статичних перешкод.

Далі наведено приклад розпізнаних великих статичних блоків у денний час з камери наземного БпА (рисунок 3.15).



Рисунок 3.15 – Виявлення великих статичних блоків за допомогою камери БпНА у денний час

Далі наведено приклад розпізнавання великих статичних перешкод з використанням камери літального БПА у денний час (рисунок 3.16).

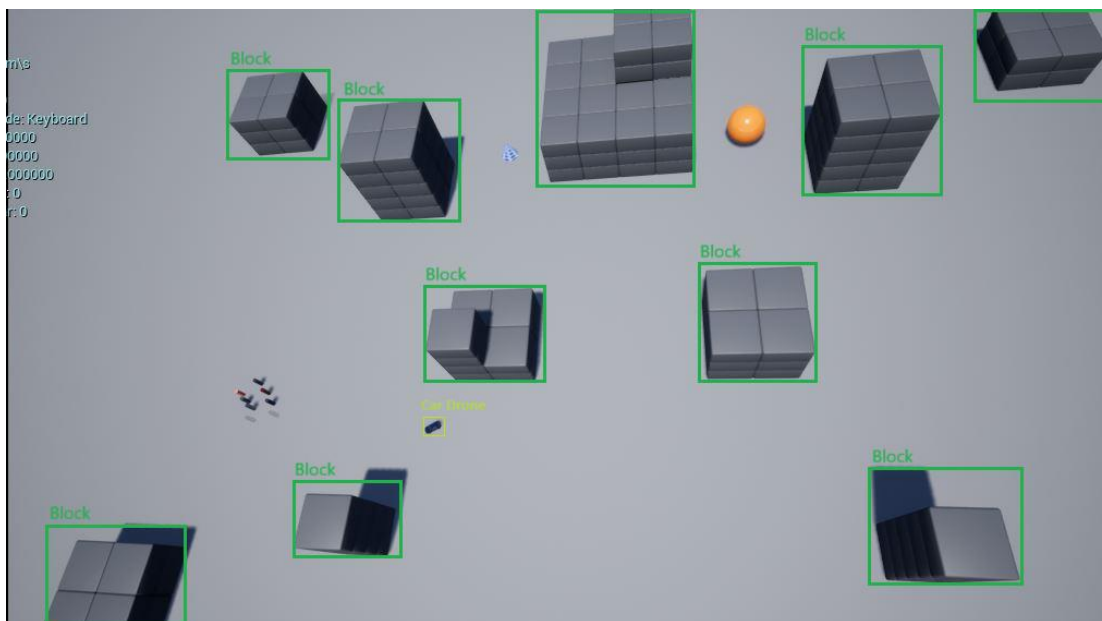


Рисунок 3.16 – Виявлення великих статичних блоків за допомогою камери БПЛА у денний час

В процесі дисертаційного дослідження проведено 7 експериментів з виявлення статичних перешкод у денний час. Кожен експеримент виявляв перешкоди з різних ракурсів камер, як наземного так і літального БПА. Також розраховано ефективність виявлення перешкод у денний час.

Далі наведено формулу для розрахунку ефективності виявлення перешкод в межах одного експерименту:

$$\text{Ефективність}(\%) = \frac{\text{Кількість виявлених перешкод}}{\text{Загальна кількість перешкод}} * 100\%$$

Загальна ефективність виявлення перешкод для всіх експериментів має наступний вигляд:

$$\text{Загальна ефективність}(\%) = \frac{\text{Сума ефективності експериментів}}{\text{Кількість експериментів}} * 100\%$$

Записи результатів проведених експериментів наукового дослідження щодо виявлення статичних перешкод у денний час наведено у таблиці 3.1.

Таблиця 3.1 – Виявлення статичних перешкод у денний час

Номер експерименту	Загальна кількість статичних перешкод	Кількість виявлених статичних перешкод
1	15	14
2	21	18
3	8	8
4	4	4
5	5	5
6	4	4
7	7	7

За проведеними розрахунками загальна ефективність виявлення перешкод у денний час становить 93,75%.

Аналогічним чином проведено експерименти з виявлення великих статичних блоків у нічний час (рисунок 3.17). Кількість проведених експериментів становить 7.

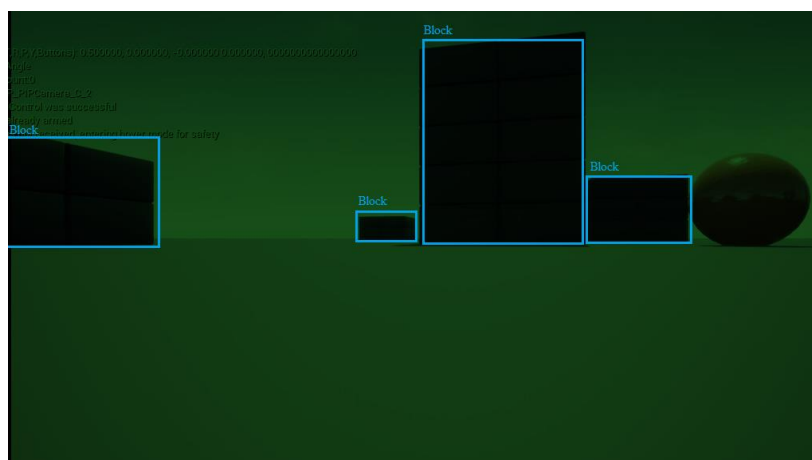


Рисунок 3.17 – Виявлення великих статичних блоків за допомогою камери БпНА у нічний час

Результати розрахунків загальної ефективності виявлення перешкод у нічний час наведено у таблиці 3.2.

Таблиця 3.2 – Виявлення статичних перешкод у нічний час

Номер експерименту	Загальна кількість статичних перешкод	Кількість виявлених статичних перешкод
1	15	9
2	21	14
3	8	5
4	4	3
5	5	3
6	4	3
7	7	4

За результатами проведених експериментів встановлено, що загальна ефективність виявлення перешкод у нічний час становить 65,07%.

Додатково проведено експерименти з виявлення великих статичних блоків в умовах туману (рисунок 3.18).

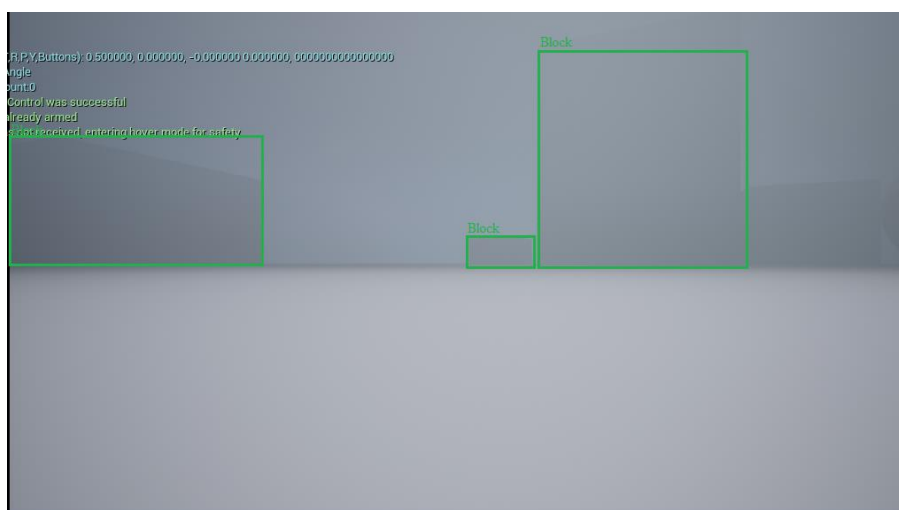


Рисунок 3.18 – Виявлення великих статичних блоків за допомогою камери БпНА в умовах туману

Результати розрахунків загальної ефективності виявлення перешкод в умовах туману за наведеною таблицею 3.3.

Таблиця 3.3 – Виявлення статичних перешкод в умовах туману

Номер експерименту	Загальна кількість статичних перешкод	Кількість виявлених статичних перешкод
1	15	8
2	21	12
3	8	4
4	4	2
5	5	3
6	4	2
7	7	4

За результатами проведених експериментів встановлено, що загальна ефективність виявлення перешкод в умовах туману становить 53,85%.

Загальна ефективність виявлення перешкод у нічний час становить 65,07%, а в умовах туману – 53,85%. Такі показники є незадовільними, тому є необхідність навчити нейронну мережу розпізнавати перешкоди у нових умовах з метою підвищення ефективності виявлення.

Таблиця 3.4 – Порівняння загальної ефективності виявлення перешкод до та після навчання нейронної мережі в різних умовах

	Денний час	Нічний час	Туман
Ефективність до навчання (%)	93,75	65,07	53,85
Ефективність після навчання (%)	93,75	85,94	71,8

Аналізуючи отримані результати, можна зробити висновок, що після навчання нейронної мережі загальна ефективність виявлення перешкод значно підвищилася, а саме: у нічний час на 20,87%, в умовах туману на 17,95%.

3.6 Навчання з підкріпленням

Розпізнавання перешкод для налаштування автономної роботи обох БПА є недостатнім, оскільки потрібен додатковий інструмент із застосуванням якого перешкоди будуть подолані. Саме таким інструментом і є навчання з підкріпленням.

Навчання з підкріпленням – це галузь машинного навчання, в якій агенти взаємодіють з навколишнім середовищем для досягнення цілей [58].

Взаємодія між агентом та середовищем – це дії, які виконуються агентом у відповідь на стан середовища з метою максимізації загальної винагороди [58].

Будемо притримуватися наступних визначень.

Дія – механізм, завдяки якому агент виконує перехід між дозволеними середовищем станами. Агент обирає дію, застосовуючи стратегію [59].

Стратегія – це план переходу між станом агента до дії [59].

Агент – це об'єкт, який спостерігає та використовує певну стратегію в даному середовищі, щоб максимізувати очікувану винагороду [59].

Винагорода – числовий результат, який отримує агент внаслідок переходу між станами, які визначаються середовищем (дією) [59].

Стан – значення параметрів, що описують поточне налаштування середовища. Агент застосовує ці параметри для вибору дії [59].

Розглянемо застосування навчання з підкріпленням, яке використано у дисертаційному дослідженні.

В подальшому викладенні будемо розглядати БпНА в якості агента.

Застосування згорткових нейронних мереж (ЗНМ) надає можливість агенту (БпНА) обробляти 2D-зображення при глибинному навчанні з підкріпленням.

Агент навчається тому, що «побачив» завдяки ЗНМ. ЗНМ перетворює набір пікселів у дії [56].

Навчання з підкріпленням застосовано в науковому дослідженні для успішного подолання великих статичних та раптових перешкод наземним БПА.

Розглянемо інструмент навчання з підкріпленням, який дозволяє агенту (БпНА):

- самостійно тренуватися для виконання бажаної поведінки;
- забезпечити автономне прийняття рішень в різних ситуаціях;
- проводити велику кількість навчальних експериментів.

Загальна функція винагороди визначена як:

$$R(s, a) = \begin{cases} +1, & \text{якщо } d > 2 \text{ м (цільовий стан);} \\ 0, & \text{якщо } 1 \leq d < 2 \text{ м (проміжний стан);} \\ -1, & \text{якщо } d \leq 1 \text{ м (критичний стан).} \end{cases}$$

де $R(s, a)$ – винагорода, яку отримує БпНА в стані s після виконання дії a ,

d – відстань між БпНА і перешкодою;

+1: БпНА отримує позитивну винагороду, якщо він виконав необхідні дії для об'їзду перешкоди;

0: нейтральна винагорода, коли БпНА знаходиться на прийнятній, але не ідеальній відстані (1-2 метри) до перешкоди;

-1: негативна винагорода за недотримання оптимальної відстані (менше 1 метра) або зіткнення.

При реалізації навчання з підкріпленням використано симулятор Microsoft AirSim, оскільки тестування алгоритмів на реальному БПА недоцільне.

Тренування поведінки наземного БПА проведено у середовищі з великими статичними блоками (рисунок 3.15).

Для налаштування агента (БпНА) використано бібліотеку «TensorFlow Agents» з метою реалізації базового алгоритму навчання з підкріпленням – алгоритму Q-навчання (зокрема, Deep Q-Network або DQN) [60–62].

Розглянемо взаємодію навчання з підкріпленням з середовищем Microsoft AirSim.

Для роботи з агентом (БпНА) використано Microsoft AirSim API на Python.

Агенти мають можливість обробляти кожен дію окремо.

Враховано зміну поточного стану БпНА та виконання переміщень у віртуальному середовищі.

Епізодом вважаємо одну завершену послідовність дій агента в середовищі, що починається зі стартового стану і закінчується, коли досягається певна умова завершення.

Взаємодія агента «TensorFlow Agents» з середовищем передбачає використання методів `reset` та `step` [61].

Метод `reset` встановлює початковий стан епізоду, розміщуючи БпНА у стартовій точці. Таке налаштування дозволяє запускати новий навчальний епізод.

Метод `step` виконує певну дію, що дозволяє отримати значення нового стану і визначити величину винагороди.

Отже, відбувається створення досвіду для запам'ятовування та подальшого навчання агента (БпНА). Наприклад, коли БпНА відійшов від маршруту, стикнувся з перешкодою або завершив маршрут, відбувається закінчення епізоду.

Кожним епізодом вважається і повне виконання місії (місія успішна), і неповне виконання місії (місія почалася, але не закінчилась через неподолання перешкоди).

Інформація про поточне місцезнаходження перешкоди у полі зору агента надзвичайно важлива. Для формування такої інформації використано штучний обмежувальний кадр. Крім того, агент (БпНА) отримує набір з трьох кадрів та карту глибини для врахування перешкод [60].

Навчання агента (БпНА) здійснюється через `Replay Buffer`, де зберігаються історія дій агента. Заповнення `Replay Buffer` починається з випадкових рухів агента, які моделюють різні ситуації. Агент (БпНА) виконує дії в середовищі та навчається на зразках з `Replay Buffer` і повторює ці дії циклічно [60-62].

Крок – окремі дії, які агент виконує в середовищі протягом одного епізоду.

В експерименті вважаємо, що один успішний крок відповідає одній одиниці винагороди.

Тоді 45 крокам ідеального епізоду відповідає 45 одиниць винагород.

В результаті проведеного наукового дослідження отримано:

- середню тривалість епізоду, яка становить близько 45 кроків під час навчання за 2000 епізодів;
- середню винагороду розміром 32 одиниці;
- успішне виконання місії агентом (БпНА), оскільки агент подолав великі статичні та раптові перешкоди.

Винагорода середнього реального епізоду у порівнянні з ідеальним епізодом становить: $\frac{32}{45} * 100\% = 71,1\%$.

Тобто винагорода середнього реального епізоду становить 71,1% від винагороди ідеального епізоду.

Результати проведеного наукового дослідження наведено на рисунку 3.19.

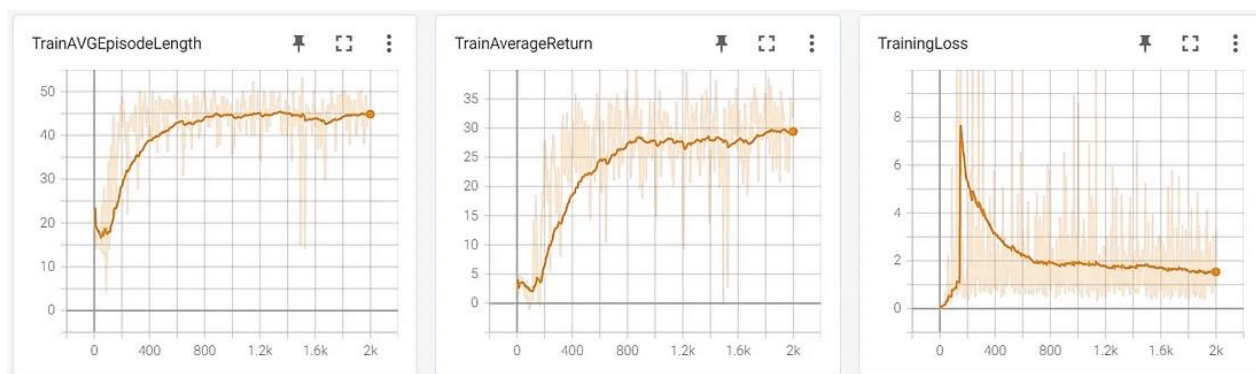


Рисунок 3.19 – Процес навчання з відстеженням середньої тривалості епізоду, середньої винагороди та втрати під час навчання

3.7 Висновки до розділу 3

У третьому розділі проведено порівняльний аналіз відомих симуляторів, які необхідні для проведення практичних експериментів. Розглянуто найпоширеніші симулятори (ArduPilot, PX4, ROS, Gazebo, Microsoft AirSim), а також проаналізовані їхні переваги та недоліки.

Для проведення досліджень та експериментів обрано симулятор Microsoft AirSim, оскільки:

- містить програмні модулі та дозволяє вибрати різноманітні віртуальні середовища, які якнайкраще імітують реальність;
- передбачена підтримка навчання нейронних мереж за допомогою внутрішніх підпрограм для збору даних.

В процесі наукового дослідження спочатку розглянуто базову архітектуру роботи симулятора Microsoft AirSim. Наукове дослідження вимагало внести зміни до базової архітектури та створити нові модифікації симулятора Microsoft AirSim, оскільки передбачало розробити інформаційну технологію керування безпілотними апаратами на базі ігрового підходу та нейронних мереж, що включало необхідність, зокрема, подолання великих статичних та раптових перешкод.

Запропоновані в науковому дослідженні модифікації симулятора Microsoft AirSim наступні:

- прибрано центральний контролер (Flight Controller Firmware) та окремий обчислювальний мікрокомп'ютер (Companion Computer);
- додано нові фізичні контролери (Physical Controllers);
- додано блок вбудованих алгоритмів (Built-in algorithms);
- налагоджено взаємодію вбудованими алгоритмами з API Layer;
- збережено симуляційний модуль, який залишився без змін, у тому числі, блоки Vehicle Model, Environment Model, Sensor Models, Physics Engine та Rendering Engine.

Для проведення експериментів наукового дослідження запропоновано використання двох типів БПА (наземного та літального), які вбудовані у Microsoft AirSim.

В дисертаційному дослідженні наведено огляд різноманітних середовищ, які дозволяють використовувати обраний симулятор, та які потенційно можна використовувати для перевірки ігрового підходу і нейронних мереж у керуванні безпілотними апаратами.

Розглянуто технології для розпізнавання об'єктів на основі згорткової нейронної мережі, які вважаються найактуальнішими, а саме: YOLO, Single Shot MultiBox Detector (SSD), MobileNet.

Обрано комбінований підхід, який дозволяє поєднати швидкість розпізнавання MobileNet з точністю виявлення об'єктів за допомогою SSD та перевагами трансферного навчання.

Розглянуто та досліджено ефективність розпізнавання раптових перешкод (вибоїн та ям) за допомогою згорткової нейронної мережі.

Проведені експерименти склалися з навчання згорткової нейронної мережі розпізнавати великі статичні перешкоди за допомогою трансферного навчання. Зображення великих статичних перешкод бралися за різних умов, у тому числі несприятливих (ніч, туман).

Відповідно до отриманих наукових результатів запропонована у дослідженні комбінована згорткова нейронна мережа ефективно розпізнала нові об'єкти, а саме – великі статичні блоки та раптові перешкоди (вибоїни, ями).

Підвищено загальну ефективність виявлення перешкод у нічний час на 20,87%, в умовах туману на 17,95%.

Розраховано загальну ефективність виявлення перешкод в різних умовах:

- денний час – 93,75%;
- нічний час – 85,94%;
- в умовах туману – 71,8%.

Навчання з підкріпленням застосовано в науковому дослідженні для успішного подолання великих статичних та раптових перешкод наземним БпА.

Використано основні поняття навчання з підкріпленням, а саме: агент, дія, стратегія, винагорода та стан.

В результаті проведеного наукового дослідження отримано:

- середню тривалість епізоду, яка становить близько 45 кроків під час навчання за 2000 епізодів;
- середню винагороду розміром 32 одиниці;

– успішне виконання місії агентом (БпНА), оскільки агент подолав великі статичні та раптові перешкоди.

Винагорода середнього реального епізоду у порівнянні з ідеальним епізодом становить: $\frac{32}{45} * 100\% = 71,1\%$.

Тобто винагорода середнього реального епізоду становить 71,1% від винагороди ідеального епізоду.

4 СТРУКТУРА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПІЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ

4.1 Обробка і збереження даних в процесі керування безпілотними апаратами

При проведенні дисертаційного дослідження дані з датчиків та інших пристроїв віртуальних БПЛА та БпНА зчитувалися і передавалися за допомогою API симулятора Microsoft AirSim до бази даних центра керування (оператора).

В процесі проектування інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж використовувалися можливості програмного забезпечення Apache Kafka.

Apache Kafka – це розподілена система обміну повідомленнями з високими пропускними можливостями між компонентами системи, діє по принципу «Публікація - Підписка» [63].

Apache Kafka дозволяє обробляти велику кількість повідомлень і зберігати їх, не турбуючись про продуктивність і втрату даних [63].

Перед інтеграцією Apache Kafka у систему віртуальних безпілотних апаратів необхідно визначити, які дані будемо збирати з зовнішніх віртуальних сенсорів збору та модулів обробки даних.

1. Дані про навігацію та політ:
 - а) GPS координати – широта, довгота, висота;
 - б) з акселерометрів та гіроскопів: прискорення, кутові швидкості.
2. Дані з камер та оптичних сенсорів:
 - а) фотографії та короткі відео.
3. Дані з погодних датчиків:
 - а) температура – поточна температура навколишнього середовища;
 - б) рівень освітлення – інтенсивність світла.
4. Технічні дані:
 - а) ідентифікатор БпА;

б) рівень заряду батареї;

в) якість передачі даних – швидкість передачі, затримка, втрати пакетів.

5. Дані про виконання місії:

а) плани польотів – маршрути, завдання;

б) логування виконаних завдань – історія виконаних дій, результати завдань;

в) аномалії та події – випадки відхилень від плану, аварійні ситуації;

г) перенаправлення завдання на інший безпілотний апарат.

В процесі виконання наукового дослідження для збереження даних використано базу даних PostgreSQL, яка встановлена на кожному БПА, а також у центрі керування. Для запуску бази даних на кожному БПА та центрі керування використано Docker [64]. На рисунку 4.1 наведено скрипт створення трьох баз даних (БД). Дві БД знаходяться на кожному з БПА, а третя у центрі керування.

Рисунки 4.1 – 4.11 наведені з відповідними номерами Д.1 – Д.11 у додатку Д.

```

1  ---
2  version: '2'
3  services:
4    postgres1:
5      image: postgres:11
6      container_name: postgresDrone1
7
8      ports:
9        - 5435:5432
10     environment:
11       - POSTGRES_USER=postgres1
12       - POSTGRES_PASSWORD=postgres1
13       - POSTGRES_DB=postgresDrone1
14     command: ['postgres', '-c', 'wal_level=logical']
15
16
17    postgres2:
18      image: postgres:11
19      container_name: postgresDrone2
20
21      ports:
22        - 5436:5432
23     environment:
24       - POSTGRES_USER=postgres2
25       - POSTGRES_PASSWORD=postgres2
26       - POSTGRES_DB=postgresDrone2
27     command: ['postgres', '-c', 'wal_level=logical']
28
29    postgresMain:
30      image: postgres:11
31      container_name: postgresMain
32
33      ports:
34        - 5438:5432
35     environment:
36       - POSTGRES_USER=postgresMain
37       - POSTGRES_PASSWORD=postgresMain
38       - POSTGRES_DB=postgresMain
39     command: ['postgres', '-c', 'wal_level=logical']

```

Рисунок 4.1 – Створення баз даних PostgreSQL через Docker

Далі наведено скрипт створення таблиці для збереження даних з зовнішніх віртуальних сенсорів збору та модулів обробки (рисунк 4.2).

```

1 create table "DroneInfo"
2 (
3     id TEXT NOT NULL
4     constraint "DroneInfo_pk"
5         primary key,
6     "droneId" TEXT,
7     timestamp TIMESTAMP DEFAULT now() NOT NULL,
8
9     -- Navigation and flight data
10    coordinates TEXT,
11    acceleration DOUBLE PRECISION,
12    "angularVelocity" DOUBLE PRECISION,
13
14    -- Camera and optical sensor data
15    photo TEXT,
16    video TEXT,
17
18    -- Weather sensor data
19    temperature DOUBLE PRECISION,
20    "lightIntensity" DOUBLE PRECISION,
21
22    -- Technical data
23    "batteryLevel" INTEGER,
24    "transmissionSpeed" DOUBLE PRECISION,
25    latency DOUBLE PRECISION,
26    "packetLoss" DOUBLE PRECISION,
27
28    -- Mission execution data
29    route TEXT,
30    tasks TEXT,
31    "actionLog" TEXT,
32    result TEXT,
33    "deviationCases" TEXT,
34    "emergencySituations" TEXT,
35    "redirectedDroneId" TEXT
36 )

```

Рисунок 4.2 – Створення таблиці «DroneInfo»

Розглянемо отриману в процесі наукового дослідження структуру програмного рішення на базі FastAPI на рисунку 4.3 [65].

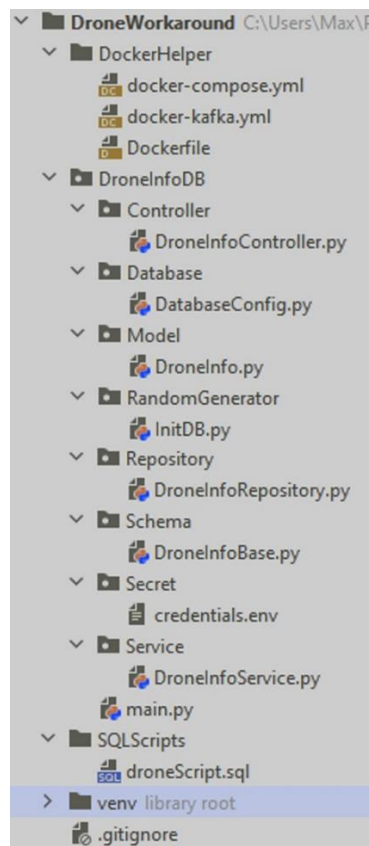


Рисунок 4.3 – Структура програмного рішення

Далі наведено опис структури запропонованого в процесі наукового дослідження програмного забезпечення для роботи з БД центра керування і БД БпА.

Директорія Controller містить файли, що відповідають за маршрутизацію та обробку HTTP-запитів. Контролери приймають запити, передають їх відповідним сервісам для обробки та повертають результати центру керування (оператору) та БпА.

Директорія Database містить конфігураційні файли підключення до бази даних.

Директорія Model містить класи, які відповідають доменним сутностям програми. Ці класи використовуються для створення таблиць в базі даних і є основою для формування і зберігання даних. Доменні сутності є центральною частиною багатоварової архітектури і відповідають за структуру даних, з якими працює програма [66].

Директорія RandomGenerator містить скрипти для генерації випадкових даних, які використовуються для наповнення бази даних тестовими записами.

Директорія Repository містить класи репозиторіїв, які реалізують доступ до бази даних. Репозиторії абстрагують операції з базою даних, надаючи універсальні методи для збереження, отримання, оновлення та видалення даних. Ця директорія представляє рівень доступу до даних в багатоваровій архітектурі [66].

Директорія Schema використовується для зміни формату (серіалізації/десеріалізації) даних, які передаються через API. Ці схеми визначають структуру даних, що приймаються і повертаються контролерами, забезпечуючи їхню коректність [38].

Директорія Secret містить конфіденційні дані, такі як параметри підключення до БД. Наприклад, файл `credentials.env` містить облікові дані для доступу до БД.

Файл `main.py` є точкою входу в додаток. Він відповідає за налаштування та запуск FastAPI додатків, підключення до баз даних на різних портах.

Директорія SQLScripts містить SQL-скрипти, необхідні для роботи з БД, такі як створення таблиць та початкове наповнення даними.

Директорія DockerHelper містить файли, необхідні для налаштування та запуску Docker-контейнерів, такі як `docker-compose.yml`, `docker-kafka.yml` та `Dockerfile` [64]. Ці файли використовуються для налаштування середовища розробки та тестування за допомогою контейнеризації.

В процесі проєктування багаторівневої архітектури взаємодії з БД використана мова програмування Python та бібліотека FastAPI, що надало такі переваги:

- можливість працювати з декількома екземплярами баз даних;
- спрощене налагодження програми.

4.2 Розробка діаграми класів для інформаційної технології керування безпілотними апаратами

В процесі наукового дослідження створені такі файли: `DroneInfoController.py`, `DatabaseConfig.py`, `DroneInfo.py`, `InitDB.py`, `DroneInfoRepository.py`, `DroneInfoService.py`, `main.py`.

Надамо опис цих файлів.

Файл `DroneInfoController.py` забезпечує маршрутизацію та обробку HTTP-запитів (таблиця 4.1).

Таблиця 4.1 – Клас `DroneInfoController`

Назва	Тип даних	Призначення
router	APIRouter	Маршрутизатор для обробки HTTP-запитів

Файл `DatabaseConfig.py` містить конфігураційні дані для підключення до бази даних (таблиця 4.2).

Таблиця 4.2 – Клас `DatabaseConfig`

Назва	Тип даних	Призначення
<code>SQLALCHEMY_DATABASE_URL</code>	str	URL підключення до першої бази даних

Продовження таблиці 4.2

Назва	Тип даних	Призначення
SQLALCHEMY_DATABASE_URL2	str	URL підключення до другої бази даних
engine1	Engine	Підключення до першої бази даних
SessionLocal1	None	Сесія для роботи з першою базою даних
engine2	Engine	Підключення до другої бази даних
SessionLocal2	None	Сесія для роботи з другою базою даних

Файл DroneInfo.py містить клас, що представляє сутність DroneInfo (таблиця 4.3).

Таблиця 4.3 – Клас DroneInfo

Назва	Тип даних	Призначення
id	String	Унікальний ідентифікатор БпА
droneId	String	Ідентифікатор БпА
timestamp	DateTime	Часова мітка
coordinates	String	Координати БпА
acceleration	Float	Прискорення
angularVelocity	Float	Кутова швидкість
photo	String	Фото
video	String	Відео
temperature	Float	Температура
lightIntensity	Float	Інтенсивність світла
batteryLevel	Integer	Рівень заряду батареї
transmissionSpeed	Float	Швидкість передачі
latency	Float	Затримка
packetLoss	Float	Втрата пакетів
route	String	Маршрут
tasks	String	Завдання
actionLog	String	Журнал дій
result	String	Результат
deviationCases	String	Випадки відхилення
emergencySituations	String	Надзвичайні ситуації
redirectedDroneId	String	Ідентифікатор перенаправленого БпА

Файл InitDB.py містить функції для генерації та додавання тестових даних до бази даних (таблиця 4.4).

Таблиця 4.4 – Клас InitDB

Назва	Тип даних	Призначення
addTestData	None	Функція додавання тестових даних до бази даних
testData1	list[DroneInfo]	Список тестових даних для першої бази даних
testData2	list[DroneInfo]	Список тестових даних для другої бази даних

Файл DroneInfoRepository.py містить репозиторії для роботи з базою даних (таблиця 4.5).

Таблиця 4.5 – Клас DroneInfoRepository

Назва	Тип даних	Призначення
getAllDroneInfo	list[Type[DroneInfo]]	Отримання списку всіх записів з БД БпА
getDroneInfoById	Any	Пошук запису у БД БпА за ідентифікатором
insertDroneInfo	DroneInfo	Створення нового запису у БД БпА
updateDroneInfo	None	Оновлення запису у БД БпА
deleteDroneInfo	None	Видалення запису у БД БпА

Файл DroneInfoService.py містить бізнес-логіку програми (таблиця 4.6).

Таблиця 4.6 – Клас DroneInfoService

Назва	Тип даних	Призначення
getAllDroneInfo	list[Type[DroneInfo]]	Отримання списку всіх записів з БД БпА
getDroneInfoById	Any	Пошук запису у БД БпА за ідентифікатором
insertDroneInfo	DroneInfo	Створення нового запису у БД БпА
updateDroneInfo	None	Оновлення запису у БД БпА
deleteDroneInfo	None	Видалення запису у БД БпА

Файл main.py є точкою входу в додаток (таблиця 4.7).

Таблиця 4.7 – Клас main

Назва	Тип даних	Призначення
createApp	FastAPI	Створення та налаштування додатків
runApp1	None	Запуск першого додатку
runApp2	None	Запуск другого додатку

Діаграму класів для інформаційної технології керування безпілотними апаратами з усіма класами надано у додатку Е.

4.3 Забезпечення системи обміну даними при керуванні безпілотними апаратами. Інтеграція Apache Kafka у систему інформаційної технології керування БпА

Одна з проблем, яку треба вирішити в процесі дисертаційного дослідження – забезпечити надійний та безпечний обмін даними в процесі керування коаліцією БпА.

В результаті проведеного аналізу актуальних інструментів, які забезпечують обмін даними та можуть бути застосовані для обміну даними в процесі керування БпА, обрано Apache Kafka.

Обґрунтування вибору розподіленої системи Apache Kafka з огляду на її можливості в якості інструменту збору та передачі інформації надано у підрозділі 4.1.

Отже, при виконанні дисертаційного дослідження обмін даними відбувався із застосуванням хмарного серверу, на якому встановлена Apache Kafka.

Перейдемо до опису процесу розгортання серверу Apache Kafka.

Розгортання серверу передбачало встановлення програмного забезпечення Kafka Connect у необхідній конфігурації. За замовчуванням розглядався: один екземпляр серверу або відмовостійкий кластер.

В процесі наукового дослідження для підготовки образу (image) і запуску одного екземпляру серверу створено файл образу з використанням Dockerfile.

Скрипт Dockerfile сформовано заздалегідь відповідно до умов оточення і середовища розгортання. Приклад побудови Dockerfile наведено на рисунку 4.4.

```

1 FROM confluentinc/cp-kafka-connect:7.6.1
2
3 ENV CONNECT_PLUGIN_PATH="/usr/share/java,/usr/share/confluent-hub-components,/usr/share/java/kafka-connect-jdbc,/usr/share/java/kafka"
4
5 RUN confluent-hub install --no-prompt confluentinc/kafka-connect-jdbc:latest
6 RUN confluent-hub install --no-prompt debezium/debezium-connector-postgresql:latest
7 RUN confluent-hub install --no-prompt jcustenborder/kafka-connect-transform-common:0.1.0.54
8

```

Рисунок 4.4 – Скрипт для створення модифікованого образу Kafka Connect

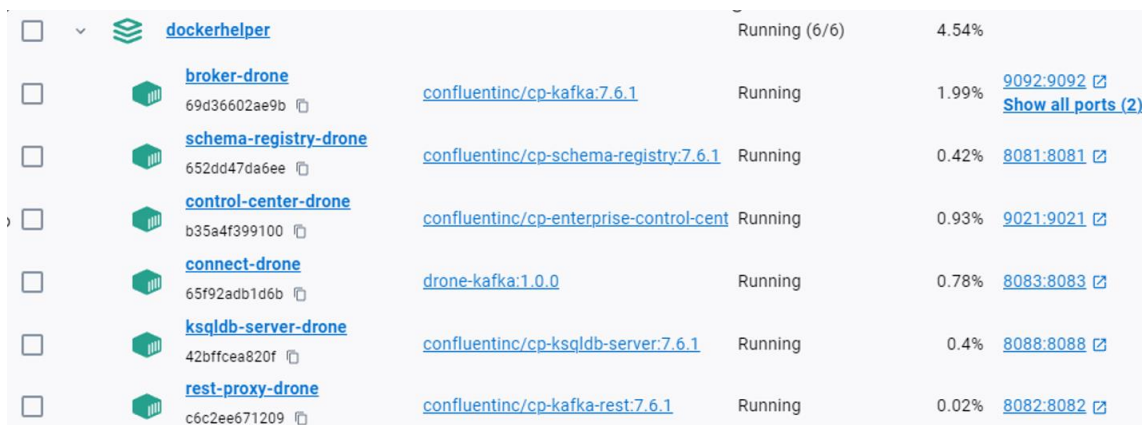
Проектування інформаційної технології керування БПА на базі ігрового підходу та нейронних мереж вимагало проведення модифікації образу Kafka Connect.

За допомогою команди «docker build . -t drone-kafka:1.0.0» створено модифікований образ Kafka Connect, який містить:

1. cp-server-connect – базовий образ для контейнера;
2. kafka-connect-jdbc – забезпечує підключення до реляційних баз даних;
3. CONNECT_PLUGIN_PATH – налаштовує місця пошуку плагінів;
4. debezium-connector-postgresql – відстежує зміни в базах даних PostgreSQL;
5. kafka-connect-transform-common – перетворює деякі типи даних (за запитом).

Всі образи, які необхідні для запуску екземпляру сервера, зібрані у файлі docker-kafka.yml. Цей файл створено на основі docker-compose.yml з офіційного сайту confluent [67].

Для запуску образів у контейнері виконано команду «docker-compose -f docker-kafka.yml up -d» (рисунок 4.5).



Container Name	Image	Status	Usage	Ports
broker-drone 69d36602ae9b	confluentinc/cp-kafka:7.6.1	Running	1.99%	9092:9092 Show all ports (2)
schema-registry-drone 652dd47da6ee	confluentinc/cp-schema-registry:7.6.1	Running	0.42%	8081:8081 Show all ports (2)
control-center-drone b35a4f399100	confluentinc/cp-enterprise-control-cent	Running	0.93%	9021:9021 Show all ports (2)
connect-drone 65f92adb1d6b	drone-kafka:1.0.0	Running	0.78%	8083:8083 Show all ports (2)
ksqldb-server-drone 42bfffca820f	confluentinc/cp-ksqldb-server:7.6.1	Running	0.4%	8088:8088 Show all ports (2)
rest-proxy-drone c6c2ee671209	confluentinc/cp-kafka-rest:7.6.1	Running	0.02%	8082:8082 Show all ports (2)

Рисунок 4.5 – Контейнери для роботи з Apache Kafka

В процесі наукового дослідження використані технічні терміни, а саме: «конектор» та «топик».

Конектор – невід’ємна частина платформи Apache Kafka. Конектор забезпечує обмін даними між Kafka та зовнішніми сховищами даних (базами даних, файловими системами і хмарними серверами). Конектор забезпечує підключення з метою отримання та передачі поточкових даних, використовуючи «Source-конектори» (для отримання даних) та «Sink-конектори» (для передачі даних у зовнішні сховища даних) [68–69].

Однією з основних переваг використання конектору – це економія часу проєктування інформаційних систем, зокрема, за рахунок відсутності потреби у написанні додаткового програмного продукту, яке забезпечує обмін даними.

Топик – журнал повідомлень, який використовується для групування повідомлень одного типу [68–69].

У дисертаційному дослідженні створено конектори і топіки для передачі інформації з БПА до загальної бази даних за допомогою Apache Kafka.

В процесі наукового дослідження сформовані Source-конектори, які забезпечують передачу даних з безпілотних апаратів (БПА) на сервер з Apache Kafka. Для створення Source-конекторів використано бібліотеку Debezium для PostgreSQL.

Для проєктування інформаційної технології особливе значення має використання спеціалізованої бібліотеки конекторів Debezium.

Debezium – спеціалізована бібліотека конекторів, яка фіксує зміни з різноманітних систем керування базами даних [70].

Конектор Debezium PostgreSQL фіксує зміни кожного рядка бази даних PostgreSQL. Розглянемо порядок створення Debezium конектора для отримання даних з бази даних БпНА, запропонований в процесі виконання наукової роботи.

Для підключення до баз даних, розгорнутих у Docker-контейнерах на WSL, достатньо виконати наступні кроки:

1. Відкрити cmd;
2. Ввести команду ipconfig;
3. Знайти локальну ір адресу (рисунок 4.6).

```
Ethernet adapter Ethernet 2:

Connection-specific DNS Suffix  . : 
Link-local IPv6 Address . . . . . : fe82::1cba:f23a:f659:8c15%22
IPv4 Address. . . . . : 192.168.0.131
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1
```

Рисунок 4.6 – Локальна ір адреса

Опис створеного Debezium Source-конектора наведено на рисунку 4.7.

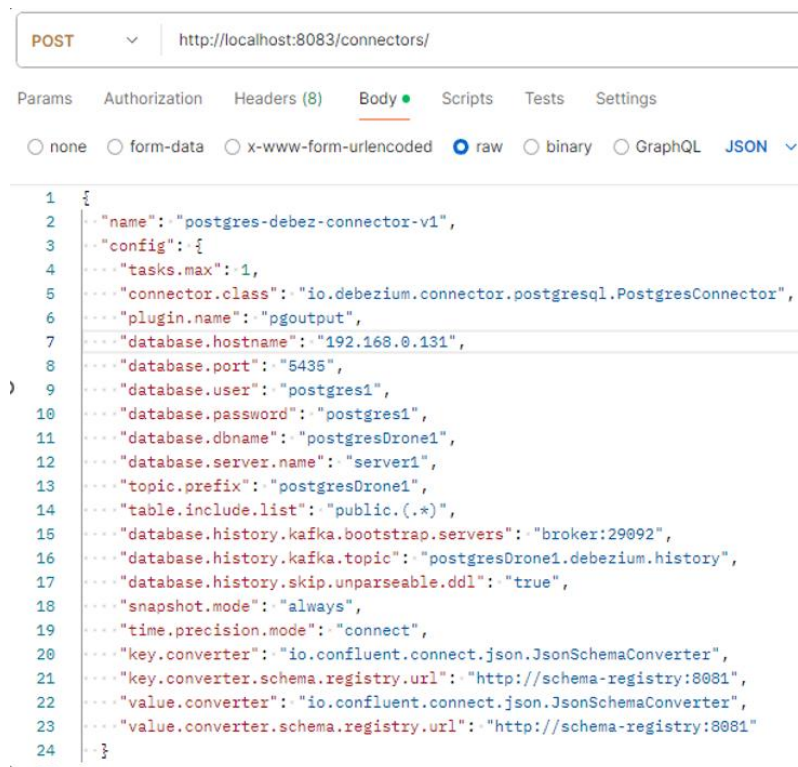


Рисунок 4.7 – Створення Debezium Source-конектора на прикладі БД БпНА

У таблиці 4.8 наведені налаштування та пояснення до кожного параметра Debezium Source-конектора.

Таблиця 4.8 – Налаштування та пояснення до кожного параметра Debezium Source-конектора

Назва параметра	Значення	Пояснення
name	postgres-debez-connector-v1	Назва конектора
tasks.max	1	Максимальна кількість задач, які потенційно може виконувати конектор одночасно
connector.class	io.debezium.connector.postgresql.PostgresConnector	Клас конектора для PostgreSQL, який буде використовуватись
plugin.name	pgoutput	Назва плагіна для захоплення змін з PostgreSQL
database.hostname	192.168.0.131	IP-адреса сервера бази даних
database.port	5435	Порт для підключення до бази даних
database.user	postgres1	Ім'я користувача
database.password	postgres1	Пароль користувача
database.dbname	postgresDrone1	Назва бази даних
database.server.name	server1	Унікальне ім'я сервера бази даних
topic.prefix	postgresDrone1	Ознака для всіх Kafka топіків, які будуть створені для цієї бази даних.
topic.creation.default.cleanup.policy	compact	Правило очищення топіку, яке використовується для зберігання лише останнього значення ключа в топіку.
topic.creation.default.replication.factor	1	Кількість копій для кожного топіка.
topic.creation.default.partitions	1	Кількість розбиття для кожного топіка.

Продовження таблиці 4.8

Назва параметра	Значення	Пояснення
table.include.list	public.(.*)	Перелік таблиць, які потрібно включити в процес «захоплення змін» (перелік таблиць до яких звертається конектор, щоб створити топіки).
database.history.kafka.bootstrap.servers	broker:29092	Список Kafka серверів, які використані для зберігання історії подій.
database.history.kafka.topic	postgresDrone1.debezium.history	Топік Kafka, в якому збережено події.
snapshot.mode	always	Режим створення знімків (snapshot). В даному випадку створювався виключно знімок всієї бази даних при запуску конектора.
time.precision.mode	connect	Режим точності часу, який використовувався для підключення.
key.converter	io.confluent.connect.json.JsonSchemaConverter	Конвертер ідентифікаторів, який використовувався для перетворення ключів повідомлень у формат даних JSON Schema.
key.converter.schema.registry.url	http://schema-registry:8081	URL реєстру схем для конвертера ключів.
value.converter	io.confluent.connect.json.JsonSchemaConverter	Конвертер значень, який використовувався для перетворення значень повідомлень у формат даних JSON Schema.
value.converter.schema.registry.url	http://schema-registry:8081	URL реєстру схем для конвертера значень.

Процедура створення Debezium Source-конекторів повторюється для кожної бази даних (база даних для БПЛА).

Перевірка працездатності Debezim Source-конектора продемонстрована на сторінці Control Center [67]. Control Center включено в docker-kafka.yml (рисунок 4.8).

Connectors			
2		2	0
Total		Running	Degraded

<input type="text" value="Search connectors"/>	Filter by category ▼
--	-----------------------------------



Connector name	Status	Category	Plugin name
 postgres-debez-connector-v2	● Running	Source	PostgresConnector
 postgres-debez-connector-v1	● Running	Source	PostgresConnector

Рисунок 4.8 – Створені Debezium Source-конектори

На рисунку 4.8 можна побачити всі два працюючі Debezium Source-конектори. Тепер переглянемо топіки, які створено після запуску Debezium Source-конекторів. Як результат Debezium Source-конекторів, створено два топіки, які містять по одній таблиці з кожної бази даних (рисунок 4.9).

Topics

<input type="text" value="Search topics"/>	<input checked="" type="checkbox"/> Hide internal topics
--	--

Topic name	Partitions
default_ksql_processing_log	1
docker-connect-configs	1
docker-connect-offsets	25
docker-connect-status	5
postgresDrone1.public.DroneInfo	1
postgresDrone2.public.DroneInfo	1

Рисунок 4.9 – Топіки, створені на основі Debezium Source-конекторів

Кожен топік містить повідомлення про операції вставки, оновлення та видалення даних з БД (рисунок 4.10).

postgresDrone1.public.DroneInfo

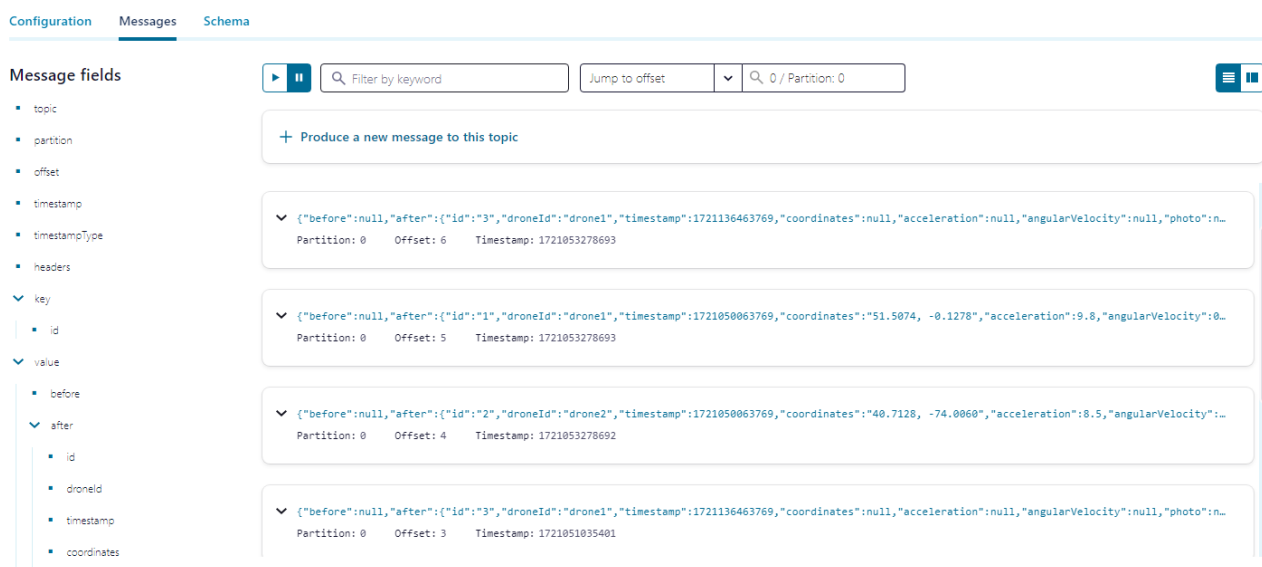


Рисунок 4.10 – Зразок повідомлень у топіку

Після завантаження даних у сервер з Kafka, починається етап передачі даних до загальної СУБД (основної системи управління базами даних).

Етапи передачі даних до загальної бази даних (бази даних центру керування)

Використання JDBC Sink-конектору.

JDBC Sink-конектор експортує дані з топіків Apache Kafka до бази даних за допомогою драйвера JDBC.

При виконанні наукової роботи використано команду «upsert» для гарантованого отримання запису у БД, а також підтримувалося автоматичне створення таблиць [71].

Створення JDBC Sink-конектора (рисунок 4.11).

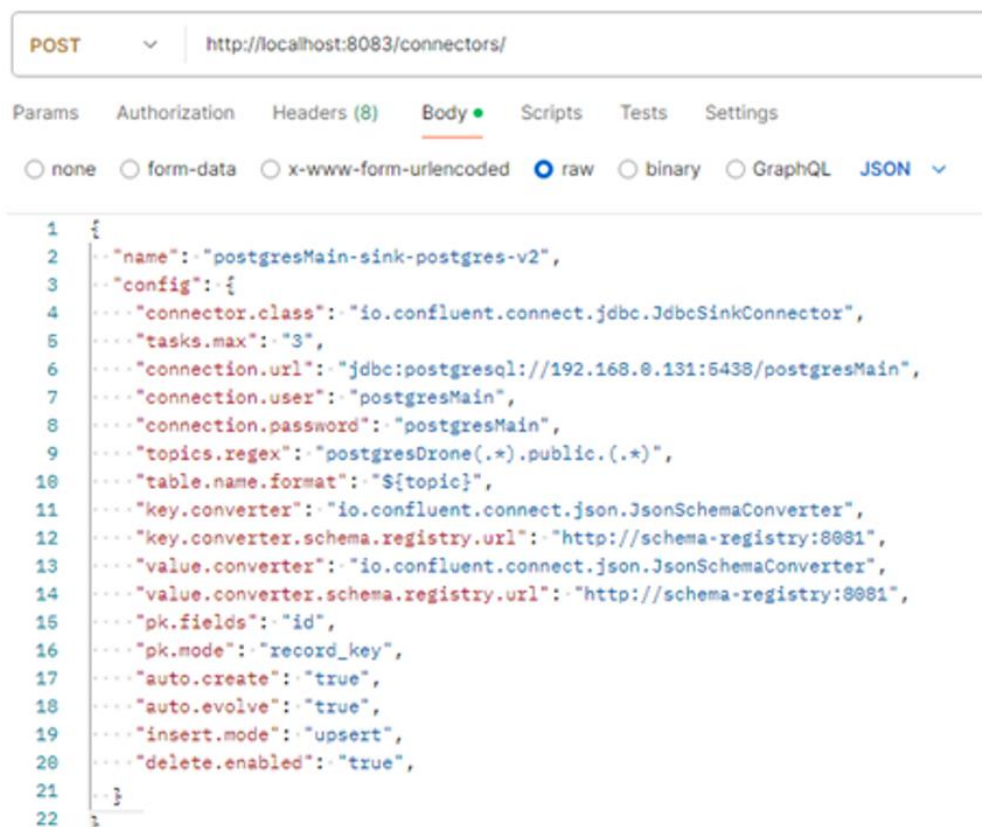


Рисунок 4.11 – Зразок створення JDBC Sink-конектора через Postman для загальної СУБД

У таблиці 4.9 наведені налаштування та пояснення до кожного параметра JDBC Sink-конектора.

Таблиця 4.9 – Налаштування та пояснення до кожного параметра JDBC Sink-конектора

Назва параметра	Значення	Пояснення
name	postgresMain-sink-postgres-v2	Назва конектора
connector.class	io.confluent.connect.jdbc.JdbcSinkConnector	Клас конектора для PostgreSQL, який використано для запису даних з Kafka в PostgreSQL
tasks.max	2	Максимальна кількість задач, які виконано конектором одночасно. Маємо дві задачі для отримання даних з двох БД.

Продовження таблиці 4.9

Назва параметра	Значення	Пояснення
connection.url	jdbc:postgresql://192.168.0.131:5438/postgresMain	URL підключення до бази даних PostgreSQL.
connection.user	postgresMain	Ім'я користувача
connection.password	postgresMain	Пароль користувача
topics.regex	postgresDrone(.*).public(.*)	Регулярний вираз для вибору топіків Kafka, які оброблено.
table.name.format	\${topic}	Формат назви таблиці, в яку записано дані.
key.converter	io.confluent.connect.json.JsonSchemaConverter	Конвертер ключів, який використано для перетворення ключів повідомлень у формат даних JSON Schema
key.converter.schema.registry.url	http://schema-registry:8081	URL (посилання) реєстру схем для конвертера ключів
value.converter	io.confluent.connect.json.JsonSchemaConverter	Конвертер значень, який використано для перетворення значень повідомлень у формат даних JSON Schema
value.converter.schema.registry.url	http://schema-registry:8081	URL (посилання) реєстру схем для конвертера значень.
pk.fields	id	Поля, які використано як первинні ключі
pk.mode	record_key	Режим використання ключа запису в якості первинного ключа
auto.create	true	Автоматичне створення таблиці, якщо вони потрібні
auto.evolve	true	Автоматична зміна структури таблиць
insert.mode	upsert	Режим вставки (оновлення запису)
delete.enabled	true	Увімкнення можливості видалення записів

Отже, JDBC Sink-конектор створено для забезпечення надійного та автоматичного перенесення даних з Kafka у БД PostgreSQL.

JDBC Sink-конектор отримує дані з топиків Kafka, які містять інформацію з усіх БПА, і передає їх до центру керування (загальної БД).

Використання JDBC Sink-конектора дозволяє оперативно реагувати на зміни. Цей конектор автоматично створює і оновлює структуру таблиць у базі даних відповідно до схеми повідомлень у Kafka.

Створення конекторів і топиків для передачі інформації з центру керування (загальної бази даних) до баз даних безпілотних апаратів

Передача даних з загальної БД (центра керування) до серверу з Kafka забезпечено іншим Debezium Source-конектором, застосування якого повністю аналогічно роботі Debezium Source-конектору, який використовувався для передачі даних з БД БПНА до серверу з Kafka (таблиця 4.8).

Результатом застосування Debezium Source-конектора для передачі даних з загальної БД (центру керування) до сервера з Kafka є отримання топіку postgresMain.public.DroneInfo. Цей топик використано для передачі даних до обох БПА.

Створення Debezium Source-конекторів і JDBC Sink-конекторів забезпечують надійну двонаправлену синхронізацію обміну даними між обома БПА та центром керування.

4.4 Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж

Далі наведено детальний опис елементів схеми реалізації інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Схема реалізації інформаційної технології представлена на рисунку 4.12 та у додатку Ж.

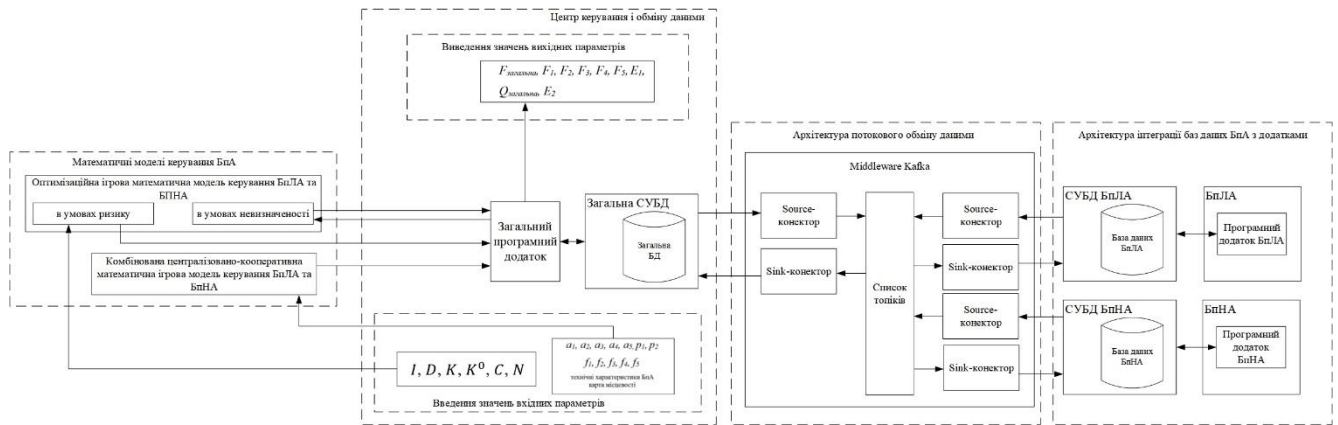


Рисунок 4.12 – Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж

1. Введення значень вхідних параметрів:

- а) технічні характеристики коаліції БпА;
- б) карта місцевості;
- в) $a_1, a_2, a_3, a_4, a_5, p_1, p_2, f_1, f_2, f_3, f_4, f_5$ – повний опис наведено у підрозділі 2.3 дисертаційного дослідження;
- г) I, D, K, K^0, C, N – повний опис наведено у підрозділі 2.5 дисертаційного дослідження.

2. Математичні моделі керування БпА:

а) комбінована централізовано-кооперативна математична ігрова модель керування БпЛА та БпНА (її застосування) сприяє успішному виконанню місії з максимальним покриттям території, швидкістю, корисним навантаженням з найменшими витратами часу та ресурсів (людських, фінансових та енергетичних);

б) оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності побудована для отримання оптимального маршруту руху БпА з урахуванням невизначеності в появі різного виду фізичних перешкод та небажаних подій, наявність яких необхідно було врахувати в процесі моделювання, а саме:

- в умовах ризику (сезонне бездоріжжя, метеорологічні фактори, несприятливі умови виконання місії). Моделювання небажаного явища

або перешкоди виконано з використанням апарату теорії ймовірностей та статистичних спостережень;

– в умовах невизначеності (раптові перешкоди). Моделювання подолання раптових та великих статичних перешкод виконано з використанням теорії ігор, теорії графів у поєднанні з можливостями нейронної мережі.

3. Виведення значень вихідних параметрів:

а) $F_{\text{загальна}}, F_1, F_2, F_3, F_4, F_5, E_1$ – повний опис наведено у підрозділі 2.3 дисертаційного дослідження;

б) $Q_{\text{загальна}}, E_2$ – повний опис наведено у підрозділі 2.5 дисертаційного дослідження.

4. Центр керування і обміну даними складається з:

а) загального програмного додатку – програмного компоненту взаємодії оператора з системою. Загальний програмний додаток містить програмне забезпечення керування БпА, створене на підставі розроблених математичних моделей у ході дисертаційного дослідження;

б) загальної СУБД – програмного комплексу для керування загальною БД. Загальна СУБД містить програмні засоби отримання та внесення змін до БД, а також взаємодіє з архітектурою потокового обміну даними.

5. Архітектура потокового обміну даними містить Middleware Kafka:

а) source-конектори забезпечують підключення Middleware Kafka до БД з метою отримання поточкових даних;

б) sink-конектори забезпечують підключення Middleware Kafka до БД з метою передачі поточкових даних;

в) список топіків містить всі повідомлення, отримані з усіх БД.

6. Архітектура інтеграції баз даних з БпА містить:

а) програмний додаток БпНА, тобто програмний компонент взаємодії БпНА з БД; збирає та обробляє дані про навколишнє середовище з різних датчиків БпНА; враховує великі статичні перешкоди;

б) програмний додаток БпЛА, тобто програмний компонент взаємодії БпЛА з БД; збирає та обробляє дані про навколишнє середовище з різних датчиків БпЛА; враховує великі статичні перешкоди.

в) СУБД БпНА, програмний комплекс, який використовується для отримання, передачі та збереження даних БпНА;

г) СУБД БпЛА, програмний комплекс, який використовується для отримання, передачі та збереження даних БпЛА.

4.5 Висновки до розділу 4

У четвертому розділі дисертаційного дослідження багато уваги приділено питанню обробки та збереження даних стосовно керування безпілотними апаратами.

В процесі проєктування інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж використані переваги програмного забезпечення Apache Kafka, оскільки вибір Apache Kafka забезпечує можливість обробляти та надійно зберігати значну кількість повідомлень.

Надано опис структури запропонованого в процесі наукового дослідження програмного забезпечення для роботи з БД центра керування і БД БпА.

Крім того, в якості інструменту проєктування багатоварової архітектури взаємодії з БД використана мова програмування Python та бібліотека FastAPI, що надало такі переваги: можливість працювати з декількома екземплярами баз даних; спрощене налагодження програми.

Розроблено діаграму класів для інформаційної технології керування безпілотними апаратами.

Ретельно опрацьовано питання забезпечення системи обміну даними при керуванні безпілотними апаратами, а також питання інтеграції Apache Kafka у систему інформаційної технології керування БпА.

При виконанні дисертаційного дослідження обмін даними відбувався із застосуванням хмарного серверу, на якому була встановлена Apache Kafka.

В процесі проєктування інформаційної технології керування БПА на базі ігрового підходу та нейронних мереж побудовано модифікований образ Kafka Connect.

Модифікований образ містить:

1. `cp-server-connect` – базовий образ для контейнера;
2. `kafka-connect-jdbc` – забезпечує підключення до реляційних баз даних;
3. `CONNECT_PLUGIN_PATH` – налаштовує місця пошуку плагінів;
4. `debezium-connector-postgresql` – відстежує зміни в базах даних PostgreSQL;
5. `kafka-connect-transform-common` – перетворює деякі типи даних (за запитом).

У дисертаційному дослідженні створено конектори і топіки для передачі інформації з БПА до загальної бази даних (і навпаки) за допомогою Apache Kafka.

В процесі наукового дослідження сформовані Source-конектори, які забезпечили передачу даних з безпілотних апаратів (БПА) на сервер з Apache Kafka.

Для проєктування інформаційної технології особливе значення має використання спеціалізованої бібліотеки конекторів Debezium.

Детально наведені етапи передачі даних до загальної бази даних.

Надано опис проведеного створення конекторів і топіків для передачі інформації з центру керування (загальної бази даних) до баз даних безпілотних апаратів.

Створення Debezium Source-конекторів і JDBC Sink-конекторів забезпечують надійну двонаправлену синхронізацію обміну даними між обома БПА та центром керування.

Розроблено схему інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

ВИСНОВКИ

Дисертаційна робота «Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж» – нове актуальне наукове дослідження в розробці інформаційних технологій.

В дисертаційній роботі отримано нове вирішення науково-технічної задачі підвищення ефективності виконання пошуково-рятувальних операцій та доставки вантажів безпілотними апаратами з пункту відправки до місця призначення шляхом розробки нової інформаційної технології керування безпілотними апаратами та їх взаємодії в умовах ризику та невизначеності, а також у злагодженому функціонуванні запропонованого програмно-технічного комплексу.

Наукове дослідження відрізняється не тільки новизною щодо методів проєктування інформаційних технологій, але й актуальністю та зростаючою потребою щодо розробки таких новітніх технологій стосовно використання безпілотних апаратів. Особливо це стосується пошуково-рятувальних операцій, а також доставки життєво-необхідних вантажів у важкодоступні місця.

В науковому дослідженні враховано необхідність успішного виконання поставленої задачі в умовах обмеженості ресурсів, зокрема, людських, фінансових та енергетичних.

Саме тому розроблена нова інформаційна технологія взаємодії безпілотних апаратів (БпА) різних типів (наземного та літального) із застосуванням нейронних мереж в умовах ризику та невизначеності, реалізація якої виконана на базі ігрового підходу, нейронних мереж, а також у злагодженому функціонуванні запропонованого програмно-технічного комплексу.

Програмно-технічний комплекс запропоновано використовувати у такому складі:

1. БпЛА;
2. БпНА;
3. Центр керування;
4. Оператор;

5. Штучна нейронна мережа, навчена для виконання задачі розпізнавання перешкод;

6. Алгоритм пошуку найкоротшого шляху A-Star;

7. GPS;

8. Розроблене унікальне програмне забезпечення (ПЗ) відповідно до нової комбінованої централізовано-кооперативної математичної моделі керування БПА різних типів (БПЛА та БПНА);

9. Розроблене унікальне ПЗ відповідно до нового комбінованого методу оптимізації маршруту руху БПЛА та БПНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

У першому розділі дослідження виконано стислий огляд сучасного стану розвитку інформаційних технологій.

Виконано аналіз позитивних та проблемних аспектів практичної реалізації розглянутих наукових досягнень.

Проведено огляд актуального стану інформаційних технологій керування безпілотними апаратами із застосуванням ігрового підходу та нейронних мереж.

Розглянуті різні типи нейронних мереж, їх застосування у різноманітних сферах життя; використання нейронних мереж у поєднанні з навчанням з підкріпленням.

Зроблено висновок, що використання нейронних мереж у поєднанні з ігровими методами має перспективу розвитку для підвищення безпеки виконання місії безпілотними апаратами.

У другому розділі виконаного наукового дослідження здійснено огляд, порівняння, аналіз існуючих методів, надано обґрунтування обрання апарату дослідження, який застосовано в процесі проєктування інформаційної технології керування безпілотними апаратами (БПА); отримано нові математичні моделі та визначена архітектура інформаційної системи.

Для обрання способів побудови інформаційної технології системи керування БПА з використанням нейронних мереж та ігрових підходів надано обґрунтування

доцільності (перспективності) використання кооперативного ігрового методу у керуванні БпА.

Проведено також огляд і порівняльний аналіз централізованого і децентралізованого методів керування БпА, окремо виявлено переваги та недоліки кожного з методів.

Зроблено висновок, що централізований метод керування коаліцією БпА у порівнянні з децентралізованим має такі переваги:

- контроль: весь контроль за виконанням місії (місій) покладається на одного оператора;
- простота: відсутність запитів щодо розробки складних алгоритмів автономії;
- безпека: оператор приймає всі рішення одноосібно.

Враховуючи переваги централізованого методу над децентралізованим, в науковому дослідженні обрано централізований метод керування БпА.

В процесі дослідження побудовані дві математичні моделі.

Розроблена комбінована централізовано-кооперативна математична ігрова модель керування БпА.

В процесі побудови цієї моделі використано підходи і апарат ігрового методу. В якості загальної функції виграшу обрано функцію, яка визначає головну ціль гри – успішне виконання місії від її початку і до кінця. Результатом моделювання є знаходження максимуму загальної функції виграшу.

Розраховано ефективність комбінованої централізовано-кооперативної математичної ігрової моделі керування БпА.

Ефективність комбінованої централізовано-кооперативної математичної ігрової моделі у порівнянні з ідеальною становить 71,12%. Ефективність децентралізовано-кооперативної моделі становить від 22,66% до 65% у порівнянні з ідеальною.

Під ідеальною розглядалася модель, де за замовчуванням при успішному виконанню місії енергетичне споживання та фінансові витрати дорівнюють нулю.

Практичне застосування комбінованої централізовано-кооперативної математичної ігрової моделі дозволить при виконанні місії двома БпА (наземним та літальним):

- 1) обрати ігровий підхід;
- 2) обрати централізовано-кооперативний метод керування БпА;
- 3) з метою найефективнішого виконання місії визначитися з вибором найбільш підходящих технічних характеристик коаліції БпЛА та БпНА.

Побудована оптимізаційна ігрова математична модель керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

В науковому дослідженні детально розглянуто підхід до вирішення проблеми обмеженості та/або неточності вхідних даних, відповідно до якого розглядаються два різних види ситуацій, які зазвичай необхідно дослідити, а саме: 1 – рішення приймаються в умовах ризику; 2 – рішення приймаються в умовах невизначеності.

Якщо рішення приймаються в умовах ризику, то появу небажаної перешкоди моделюють із застосуванням апарату теорії ймовірностей.

Якщо рішення приймається в умовах невизначеності, функцію розподілу ймовірностей знайти практично неможливо.

В науковому дослідженні розглядаються обидві ситуації, які виникають при моделюванні керування БпЛА та БпНА: 1 – ризик появи небажаного явища, зокрема весняного/осіннього бездоріжжя; 2 – невизначеність – поява раптової перешкоди (ями) в процесі виконання місії.

Результат застосування першої частини оптимізаційної ігрової математичної моделі керування БпЛА та БпНА в умовах ризику – отримання шляху з виключеними проблемними ребрами (ділянками).

Друга частина моделі вирішує проблему подолання раптової перешкоди (невизначеності) на шляху в процесі виконання місії. Запропоновано вирішувати цю проблему за допомогою можливостей нейронних мереж та навчання з підкріпленням.

У разі, коли раптову перешкоду подолати неможливо і навчання з підкріпленням не допомогло, поточні координати та маршрут руху, виявлений як

неможливий, фіксуються в базі даних БпНА та передаються до бази даних БпЛА. У цьому разі алгоритм прокладання маршруту виконується БпЛА повторно.

Проведено аналіз ефективності оптимізаційної ігрової математичної моделі керування БпЛА та БпНА із застосуванням нейронних мереж в умовах ризику та невизначеності.

По відношенню до ідеальної ефективність моделі, яка не враховувала ризики та невизначеність, становила всього 16,1%; ефективність моделі, яка враховувала тільки ризики, становила 39,9%; ефективність моделі, яка враховувала і ризики, і невизначеність, становила 73,2%.

Отже, найбільшу ефективність з розглянутих реальних моделей має модель, отримана в результаті наукового дослідження, яка враховувала і ризики, і невизначеність.

У третьому розділі проведено порівняльний аналіз відомих симуляторів, які необхідні для проведення практичних експериментів. Розглянуто найпоширеніші симулятори (Ardupilot, PX4, ROS, Gazebo, Microsoft AirSim), а також проаналізовані їхні переваги та недоліки.

Для проведення досліджень та експериментів обрано симулятор Microsoft AirSim.

Запропоновані в науковому дослідженні модифікації симулятора Microsoft AirSim наступні:

- прибрано центральний контролер (Flight Controller Firmware) та окремий обчислювальний мікрокомп'ютер (Companion Computer);
- додано нові фізичні контролери (Physical Controllers);
- додано блок вбудованих алгоритмів (Built-in algorithms);
- налагоджено взаємодію вбудованими алгоритмами з API Layer;
- збережено симуляційний модуль, який залишився без змін, у тому числі, блоки Vehicle Model, Environment Model, Sensor Models, Physics Engine та Rendering Engine.

Для проведення експериментів наукового дослідження запропоновано використання двох типів БпА (наземного та літального), які вбудовані у Microsoft AirSim.

В дисертаційному дослідженні наведено огляд різноманітних середовищ, які дозволяють використовувати обраний симулятор та які потенційно можна використовувати для перевірки ігрового підходу і нейронних мереж у керуванні безпілотними апаратами.

Обрано комбінований підхід, який дозволяє поєднати швидкість розпізнавання MobileNet з точністю виявлення об'єктів за допомогою SSD та перевагами трансферного навчання.

Проведені експерименти склалися з навчання згорткової нейронної мережі розпізнавати великі статичні перешкоди за допомогою трансферного навчання.

Відповідно до отриманих наукових результатів запропонована у дослідженні комбінована згорткова нейронна мережа ефективно розпізнала нові об'єкти, а саме – великі статичні блоки та раптові перешкоди (вибоїни, ями).

Підвищено загальну ефективність виявлення перешкод у нічний час на 20,87%, в умовах туману на 17,95%.

Розраховано загальну ефективність виявлення перешкод в різних умовах:

- денний час – 93,75%;
- нічний час – 85,94%;
- в умовах туману – 71,8%.

Навчання з підкріпленням застосовано в науковому дослідженні для успішного подолання великих статичних та раптових перешкод наземним БпА.

У четвертому розділі дисертаційного дослідження багато уваги приділено питанню обробки та збереження даних стосовно керування безпілотними апаратами.

Надано опис структури запропонованого в процесі наукового дослідження програмного забезпечення для роботи з БД центра керування і БД БпА.

Крім того, в якості інструменту проєктування багатошарової архітектури взаємодії з БД використана мова програмування Python та бібліотека FastAPI.

Розроблено діаграму класів для інформаційної технології керування безпілотними апаратами.

В процесі дослідження побудовано модифікований образ Kafka Connect.

Модифікований образ містить:

1. `cp-server-connect` – базовий образ для контейнера;
2. `kafka-connect-jdbc` – забезпечує підключення до реляційних баз даних;
3. `CONNECT_PLUGIN_PATH` – налаштовує місця пошуку плагінів;
4. `debezium-connector-postgresql` – відстежує зміни в базах даних PostgreSQL;
5. `kafka-connect-transform-common` – перетворює деякі типи даних (за запитом).

Розроблено схему інформаційної технології керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Результати наукового дослідження мають як теоретичне, так і практичне значення.

Наукове значення отриманих результатів полягає в отриманні нових методів та математичних моделей для проєктування інформаційних технологій керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

На основі одержаних в результаті дослідження алгоритмів розроблено програмне забезпечення і проведено комп'ютерне моделювання різних можливих ситуацій. Запропоновані методи і алгоритми можуть стати основою для керування взаємодією безпілотних апаратів.

Характерною особливістю запропонованого дослідження є використання ігрового та ймовірнісного підходів до керування об'єктами із застосуванням нейронних мереж в умовах ризику та невизначеності. Такий підхід до керування БПА забезпечує досягнення найкращого результату при несприятливих умовах. Використання ігрового підходу з ймовірнісним є найбільш ефективним для безпілотних апаратів, при виконанні задач пошуку та порятунку людей.

СПИСОК ЛІТЕРАТУРИ

1. Плескач, В. Л., & Затонацька, Т. Г. (2011). Інформаційні системи і технології на підприємствах. Київ: Знання. 718 с.
2. Васа, Т., Loiano, G., & Saska, M. (2016). Embedded model predictive control of unmanned micro aerial vehicles. In Proceedings of the 21st International Conference on Methods and Models in Automation and Robotics (MMAR) (pp. 992–997). Miedzyzdroje, Poland: IEEE. <https://doi.org/10.1109/MMAR.2016.7575273>.
3. Fan, Y. (2022). Flight control system simulation for quadcopter unmanned aerial vehicle (UAV) based on Matlab Simulink. In School of Mechatronic Engineering and Automation, Shanghai University, (pp. 1–8). <https://doi.org/10.1088/1742-6596/2283/1/012011>.
4. Дружинін, Є. А., Ковалевський, М. І., Погудіна, О. К., & Черановський, В. О. (2021). Методи та інформаційні технології впровадження безпілотних літальних апаратів в повітряний простір України. Системи обробки інформації (Вип. 4(68), с. 84–90). <https://doi.org/10.30748/soivt.2021.68.12>.
5. Mebarki, R., Lippiello, V., & Siciliano, B. (2015). Nonlinear visual control of unmanned aerial vehicles in GPS-denied environments. In IEEE Transactions on Robotics (Vol. 31, Issue 5, pp. 1004–1017). <https://doi.org/10.1109/TRO.2015.2451371>.
6. Oneata, D., & Cucu, H. (2019). Kite: Automatic speech recognition for unmanned aerial vehicles. In Proceedings of Interspeech 2019 (pp. 2998–3002). <https://doi.org/10.21437/Interspeech.2019-1390>.
7. Zhao, Z., Zhang, Y., Long, L., Lu, Z., & Shi, J. (2022). Efficient and adaptive lidar–visual–inertial odometry for agricultural unmanned ground vehicle. In International Journal of Advanced Robotic Systems (Vol. 19, pp. 1–13). <https://doi.org/10.1177/17298806221094925>.
8. Лисенко, С. М., & Румянцев, С. В. (2020). Інтелектуалізована система керування безпілотними літальними апаратами. Computer Systems and Information Technologies (Вип. 1, с. 22–27). <https://csitjournal.khmnu.edu.ua/index.php/csit/article/view/7/3>.

9. Барановська Л. В. (2022). ТЕОРІЯ ІГОР: КУРС ЛЕКЦІЙ. URL: https://ela.kpi.ua/bitstream/123456789/49092/1/Teoriia_igor.pdf.
10. Корнієнко В. О., Денисюк С. Г., Шиян А. А. Теорія некооперативних ігор. URL: <https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/12181/Теорія%20некооперативних%20ігор.pdf>.
11. Глушков В. (1973). Енциклопедія кібернетики. Київ: Вид. Академія Наук. 582 с.
12. Yang, Z. R., & Yang, Z. (2014). Comprehensive Biomedical Physics. Karolinska Institute, Stockholm, Sweden. In Elsevier. ISBN 978-0-444-53633-4. <https://doi.org/10.1016/B978-0-444-53632-7.01101-1>.
13. Субботін, С. О. (2020). Нейронні мережі: теорія та практика: навч. посіб. Житомир: Вид. О. О. Євенок. 184 с.
14. Prathap P. (2023). Feed-forward and Recurrent Neural Networks: The Future of Machine Learning. URL: <https://medium.com/@prajeeshprathap/feed-forward-and-recurrent-neural-networks-the-future-of-machine-learning-8b2c1975f0c5>.
15. Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent Advances in Recurrent Neural Networks. URL: <https://arxiv.org/abs/1801.01078>.
16. Mohebbali, B., Tahmassebi, A., Meyer-Baese, A., & Gandomi, A. H. (2020). Probabilistic neural networks: a brief overview of theory, implementation, and application. In Samui, P., Bui, D. T., Chakraborty, S., & Deo, R. C. In (Eds.), Handbook of Probabilistic Models (pp. 347–367). <https://doi.org/10.1016/B978-0-12-816514-0.00014-X>.
17. Purwono, P., Ma'arif, A., Rahmانيar, W., Fathurrahman, H. I. K., Frisky, A. Z. K., & Mazhar ul Haq, Q. (2022). Understanding of Convolutional Neural Network (CNN): A Review. In International Journal of Robotics and Control Systems (Vol. 2, Issue 4, pp. 739–748). <https://doi.org/10.31763/ijrcs.v2i4.888>.
18. Huang, G.-B. (2024). Dynamic neural networks: advantages and challenges. In National Science Review (Vol. 11). <https://doi.org/10.1093/nsr/nwae088>.
19. Coloma-Salazar, M. E., Arzola-Ruiz, J., Marrero-Fornaris, C.-E., Socha, V., & Asgher, U. (2024). A Combinatorial Approach for Optimizing Transportation System:

- Multi-Objective Decision-Making Framework. In *Neural Network World* (Vol. 34, pp. 135–168). <https://doi.org/10.14311/NNW.2024.34.008>.
20. Kim, S., Jung, S., Son, Y., & Lee, Y. (2024). A Study on the Security Weakness Detection of Solidity Smart Contracts using Graph Neural Networks on Blockchain Platforms. In *Journal of Machine and Computing*. <https://doi.org/10.53759/7669/jmc202505019>.
21. Ryabko, A. V., Vakaliuk, T. A., Zaika, O. V., Kukharchuk, R. P., Osadchyi, V. V., & Novitska, I. V. (2023). Methodology for Assessing the Quality of an Educational Program and Educational Activities of a Higher Education Institution Using a Neural Network. In *Proceedings of the 2nd Myroslav I. Zhaldak Symposium on Advances in Educational Technology* (pp. 179–198). ISBN: 978-989-758-662-0. <https://doi.org/10.5220/0012062800003431>.
22. Наливайко, О. О. (2023). Перспективи використання нейромереж у вищій освіті України. *Інформаційні технології і засоби навчання* (Том 97, № 5). <https://doi.org/10.33407/itlt.v97i5.5322>.
23. Bhosale, R., & Madgule, M. (2024). Optimization techniques for material selection and manufacturing processes: a review. In *JMST Advances*. <https://doi.org/10.1007/s42791-024-00093-x>.
24. Qavi, I., Halder, S., & Tan, G. (2024). Optimization of printability of bioinks with multi-response optimization (MRO) and artificial neural networks (ANN). In *Progress in Additive Manufacturing*. <https://doi.org/10.1007/s40964-024-00828-1>.
25. Yang, C. H., Chen, Y. L., Cheung, T. H., & Chuang, L. Y. (2024). Multi-Objective Optimization Accelerates the De Novo Design of Antimicrobial Peptide for *Staphylococcus aureus*. In *International Journal of Molecular Sciences* (Vol. 25). <https://doi.org/10.3390/ijms252413688>.
26. Ogbuanya, Ch. E., Han, F., Metha, S., & Ling, Q.-H. (2023). Enhancement of multiobjective Darwinian particle swarm optimization for neural-network-based multimodal medical image fusion. School of Computer Science, Jiangsu University of Science and Technology. URL: <https://ssrn.com/abstract=4924566>.

27. AlShafeey, M., & Rashdan, O. (2024). Genetic modification optimization technique: A neural network multi-objective energy management approach. In *Energy and AI* (Vol. 18). <https://doi.org/10.1016/j.egyai.2024.100417>.
28. Собчук, А. В., & Олімпієва, Ю. І. (2020). Застосування нейромереж для забезпечення функціональної стійкості виробничих процесів. *Телекомунікаційні та інформаційні технології* (Вип. 2(67), с. 13–25). <https://doi.org/10.31673/2412-4338.2020.021328>.
29. Montalbano, N. G., & Humphreys, T. E. (2020). Intercepting Unmanned Aerial Vehicle Swarms with Neural-Network-Aided Game-Theoretic Target Assignment. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)* (pp. 36–43). <https://doi.org/10.1109/PLANS46316.2020.9110234>.
30. Jaiton, V., Rothomphiwat, K., Ebeid, E., & Manoonpong, P. (2022). Neural Control and Online Learning for Speed Adaptation of Unmanned Aerial Vehicles. *Frontiers in Neural Circuits* (Vol. 16). <https://doi.org/10.3389/fncir.2022.839361>.
31. Li, Y., Li, H., Li, Z., Fang, H., Sanyal, A., Wang, Y., & Qiu, Q. (2019). Fast and Accurate Trajectory Tracking for Unmanned Aerial Vehicles based on Deep Reinforcement Learning. In *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)* (pp. 1–9). <https://doi.org/10.1109/RTCSA.2019.8864571>.
32. Mandloi, Y., & Inada, Y. (2019). Machine Learning Approach for Drone Perception and Control. In *Proceedings of the International Conference on Machine Learning* (pp. 424–431). https://doi.org/10.1007/978-3-030-20257-6_36.
33. Shiri, H., Park, J., & Bennis, M. (2019). Massive Autonomous UAV Path Planning: A Neural Network Based Mean-Field Game Theoretic Approach. In *2019 IEEE Global Communications Conference (GLOBECOM)* (pp. 1–6). <https://doi.org/10.1109/GLOBECOM38437.2019.9013181>.
34. Semsar-Kazerooni, E., & Khorasani, K. (2009). Multi-agent team cooperation: A game theory approach. *Automatica* (Vol. 45, pp. 2205–2213). <https://doi.org/10.1016/j.automatica.2009.06.006>.

35. Alpert M. (2024). Using Game Theory to Improve Drone Operations. In *Control Systems and Computers* (Vol. 1, pp. 57-61). <https://doi.org/10.15407/csc.2024.01.057>.
36. Альперт М. І., Онищенко В.В. (2023). Пошук найкращого алгоритму найкоротшого шляху для розумної валізи. *Управління розвитком складних систем* (Вип. 55, с. 92-97). <http://dx.doi.org/10.32347/2412-9933.2023.55.92-97>.
37. Alpert M., Onyshchenko V. (2022). Recognition of Potholes with Neural Network Using Unmanned Ground Vehicles. In *Advances in Computer Science for Engineering and Education* (Vol. 134., pp. 209–220). https://doi.org/10.1007/978-3-031-04812-8_18.
38. Літвінова Н., Альперт М., Погульський А. (2021). Підвищення ефективності обміну даними сутностей у реляційному представленні та їх обробки. *Технічні науки та технології* (Вип. 1(23), с. 81–86). [https://doi.org/10.25140/2411-5363-2021-1\(23\)-81-86](https://doi.org/10.25140/2411-5363-2021-1(23)-81-86).
39. Cummings, M. L. (2015). Operator Interaction with Centralized Versus Decentralized UAV Architectures. In Valavanis, K. P., & Vachtsevanos, G. J. (Eds.), *Handbook of Unmanned Aerial Vehicles* (pp. 977–991). Dordrecht: Springer. https://doi.org/10.1007/978-90-481-9707-1_117.
40. Таршин, В. А., Компанієць, О. М., Котляренко, С. Є., & Дужий, Р. В. (2023). Розвиток методології управління роєм БПЛА на основі ройового інтелекту. *Збірник наукових праць Державного науково-дослідного інституту авіації* (Вип. 19(26), с. 109-115). <https://doi.org/10.54858/dndia.2023-19-15>.
41. Taha, H. A. (2017). *Operations Research: An Introduction* 10th ed.. Pearson Education Limited. ISBN 978-1-292-16554-7. 900 p. URL: <https://industri.fatek.unpatti.ac.id/wp-content/uploads/2019/03/172-Operations-Research-An-Introduction-Hamdy-A.-Taha-Edisi-10-2017.pdf>.
42. Вашків, П. Г., Пастер, П. І., Сторожук, В. П., & Ткач, Є. І. (2001). *Теорія статистики: Навчальний посібник*. Київ: Либідь. 320 с.
43. Спекторський, І. Я. (2004). *Дискретна математика*. Київ: Либідь. URL: <http://mmsa.kpi.ua/sites/default/files/publications/Спекторський%20Ігор%20Якович/spektorskii-i-ya-diskretna-matematika-chastina-1-2.pdf>.


44. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). Cambridge: MIT Press. 1312 p.
45. Hua, C., Niu, R., Yu, B., Zheng, X., Bai, R., & Zhang, S. (2022). A global path planning method for unmanned ground vehicles in off-road environments based on mobility prediction. *Machines* (Vol. 10, Article 375, pp.1-22). <https://doi.org/10.3390/machines10050375>
46. Shi, G., Karapetyan, N., Asghar, A., Reddinger, J.-P., Dotterweich, J., Humann, J., & Tokekar, P. (2022). Risk-aware UAV-UGV rendezvous with chance-constrained Markov decision process. arXiv preprint. <https://doi.org/10.48550/arXiv.2204.04767>
47. Ardupilot. (2024). URL: <https://ardupilot.org/>.
48. PX4 Autopilot. (2024). URL: <https://px4.io/>.
49. PX4 Instability/Tuning Issue. (2022). URL: <https://discuss.px4.io/t/px4-instability-tuning-issue/29586>.
50. Getting Started With Robotic Operating System. (2014). URL: <https://www.instructables.com/Getting-Started-with-ROS-Robotic-Operating-Syste/>.
51. Gazebo. (2024). URL: <https://gazebo.org/docs/all/getstarted/>.
52. Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In FSR 2017 Conference. URL: <https://arxiv.org/abs/1705.05065>.
53. Bondi, E., Dey, D., Kapoor, A., Piavis, J., Shah, S., Fang, F., Dilkina, B., Hannaford, R., Iyer, A., Joppa, L., & Tambe, M. (2018). AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs. In COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies, June 20–22, 2018, Menlo Park and San Jose, CA, USA. New York, NY: ACM. <https://doi.org/10.1145/3209811.3209880>.
54. Kinasih, F. M. T. R., Machbub, C., Yulianti, L., & Rohman, A. S. (2023). Two-stage multiple object detection using CNN and correlative filter for accuracy improvement. *Heliyon* (Vol. 9, pp. 1-18). <https://doi.org/10.1016/j.heliyon.2022.e12716>.
55. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision (ECCV). https://doi.org/10.1007/978-3-319-46448-0_2.

56. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). URL: <https://arxiv.org/abs/1704.04861>.
57. West, J., Ventura, D., & Warnick, S. (2007). Spring Research Presentation: A Theoretical Foundation for Inductive Transfer. Brigham Young University, College of Physical and Mathematical Sciences (Vol. 1, Issue 8).
58. Fan, Z. (2024). An exploration of reinforcement learning and deep reinforcement learning. Applied and Computational Engineering (Vol. 73, pp. 154–159). <https://doi.org/10.54254/2755-2721/73/20240386>.
59. Кочура Ю. П. Навчання нейронних мереж з підкріпленням URL: <https://comsys.kpi.ua/upload/Навчання%20%20з%20підкріпленням%20.Конспект%20лекцій.pdf>.
60. Pliev A. (2021). How to (Safely) Train an Autonomous Drone in a game engine URL: <https://blog.ml6.eu/implementing-a-reinforcement-learning-agent-with-tf-agents-to-train-an-autonomous-drone-in-airsim-b1e85b5994a6>.
61. TensorFlow Agents (2023). URL: <https://www.tensorflow.org/agents/overview>.
62. Spryn, M., & Sharma, A. (2018). Autonomous Driving using End-to-End Deep Learning: an AirSim tutorial. URL: <https://github.com/Microsoft/AutonomousDrivingCookbook/tree/master/AirSimE2EDeepLearning>.
63. Анісімов, А. В., Завадський, І. О., Кулябко, П. П. (2024). Інформаційні системи та бази даних: Частина 2: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. Київ. 104 с.
64. Docker Docs. (2024). URL: <https://docs.docker.com/>.
65. Ramírez S. FastAPI framework (2024). URL: <https://fastapi.tiangolo.com/>.
66. Percival, H. (2017). Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript (2nd ed.). 624 p.

67. Quick Start for Confluent Platform. URL: <https://docs.confluent.io/platform/current/platform-quickstart.html>.
68. Martín, C., Langendoerfer, P., Zarrin, P. S., Díaz, M., & Rubio, B. (2022). Kafka-ML: Connecting the data stream with ML/AI frameworks. In *Future Generation Computer Systems* (Vol. 126, pp. 15–33). ISSN 0167-739X. <https://doi.org/10.1016/j.future.2021.07.037>.
69. Shapira, G., Palino, T., Sivaram, R., & Petty, K. (2021). *Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale*. O'Reilly Media.
70. Debezium connector for PostgreSQL. URL: <https://debezium.io/documentation/reference/stable/connectors/postgresql.html>.
71. JDBC Sink Connector for Confluent Platform. URL: <https://docs.confluent.io/kafka-connectors/jdbc/current/sink-connector/overview.html>.

ДОДАТОК А. АКТ ВПРОВАДЖЕННЯ У НАВЧАЛЬНИЙ ПРОЦЕС

Декан факультету інформатики
та обчислювальної техніки
КПІ ім. Сікорського

 Ярослав КОРНАГА
«10» есня 2025 року

АКТ

про впровадження в навчальний процес кафедри Інформаційних систем та технологій факультету Інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» результатів дисертаційної роботи Альперта Максима Іогановича «Інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж», поданої на здобуття наукового ступеня доктора філософії.

Ми, що нижче підписалися: завідувач кафедри Інформаційних систем та технологій, д.т.н., проф. Ролік О. І., вчений секретар кафедри, к.ф.-м.н., доц. Гавриленко О. В., професор кафедри, д.т.н., проф. Онищенко В.В підтверджуємо, що результати дисертаційної роботи Альперта М. І. включені в матеріали навчально-методичного забезпечення курсів «Прикладні задачі машинного навчання», «Системи штучного інтелекту» та «Штучний інтелект в задачах обробки зображень», а саме:

- комбінована централізовано-кооперативна математична ігрова модель керування безпілотними апаратами;
- оптимізаційна ігрова математична модель керування безпілотними апаратами із застосуванням нейронних мереж в умовах ризику та невизначеності;
- інформаційна технологія керування безпілотними апаратами на базі ігрового підходу та нейронних мереж.

Впровадження результатів дисертаційної роботи Альперта М. І. в навчальний процес дозволило підвищити якість підготовки студентів освітнього рівня «Бакалавр» зі спеціальності 126 «Інформаційні системи та технології» на кафедрі Інформаційних систем та технологій КПІ ім. Ігоря Сікорського.

Завідувач кафедри
інформаційних систем та технологій,
д.т.н., проф.



Олександр РОЛІК

Вчений секретар кафедри
інформаційних систем та технологій,
к.ф.-м.н., доц.



Олена ГАВРИЛЕНКО

Професор кафедри
інформаційних систем та технологій,
д.т.н., проф.



Вікторія ОНИЩЕНКО

ДОДАТОК Б. ЛІСТИНГ КОДУ КОМБІНОВАНОЇ ЦЕНТРАЛІЗОВАНО- КООПЕРАТИВНОЇ МОДЕЛІ КЕРУВАННЯ БПЛА ТА БПНА

```

import matplotlib.pyplot as plt
import numpy as np

# Вхідні дані
A1, A2 = 120, 80 # Покриття території (км²)
B1, B2 = 60, 30 # Швидкість (км/год)
C1, C2 = 5, 120 # Навантаження (кг)
u1, u2 = 3, 4 # Керуюча дія (умовні одиниці для енергії та фінансів)

# Параметри витрат
k1, k2 = 0.2, 0.3 # Коефіцієнти для фінансових витрат
c1, c2 = 50, 30 # Фіксовані фінансові витрати

# Пріоритети цільових функцій
a, b, c, d, e = 0.25, 0.1, 0.3, 0.15, 0.2
w1, w2 = 0.4, 0.6 # Пріоритети для апаратів

# Функція для нормалізації значень
def normalize(value, max_value):
    return value / max_value

# Нормалізація вхідних даних
A1_norm = normalize(A1, A1)
B1_norm = normalize(B1, B1)
C1_norm = normalize(C1, C1)
A2_norm = normalize(A2, A1)
B2_norm = normalize(B2, B1)
C2_norm = normalize(C2, C1)

# Функції розрахунку
def calc_coverage(A1_norm, A2_norm):
    return a * (w1 * 1 + w2 * A2_norm)

def calc_speed(B1_norm, B2_norm):
    return b * (w1 * 1 + w2 * B2_norm)

def calc_load(C1_norm, C2_norm):
    return c * (w1 * 1 + w2 * C2_norm)

def calc_energy(u1, u2):
    return d * (w1 * u1**2 + w2 * u2**2)

def calc_financial(u1, u2, k1, k2, c1, c2):
    return e * (w1 * (k1 * u1**2 + c1) + w2 * (k2 * u2**2 + c2))

# Базові значення

```

```

coverage = calc_coverage(A1, A2)
speed = calc_speed(B1, B2)
load = calc_load(C1, C2)
energy = calc_energy(u1, u2)
financial = calc_financial(u1, u2, k1, k2, c1, c2)
financial_2x = calc_financial(2 * u1, 2 * u2, k1, k2, c1, c2)
financial_5x = calc_financial(5 * u1, 5 * u2, k1, k2, c1, c2)

# Ідеальна цільова функція
J_ideal = coverage + speed + load

# Підсумок глобальної цільової функції
J_total = max(0, J_ideal - (energy + financial))

# Підсумок глобальної цільової функції з подвійними фін. витратами
J_total_inc_2x = max(0, J_ideal - (energy + financial_2x))

# Підсумок глобальної цільової функції з фін. витратами в 5 раз
J_total_inc_5x = max(0, J_ideal - (energy + financial_5x))

# Ефективність (обмежена діапазоном 0-100)
effectiveness = max(0, min((J_total / J_ideal) * 100 if J_ideal != 0 else 0,
100))

# Ефективність (обмежена діапазоном 0-100 з подвійними фін. витратами)
effectiveness_fin_2x = max(0, min((J_total_inc_2x / J_ideal) * 100 if J_ideal
!= 0 else 0, 100))

# Ефективність (обмежена діапазоном 0-100 з фін. витратами в 5 раз)
effectiveness_fin_5x = max(0, min((J_total_inc_5x / J_ideal) * 100 if J_ideal
!= 0 else 0, 100))

# Виведення результатів
print(f"Покриття території: {coverage:.2f}")
print(f"Швидкість: {speed:.2f}")
print(f"Навантаження: {load:.2f}")
print(f"Енергоспоживання: {energy:.2f}")
print(f"Фінансові витрати: {financial:.2f}")
print(f"Ідеальна цільова функція J_ideal: {J_ideal:.2f}")
print(f"Глобальна цільова функція J_total: {J_total:.2f}")
print(f"Ефективність центр: {effectiveness:.2f}%")
print(f"Ефективність дец 1: {effectiveness_fin_2x:.2f}%")
print(f"Ефективність дец 2: {effectiveness_fin_5x:.2f}%")

# Генерація значень керуючої дії u
u_values = np.linspace(1, 10, 100)
J_total_base = []
J_total_cov_fin_2x = []
J_total_fin_2x = []

```

```

J_total_cov_fin_5x = []
J_total_fin_5x = []

J_total_ideal = []

for u in u_values:
    energy = calc_energy(u, u)
    financial_base = calc_financial(u, u, k1, k2, c1, c2)
    financial_2x = calc_financial(2 * u, 2 * u, k1, k2, c1, c2)
    financial_5x = calc_financial(5 * u, 5 * u, k1, k2, c1, c2)

    # Базовий випадок
    J_total_base.append(max(0, J_ideal - (energy + financial_base)))

    # Ідеальний випадок
    J_total_ideal.append(max(0, J_ideal))

    # Тільки фінансові витрати у 2 рази
    # J_total_fin_2x.append(max(0, J_ideal - (energy + financial_2x)))
    J_fin_2x = J_ideal - (energy + financial_2x)
    J_total_fin_2x.append(J_fin_2x if J_fin_2x > 0 else np.nan) # Прибираємо
значення <= 0

    # Тільки фінансові витрати у 5 разів
    J_fin_5x = J_ideal - (energy + financial_5x)
    # J_total_fin_5x.append(max(0, J_ideal - (energy + financial_5x)))
    J_total_fin_5x.append(J_fin_5x if J_fin_5x > 0 else np.nan) # Прибираємо
значення <= 0

# Побудова графіків
plt.figure(figsize=(10, 6))
plt.plot(u_values, J_total_base, label="Централізовано-кооперативна
математична модель", color="red")
# plt.plot(u_values, J_total_cov_fin_2x, label="Покриття та фінансові витрати
у 2 рази", color="green")
plt.plot(u_values, J_total_fin_2x, label="Децентралізовано-кооперативна
математична модель 1", color="black")
plt.plot(u_values, J_total_fin_5x, label="Децентралізовано-кооперативна
математична модель 2", color="green")
# plt.plot(u_values, J_total_ideal, label="Ідеальна модель", color="black")

# Оформлення графіка
plt.title("Залежність загальної цільової функції від типу моделі") #""
Порівняння загальної цільової функції
plt.xlabel("Фінансові витрати")
plt.ylabel("Загальна цільова функція")
plt.legend()
plt.grid(True)
plt.show()

```

ДОДАТОК В. ЛІСТИНГ КОДУ ОПТИМІЗАЦІЙНОЇ ІГРОВОЇ МАТЕМАТИЧНОЇ МОДЕЛІ КЕРУВАННЯ БПЛА ТА БПНА В УМОВАХ РИЗИКУ ТА НЕВИЗНАЧЕНОСТІ

```

import networkx as nx
import random
import matplotlib.pyplot as plt

def calculate_path_length(graph, path):
    total_length = 0
    for i in range(len(path) - 1):
        total_length += graph.edges[path[i], path[i + 1]]["weight"]
    return total_length

# Функція для застосування ризиків
def apply_risks(graph, risk_factors):
    new_graph = graph.copy()
    for edge, risk in risk_factors.items():
        original_weight = graph.edges[edge]["weight"]
        new_graph.edges[edge]["weight"] = original_weight * risk
    return new_graph

# Функція для застосування невизначеностей
def apply_uncertainty(graph, risk_factors):
    new_graph = graph.copy()
    for edge in new_graph.edges():
        risk = risk_factors[edge]
        uncertainty = random.uniform(0.9, 1.2) # Стохастичний коефіцієнт
        original_weight = new_graph.edges[edge]["weight"]
        new_graph.edges[edge]["weight"] = original_weight * risk * uncertainty
    return new_graph

# Функція для пошуку найкоротшого шляху за A-Star
def a_star_path(graph, start, end):
    return nx.astar_path(graph, start, end, weight="weight")

# Візуалізація графів та шляху
def visualize_graphs(graph1, graph2, path1=None, path2=None, titles=("Граф 1",
"Граф 2")):
    import matplotlib.pyplot as plt
    import networkx as nx

    fig, axes = plt.subplots(1, 2, figsize=(20, 8)) # Два підграфіка (1 ряд,
    2 стовпця)

    # Перша візуалізація
    pos1 = nx.spring_layout(graph1)

```

```

weights1 = nx.get_edge_attributes(graph1, "weight")
nx.draw(graph1, pos1, with_labels=True, node_size=700,
node_color="lightblue", font_size=10, ax=axes[0])
nx.draw_networkx_edge_labels(graph1, pos1, edge_labels={k: round(v, 2) for
k, v in weights1.items()}, font_color="red", ax=axes[0])
if path1:
    path_edges1 = list(zip(path1, path1[1:]))
    nx.draw_networkx_edges(graph1, pos1, edgelist=path_edges1,
edge_color="green", width=3, ax=axes[0])
    axes[0].set_title(titles[0])

# Друга візуалізація
pos2 = nx.spring_layout(graph2)
weights2 = nx.get_edge_attributes(graph2, "weight")
nx.draw(graph2, pos2, with_labels=True, node_size=700,
node_color="lightblue", font_size=10, ax=axes[1])
nx.draw_networkx_edge_labels(graph2, pos2, edge_labels={k: round(v, 2) for
k, v in weights2.items()}, font_color="red", ax=axes[1])
if path2:
    path_edges2 = list(zip(path2, path2[1:]))
    nx.draw_networkx_edges(graph2, pos2, edgelist=path_edges2,
edge_color="green", width=3, ax=axes[1])
    axes[1].set_title(titles[1])

plt.tight_layout()
plt.show()

def create_fixed_weight_graph_smaller():
    graph = nx.DiGraph()

    # Вершини
    nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
    graph.add_nodes_from(nodes)

    # Ребра з фіксованими вагами
    edges_with_weights = [
        ('A', 'D', 3), ('A', 'B', 2), ('A', 'E', 4),
        ('B', 'C', 1), ('B', 'F', 5),
        ('C', 'D', 2), ('C', 'F', 3),
        ('E', 'F', 1),
        ('F', 'G', 2), ('F', 'H', 4),
        ('G', 'H', 1)
    ]

    for edge in edges_with_weights:
        graph.add_edge(edge[0], edge[1], weight=edge[2])

    return graph

```

```

def create_fixed_weight_graph():
    graph = nx.DiGraph()

    # Вершини
    nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
    graph.add_nodes_from(nodes)

    # Ребра з фіксованими вагами
    edges_with_weights = [
        ('A', 'D', 3), ('A', 'B', 2), ('A', 'E', 4),
        ('B', 'C', 1), ('B', 'F', 5),
        ('C', 'D', 2), ('C', 'F', 3),
        ('E', 'F', 1),
        ('F', 'G', 4), ('F', 'H', 4),
        ('G', 'H', 1)
    ]

    for edge in edges_with_weights:
        graph.add_edge(edge[0], edge[1], weight=edge[2])

    return graph

# Пошук оптимального шляху за алгоритмом A-Star
def find_fixed_weight_path(graph, start, end):
    # Знаходимо найкоротший шлях
    path = a_star_path(graph, start, end)

    # Формуємо послідовність ребер
    path_edges = [(path[i], path[i + 1]) for i in range(len(path) - 1)]

    # Обчислюємо довжину шляху
    total_length = calculate_path_length(graph, path)

    return path, path_edges, total_length

# Основна програма
def main_fixed_weight_graph():
    # Створення графа
    graph_smaller = create_fixed_weight_graph_smaller()
    graph = create_fixed_weight_graph()

    # Візуалізація графа
    visualize_graphs(graph_smaller, graph, titles=("Граф 1", "Граф 2"))
    # visualize_graph(graph, title="Орієнтований граф з урахуванням
невизначеності")

    # Пошук оптимального шляху
    start_node, end_node = 'A', 'H'
    path_smaller, path_edges_smaller, total_length_smaller =
find_fixed_weight_path(graph_smaller, start_node, end_node)

```

```

    path, path_edges, total_length = find_fixed_weight_path(graph, start_node,
end_node)

    print(f"Оптимальний шлях (вузли): {path_smaller}")
    print(f"Послідовність ребер: {path_edges_smaller}")
    print(f"Загальна довжина шляху: {total_length_smaller}")

    print(f"-----")
    print(f"Граф з урахуванням невизначеності")

    print(f"Оптимальний шлях (вузли): {path}")
    print(f"Послідовність ребер: {path_edges}")
    print(f"Загальна довжина шляху: {total_length}")

# Запуск програми
main_fixed_weight_graph()

```

ДОДАТОК Г. ЛІСТИНГ КОДУ ТРАНСФЕРНОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

```

!pip install --upgrade pip
!pip install --upgrade protobuf

%tensorflow_version 1.15
import tensorflow as tf
print(tf.__version__)

!pip install numpy

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
# XXX: only one GPU on Colab and isn't guaranteed
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize(
psutil.virtual_memory().available ), " | Proc size: " + humanize.naturalsize(
process.memory_info().rss))
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total
{3:.0f}MB".format(gpu.memoryFree, gpu.memoryUsed, gpu.memoryUtil*100,
gpu.memoryTotal))
printm()

from google.colab import drive
drive.mount('/content/gdrive')

# change to working tensorflow directory on the drive
%cd '/content/gdrive/My Drive/models/'

!apt-get install protobuf-compiler python-pil python-lxml python-tk
!pip install Cython
%cd /content/gdrive/My Drive/models/research/
!protoc object_detection/protos/*.proto --python_out=.

```

```

import os
os.environ['PYTHONPATH'] += ':/content/gdrive/My
Drive/models/research:/content/gdrive/My Drive/models/research/slim'

!python setup.py build
!python setup.py install

import time, psutil
Start = time.time()- psutil.boot_time()
Left= 12*3600 - Start
print('Time remaining for this session is: ', Left/3600)

!pip install tf_slim

%cd /content/gdrive/My Drive/models/research/object_detection/

os.environ['PYTHONPATH'] += ':/content/gdrive/My Drive/models/research/slim'

import sys
sys.path.append("/content/gdrive/My Drive/models")

!python train.py --train_dir=training/ --
pipeline_config_path=training/ssd_mobilenet_v1_pets.config --logtostderr

!pip install tensorboardcolab
%cd /content/gdrive/My Drive/models/research/object_detection/training
%reload_ext tensorboard
%tensorboard --logdir '/content/gdrive/My
Drive/models/research/object_detection/training'

# .ckpt needs to be updated every time to match last .ckpt generated
# .config needs to be updated when changing model
%cd /content/gdrive/My Drive/models/research/object_detection
!python export_inference_graph.py --input_type image_tensor --
pipeline_config_path training/ssd_mobilenet_v1_pets.config --
trained_checkpoint_prefix training/model.ckpt-7508 --output_directory
new_graph

!zip -r model_graph.zip new_graph

```

ДОДАТОК Д. ФРАГМЕНТИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КЕРУВАННЯ БЕЗПЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ

```

1  ---
2  version: '2'
3  services:
4    postgres1:
5      image: postgres:11
6      container_name: postgresDrone1
7
8      ports:
9        - 5435:5432
10     environment:
11       - POSTGRES_USER=postgres1
12       - POSTGRES_PASSWORD=postgres1
13       - POSTGRES_DB=postgresDrone1
14     command: ['postgres', '-c', 'wal_level=logical']
15
16
17     postgres2:
18       image: postgres:11
19       container_name: postgresDrone2
20
21       ports:
22         - 5436:5432
23       environment:
24         - POSTGRES_USER=postgres2
25         - POSTGRES_PASSWORD=postgres2
26         - POSTGRES_DB=postgresDrone2
27       command: ['postgres', '-c', 'wal_level=logical']
28
29     postgresMain:
30       image: postgres:11
31       container_name: postgresMain
32
33       ports:
34         - 5438:5432
35       environment:
36         - POSTGRES_USER=postgresMain
37         - POSTGRES_PASSWORD=postgresMain
38         - POSTGRES_DB=postgresMain
39       command: [ 'postgres', '-c', 'wal_level=logical' ]

```

Рисунок Д.1 – Створення баз даних PostgreSQL через Docker

```

1  create table "DroneInfo"
2  (
3      id                                TEXT                                NOT NULL
4          constraint "DroneInfo_pk"
5              primary key,
6      "droneId"                        TEXT,
7      timestamp                        TIMESTAMP DEFAULT now() NOT NULL,
8
9      -- Navigation and flight data
10     coordinates                      TEXT,
11     acceleration                      DOUBLE PRECISION,
12     "angularVelocity"                DOUBLE PRECISION,
13
14     -- Camera and optical sensor data
15     photo                            TEXT,
16     video                            TEXT,
17
18     -- Weather sensor data
19     temperature                      DOUBLE PRECISION,
20     "lightIntensity"                 DOUBLE PRECISION,
21
22     -- Technical data
23     "batteryLevel"                   INTEGER,
24     "transmissionSpeed"               DOUBLE PRECISION,
25     latency                          DOUBLE PRECISION,
26     "packetLoss"                     DOUBLE PRECISION,
27
28     -- Mission execution data
29     route                            TEXT,
30     tasks                            TEXT,
31     "actionLog"                      TEXT,
32     result                           TEXT,
33     "deviationCases"                 TEXT,
34     "emergencySituations"            TEXT,
35     "redirectedDroneId"              TEXT
36 )

```

Рисунок Д.2 – Створення таблиці «DroneInfo»

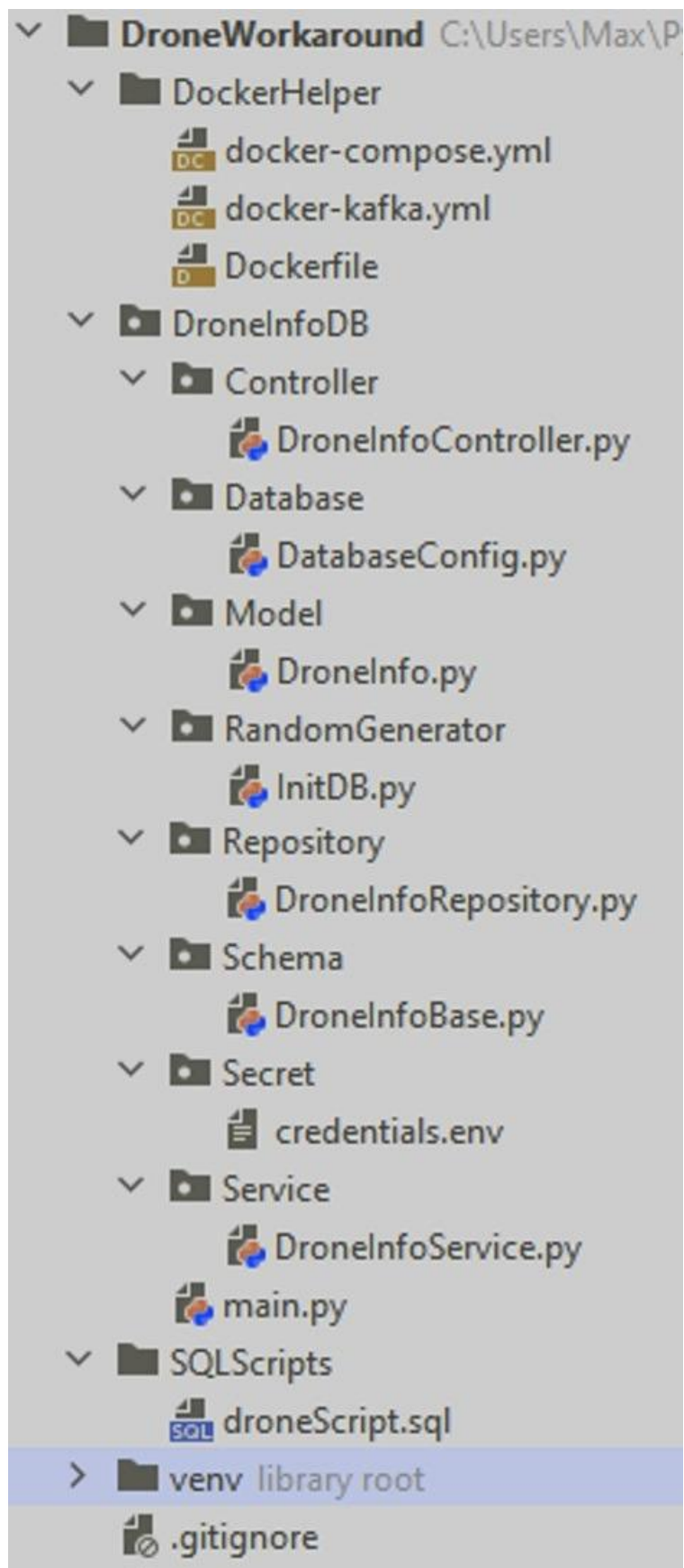


Рисунок Д.3 – Структура програмного рішення

```
FROM confluentinc/cp-kafka-connect:7.6.1
```

```
ENV CONNECT_PLUGIN_PATH="/usr/share/java,/usr/share/confluent-hub-components,/usr/share/java/kafka-connect-jdbc,/usr/share/java/kafka"
```

```
RUN confluent-hub install --no-prompt confluentinc/kafka-connect-jdbc:latest
```

```
RUN confluent-hub install --no-prompt debezium/debezium-connector-postgresql:latest
```

```
RUN confluent-hub install --no-prompt jcustenborder/kafka-connect-transform-common:0.1.0.54
```

Рисунок Д.4 – Скрипт для створення модифікованого образу Kafka Connect








<input type="checkbox"/>		dockerhelper		Running (6/6)	4.54%	
<input type="checkbox"/>		broker-drone	confluentinc/cp-kafka:7.6.1	Running	1.99%	9092:9092 Show all ports (2)
<input type="checkbox"/>		schema-registry-drone	confluentinc/cp-schema-registry:7.6.1	Running	0.42%	8081:8081 Show all ports (2)
<input type="checkbox"/>		control-center-drone	confluentinc/cp-enterprise-control-cent	Running	0.93%	9021:9021 Show all ports (2)
<input type="checkbox"/>		connect-drone	drone-kafka:1.0.0	Running	0.78%	8083:8083 Show all ports (2)
<input type="checkbox"/>		ksqldb-server-drone	confluentinc/cp-ksqldb-server:7.6.1	Running	0.4%	8088:8088 Show all ports (2)
<input type="checkbox"/>		rest-proxy-drone	confluentinc/cp-kafka-rest:7.6.1	Running	0.02%	8082:8082 Show all ports (2)

Рисунок Д.5 – Контейнери для роботи з Apache Kafka

Ethernet adapter Ethernet 2:

```

Connection-specific DNS Suffix  . :
Link-local IPv6 Address . . . . . : fe82::1cba:f23a:f659:8c15%22
IPv4 Address. . . . . : 192.168.0.131
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1

```

Рисунок Д.6 – Локальна ір адреса

POST ▼ http://localhost:8083/connectors/

Params Authorization Headers (8) **Body** ● Scripts Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```

1  {
2    "name": "postgres-debez-connector-v1",
3    "config": {
4      "tasks.max": 1,
5      "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
6      "plugin.name": "pgoutput",
7      "database.hostname": "192.168.0.131",
8      "database.port": "5435",
9      "database.user": "postgres1",
10     "database.password": "postgres1",
11     "database.dbname": "postgresDrone1",
12     "database.server.name": "server1",
13     "topic.prefix": "postgresDrone1",
14     "table.include.list": "public.(*)",
15     "database.history.kafka.bootstrap.servers": "broker:29092",
16     "database.history.kafka.topic": "postgresDrone1.debezium.history",
17     "database.history.skip.unparseable.ddl": "true",
18     "snapshot.mode": "always",
19     "time.precision.mode": "connect",
20     "key.converter": "io.confluent.connect.json.JsonSchemaConverter",
21     "key.converter.schema.registry.url": "http://schema-registry:8081",
22     "value.converter": "io.confluent.connect.json.JsonSchemaConverter",
23     "value.converter.schema.registry.url": "http://schema-registry:8081"
24   }

```

Рисунок Д.7 – Створення Debezium Source-конектора на прикладі БД БпНА

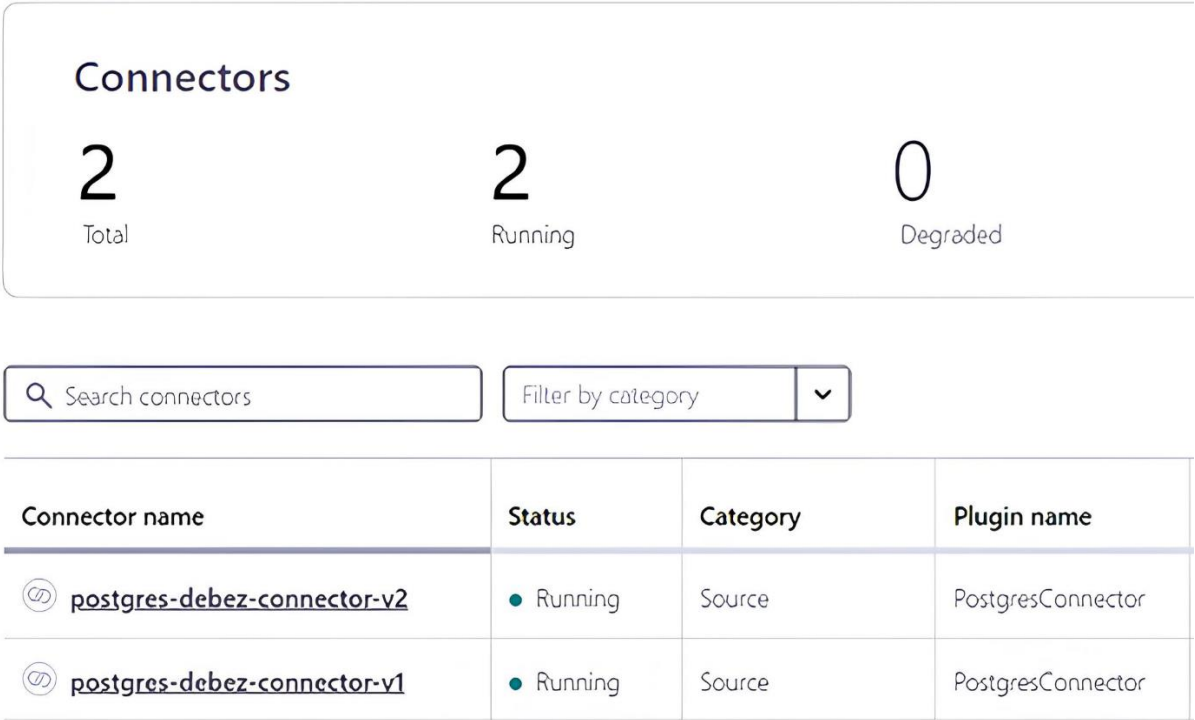


Рисунок Д.8 – Створені Debezium Source-конектори

Topics

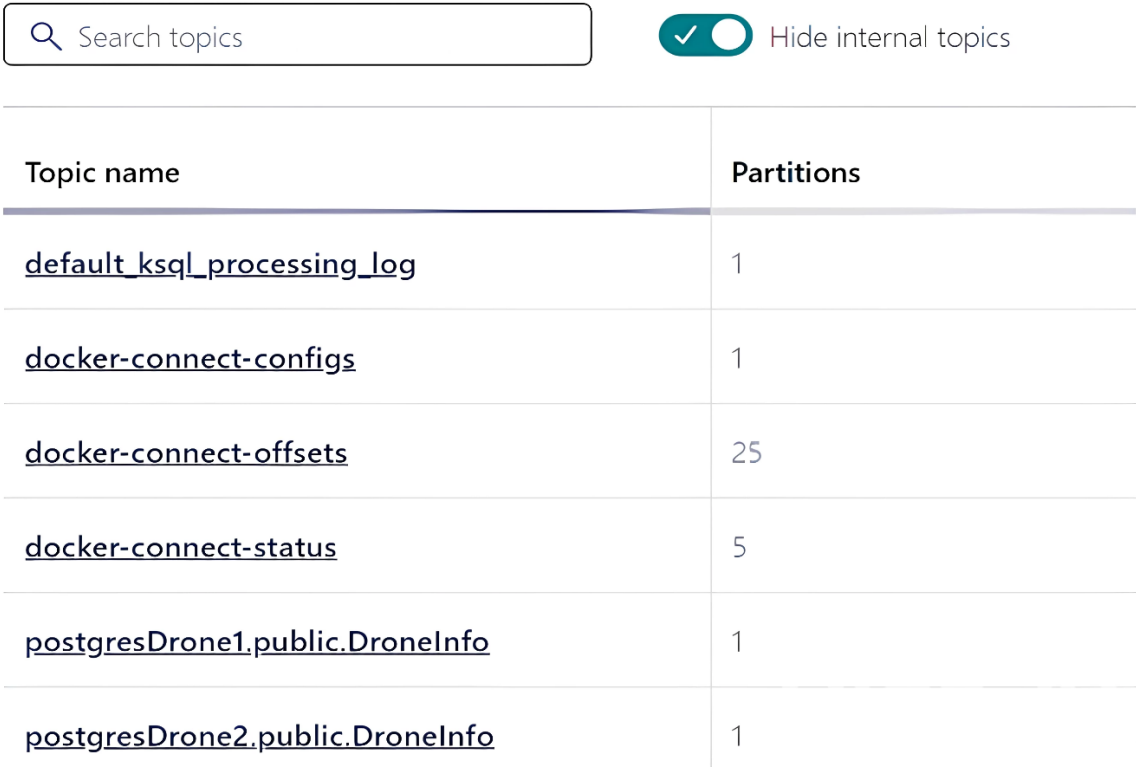


Рисунок Д.9 – Топіки, створені на основі Debezium Source-конекторів

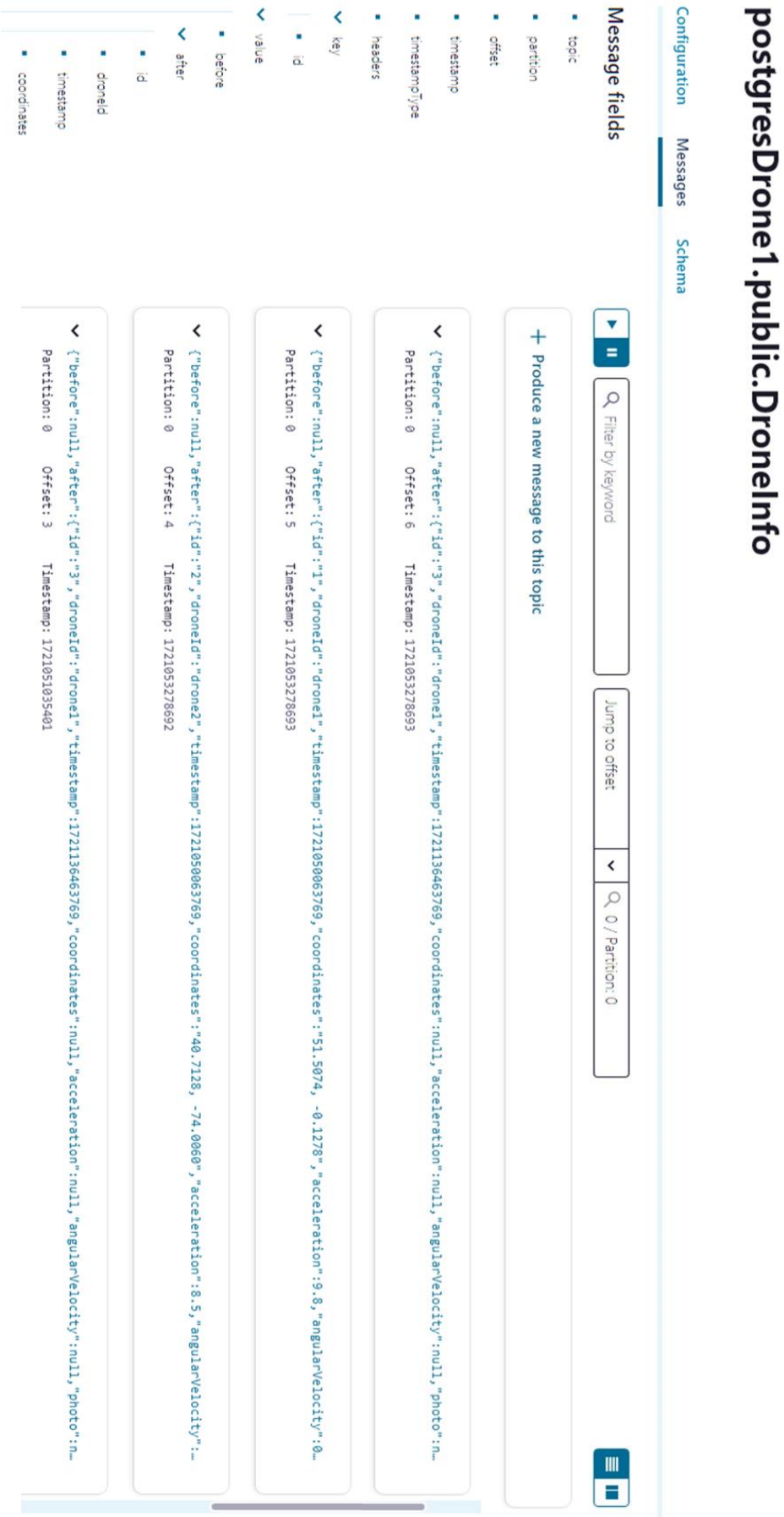


Рисунок Д.10 – Зразок повідомлень у топіку

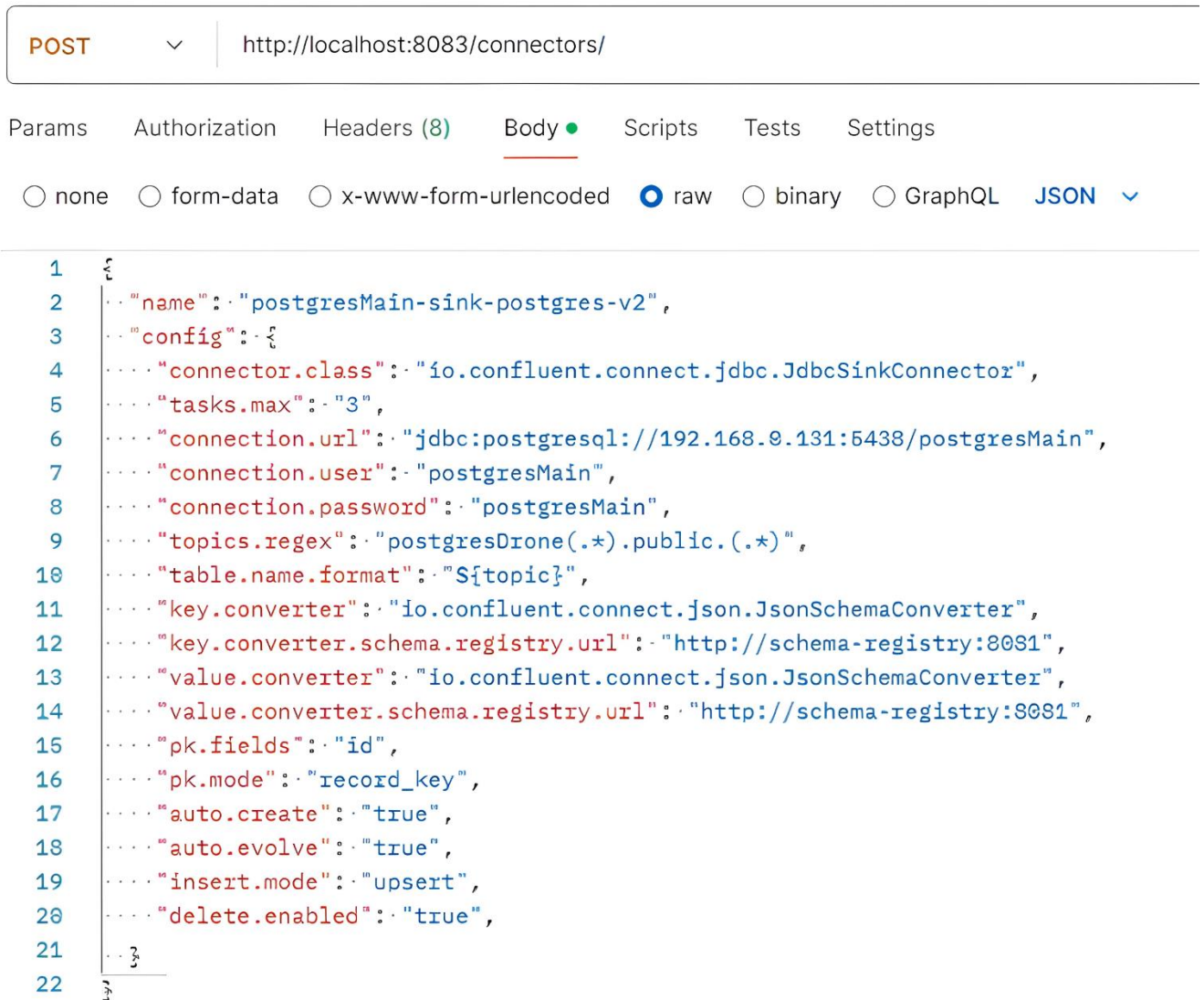


Рисунок Д.11 – Зразок створення JDBC Sink-конектора через Postman для загальної СУБД

ДОДАТОК Е. ДІАГРАМА КЛАСІВ ДЛЯ НАЛАГОДЖЕННЯ ОБМІНУ ДАНИМИ МІЖ БЕЗПЛОТНИМИ АПАРАТАМИ

DroneInfoController

```
+createRouter(get_db);
+listDroneInfo(db: Session);
+getDroneInfo(db: Session, droneId: str);
+insertDroneInfo(db: Session, drone:
DroneInfoCreate);
+updateDroneInfo(db: Session, droneId:
str, drone: DroneInfoUpdate);
+deleteDroneInfo(db: Session, droneId: str);
```

DroneInfoService

```
+listDroneInfo(db: Session);
+findDroneInfo(db: Session, droneId: str);
+insertDroneInfo(db: Session, droneData:
dict);
+updateDroneInfo(db: Session, droneId:
str, droneData: dict);
+deleteDroneInfo(db: Session, droneId: str);
```

DroneInfoRepository

```
+listDroneInfo(db: Session);
+findDroneInfo(db: Session, droneId: str);
+insertDroneInfo(db: Session, droneData:
dict);
+updateDroneInfo(db: Session, droneId:
str, droneData: dict);
+deleteDroneInfo(db: Session, droneId: str);
```

DroneInfo

```
-id: Column(String);
-droneId: Column(String);
-timestamp: Column(TIMESTAMP);
-coordinates: Column(String);
-acceleration: Column(Float);
-angularVelocity: Column(Float);
-photo: Column(String);
-video: Column(String);
-temperature: Column(Float);
-lightIntensity: Column(Float);
-batteryLevel: Column(Integer);
-transmissionSpeed: Column(Float);
-latency: Column(Float);
-packetLoss: Column(Float);
-route: Column(String);
-tasks: Column(String);
-actionLog: Column(String);
-result: Column(String);
-deviationCases: Column(String);
-emergencySituations: Column(String);
redirectedDroneId: Column(String);
```

ДОДАТОК Ж. ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ КЕРУВАННЯ БЕЗПЛОТНИМИ АПАРАТАМИ НА БАЗІ ІГРОВОГО ПІДХОДУ ТА НЕЙРОННИХ МЕРЕЖ

