

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО–КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ  
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ  
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему: "Методика автоматизованого виявлення цілі та наведення дрона з  
використанням комп'ютерного зору та методів штучного інтелекту"

на здобуття освітнього ступеня магістра  
зі спеціальності 121 Інженерія програмного забезпечення  
освітньо–професійної програми "Інженерія програмного забезпечення"

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

\_\_\_\_\_  
(підпис) Альберт ДИНДАР

Виконав: здобувач вищої освіти групи ПДМ-61  
Альберт ДИНДАР

Керівник: Наталія ТРИНТИНА  
к.т.н, доцент

Рецензент: Вікторія КОРЕЦЬКА  
к.пед.н, доцент

**Київ 2025**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ  
ІНФОРМАЦІЙНО–КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

Навчально–науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо–професійна програма "Інженерія програмного забезпечення"

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

Інженерії програмного забезпечення

\_\_\_\_\_ Ірина ЗАМРІЙ

" \_\_\_\_ " \_\_\_\_\_ 2024 р.

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Диндар Альберту Валентиновичу

1. Тема кваліфікаційної роботи: "Розробка методики автоматизованого виявлення цілі та наведення дрона з використанням комп'ютерного зору та методів штучного інтелекту"  
керівник кваліфікаційної роботи: Наталія ТРИНТИНА, к.т.н., доцент  
затверджені наказом Державного університету інформаційно–комунікаційних технологій від "15" жовтня 2024 р. № 320.
2. Строк подання кваліфікаційної роботи "17" грудня 2024 р.
3. Вхідні дані до роботи: інструменти розробки програмного забезпечення, обробки вхідних візуальних даних комп'ютерного зору, інструменти навчання та взаємодії вузлів штучного інтелекту, науково–технічна література, пов'язана з розробкою методів, науково–технічна література, пов'язана з тестуванням дронів.
4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити).
  - 4.1 Аналіз предметної галузі
  - 4.2 Опис моделі штучного інтелекту та комп'ютерного зору
  - 4.3 Створення інструменту на основі штучного інтелекту



- 4.4 Тестування розроблених методів
- 4.5 Висновки
- 5. Перелік демонстраційного матеріалу.
  - 5.1 Мета, об'єкт та предмет дослідження
  - 5.2 Порівняння аналогів
  - 5.3 Програмні засоби реалізації
  - 5.4 Діаграми
  - 5.5 Модель методу
  - 5.6 Порівняльний аналіз
  - 5.7 Результат розроблених методів
  - 5.8 Висновки
  - 5.9 Апробація результатів дослідження
- 6. Дата видачі завдання "16" жовтня 2024 р.

### КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів магістерської роботи                                                   | Строк виконання етапів роботи | Примітка |
|-------|-------------------------------------------------------------------------------------|-------------------------------|----------|
| 1     | Аналіз наявності науково–технічної літератури                                       | 16.10 – 22.10.24              |          |
| 2     | Аналіз предметної галузі                                                            | 23.10 – 01.11.24              |          |
| 3     | Аналіз методів навчання штучного інтелекту та комп'ютерного зору                    | 02.11 – 09.11.24              |          |
| 4     | Розробка методик підвищення ефективності автономної роботи дрона                    | 09.11 – 17.11.24              |          |
| 5     | Застосування розроблених методів у симуляційному середовищі та польотах на полігоні | 18.11 – 19.11.24              |          |
| 6     | Оформлення роботи: реферат, вступ, висновки                                         | 20.11 – 21.11.24              |          |
| 7     | Розробка демонстраційних матеріалів                                                 | 22.11 – 23.11.24              |          |
| 8     | Попередній захист роботи                                                            | 25.11.24                      |          |

Здобувач вищої освіти

\_\_\_\_\_

(підпис)

Альберт ДИНДАР

Керівник кваліфікаційної роботи

\_\_\_\_\_

(підпис)

Наталія ТРИНТИНА





## РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 72 стр., 5 табл., 22 рис., 19 джерел.

*Мета роботи* – вдосконалення процесу автономного наведення дронів за допомогою розробленої методики виявлення цілей на базі комп'ютерного зору та штучного інтелекту.

*Об'єкт дослідження* – процес автономного управління дроном на основі методик комп'ютерного зору та штучного інтелекту.

*Предмет дослідження* – методи виявлення цілей та наведення для дронів, що дозволяють мінімізувати участь оператора в управлінні.

Проаналізовано основні підходи до виявлення об'єктів та алгоритми навігації, що застосовуються в сучасних системах безпілотного управління. Для кожного підходу надано аналіз ефективності й можливості реалізації на базі доступних інструментів комп'ютерного зору й штучного інтелекту, після чого сформульовано рекомендації щодо вибору оптимальної методики для різних умов використання дронів.

КЛЮЧОВІ СЛОВА: КОМП'ЮТЕРНИЙ ЗІР, COMPUTER VISION, РЕБ, ШТУЧНИЙ ІНТЕЛЕКТ, ARTIFICIAL INTELLIGENCE, АКБ, ПІДКРІПЛЮВАЛЬНЕ НАВЧАННЯ, ДРОН, АВТОНОМНЕ УПРАВЛІННЯ, ВИЯВЛЕННЯ ЦІЛІ, ДОНАВЕДЕННЯ, УТРИМАННЯ ОБ'ЄКТУ, "LOCK" РЕЖИМ, FPV, ФІКСАЦІЯ ОБ'ЄКТУ, "NO-LOSS" РЕЖИМ, IDE, ХАРДВЕРНІ КОМПОНЕНТИ, GPS, FIRST PERSON VIEW, HARDWARE, UI.

## **ABSTRACT**

The text part of the qualification work for the master's degree: 72 pages, 5 tables, 22 figures, 19 sources.

The purpose of the work is to improve the process of autonomous guidance of drones using the developed methodology for detecting targets based on computer vision and artificial intelligence.

The object of research is the process of autonomous drone control based on computer vision and artificial intelligence techniques.

The subject of the study is the methods of target detection and guidance for drones that minimize the operator's participation in control.

The main approaches to object detection and navigation algorithms used in modern unmanned control systems are analyzed. For each approach, an analysis of the effectiveness and feasibility of implementation based on available computer vision and artificial intelligence tools is given, after which recommendations are formulated for choosing the optimal methodology for different conditions of using drones.

Keywords: computer vision, computer vision, EW, artificial intelligence, artificial intelligence, ACB, reinforcement learning, drone, autonomous control, target detection, homing, object holding, “lock” mode, FPV, object fixation, “no-loss” mode, IDE, hardware components, GPS, First Person View, hardware, UI.

## ЗМІСТ

|                                                                      |    |
|----------------------------------------------------------------------|----|
| ВСТУП.....                                                           | 9  |
| 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ.....                                      | 12 |
| 1.1 Поняття та огляд штучного інтелекту.....                         | 12 |
| 1.1.1 Методи навчання штучного інтелекту.....                        | 14 |
| 1.1.2 Методи вдосконалення штучного інтелекту.....                   | 19 |
| 1.2 Поняття та огляд комп'ютерного зору.....                         | 21 |
| 1.2.1 Методи створення комп'ютерного зору.....                       | 24 |
| 1.2.2 Методи навчання комп'ютерного зору.....                        | 26 |
| 1.2.3 Методи тестування комп'ютерного зору.....                      | 28 |
| 2 ОПИС МОДЕЛІ.....                                                   | 30 |
| 2.1 Загальна архітектура штучного інтелекту.....                     | 30 |
| 2.2 Загальна архітектура комп'ютерного зору.....                     | 35 |
| 3 СТВОРЕННЯ ІНСТРУМЕНТУ НА ОСНОВІ.....                               | 39 |
| ШТУЧНОГО ІНТЕЛЕКТУ ТА КОМП'ЮТЕРНОГО ЗОРУ.....                        | 39 |
| 3.1 Засоби розробки.....                                             | 39 |
| 3.1.1 Загальні вимоги до засобів розробки та мови програмування..... | 39 |
| 3.1.2 Середовища розробки Visual Studio Code та PyCharm.....         | 41 |
| 3.1.3 Інструмент для симуляції AirSim.....                           | 43 |
| 3.2 Бібліотеки та інструменти.....                                   | 45 |
| 3.2.1 Бібліотеки обробки зображень.....                              | 46 |
| 3.2.2 Бібліотеки для роботи з даними.....                            | 47 |
| 3.2.3 Інструменти для оптимізації.....                               | 48 |
| 3.3 Розробка методів.....                                            | 49 |
| 3.3.1 Архітектура створеного інструменту.....                        | 49 |
| 3.3.2 Інтеграція та налаштування бібліотек у проект.....             | 50 |
| 3.3.3 Реалізація алгоритмів комп'ютерного зору.....                  | 51 |
| 4 ПРОЦЕС ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.....                       | 59 |
| 4.1 Проведення тестування.....                                       | 59 |
| 4.2 Аналіз результатів.....                                          | 63 |
| ВИСНОВКИ.....                                                        | 65 |
| ПЕРЕЛІК ПОСИЛАНЬ.....                                                | 66 |
| ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ.....                             | 68 |

## ВСТУП

З розвитком сучасних технологій безпілотні літальні апарати (БПЛА), або простіше кажучи дрони, набувають все більшого поширення в різних галузях, таких як військові операції, сільське господарство, логістика, рятувальні місії та багато інших. Однак, ефективне використання дронів часто вимагає значної участі оператора, що може бути не тільки складним, але й ризикованим у деяких ситуаціях. В умовах, де важлива швидкість реакції, точність та мінімізація людського втручання, виникає потреба в автономних системах управління.

**Актуальність дослідження.** Автономне управління дронами є однією з ключових вимог до сучасних безпілотних систем, що використовуються у складних та небезпечних умовах, де людське втручання варто звести до мінімуму. Впровадження комп'ютерного зору та штучного інтелекту дозволяє дронам самостійно виявляти та розпізнавати об'єкти, визначати пріоритетність цілей та приймати рішення у реальному часі. З огляду на важливість точності та швидкості роботи таких систем, автономне управління БПЛА набуває актуальності у військових, рятувальних та промислових завданнях. Сучасні алгоритми комп'ютерного зору та машинного навчання, зокрема, дозволяють виконувати виявлення об'єктів на основі візуальних даних, тоді як штучний інтелект надає можливість оптимізувати траєкторію та ефективно використовувати обмежені ресурси дрона. Такий підхід дозволяє знизити людські помилки, підвищуючи автономність, ефективність та безпеку. Водночас швидка адаптація до мінливих умов середовища залишається викликом для розробників, адже необхідно поєднати передові технології штучного інтелекту з обчислювальною ефективністю для досягнення надійності в екстремальних ситуаціях.

**Мета** – вдосконалення процесу автономного наведення дронів за допомогою розробленої методики виявлення цілей на базі комп'ютерного зору та штучного інтелекту.

**Об'єкт дослідження** – процес автономного управління дроном на основі методик комп'ютерного зору та штучного інтелекту.

**Предмет дослідження** – методи виявлення цілей та наведення для дронів, що дозволяють мінімізувати участь оператора в управлінні.

Проаналізовано основні підходи до виявлення об'єктів та алгоритми навігації, що застосовуються в сучасних системах безпілотного управління. Для кожного підходу надано аналіз ефективності й можливості реалізації на базі доступних інструментів комп'ютерного зору й штучного інтелекту, після чого сформульовано рекомендації щодо вибору оптимальної методики для різних умов використання дронів.

**Завдання** дослідження полягають в наступному:

1. Провести аналіз методів та інструментів, що використовуються для розробки автономних систем управління дронами.
2. Розглянути та адаптувати алгоритми комп'ютерного зору для виявлення об'єктів у реальному часі.
3. Дослідити методи навчання нейронних мереж для оптимального використання ресурсів у дронах.
4. Розробити та протестувати алгоритми автономного наведення дрона на ціль.
5. Провести оцінку ефективності та тестування розробленої методики в умовах, наближених до реальних.



## 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

### 1.1 Поняття та огляд штучного інтелекту

Штучний інтелект – це галузь інформатики, спрямована на створення програмного забезпечення або систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту, тобто втручання людини. Такі системи згодом здатні самонавчатися, обробляти та аналізувати великі обсяги інформації, яку їй надає людина (розробник), а також приймати рішення та адаптуватися до умов, що змінюються. Основною метою ШІ є моделювання та відтворення людської здатності до навчання, вирішення проблем та задач, розпізнавання об'єктів та мовлення, а також вміння ухвалювати рішення на основі отриманих та опрацьованих даних.

Штучний інтелект можна поділити на декілька категорій залежно від його функціональності, здатності до навчання та призначення. Слабкий (або "вузький") AI здатен виконувати лише конкретно поставлені задачі (наприклад, розпізнавання силуетів), тоді як сильний штучний інтелект спрямований на створення універсальних систем, здатних розв'язувати будь-які інтелектуальні завдання у визначеній сфері подібно до людини. Однак, на сучасному етапі розвитку більшість існуючих реалізацій ШІ базуються на вузькому ШІ, який обмежений виконанням лише спеціалізованих задач. Це сучасні потреби людей до штучного інтелекту.

У технологіях штучного інтелекту широко застосовуються алгоритми машинного навчання, які дозволяють системам "вчитися" на основі даних початкових даних, або даних, які надаються у процесі розвитку AI. Це досягається шляхом налаштування математичних моделей, що коригуються та оптимізуються

для поліпшення точності виконання задач. Сучасні методи машинного навчання поділяються на декілька основних типів, які ми розглянемо пізніше.

Крім машинного навчання в штучному інтелекті також використовуються методи глибинного навчання, які дозволяють моделювати більш складні зв'язки за допомогою багаторівневих нейронних мереж. Це надає можливість для обробки великих обсягів даних та інформації, також розпізнавання складних патернів (силуетів, малюнків), що є особливо важливим у сферах, де потрібна висока точність, наприклад у таких як медична діагностика, чи комп'ютерному зорі. Точність виконання, високий відсоток успіху та низький шанс похибки у подібних сферах є надважливим та визначаючим фактором для подальшого використання конкретного AI.

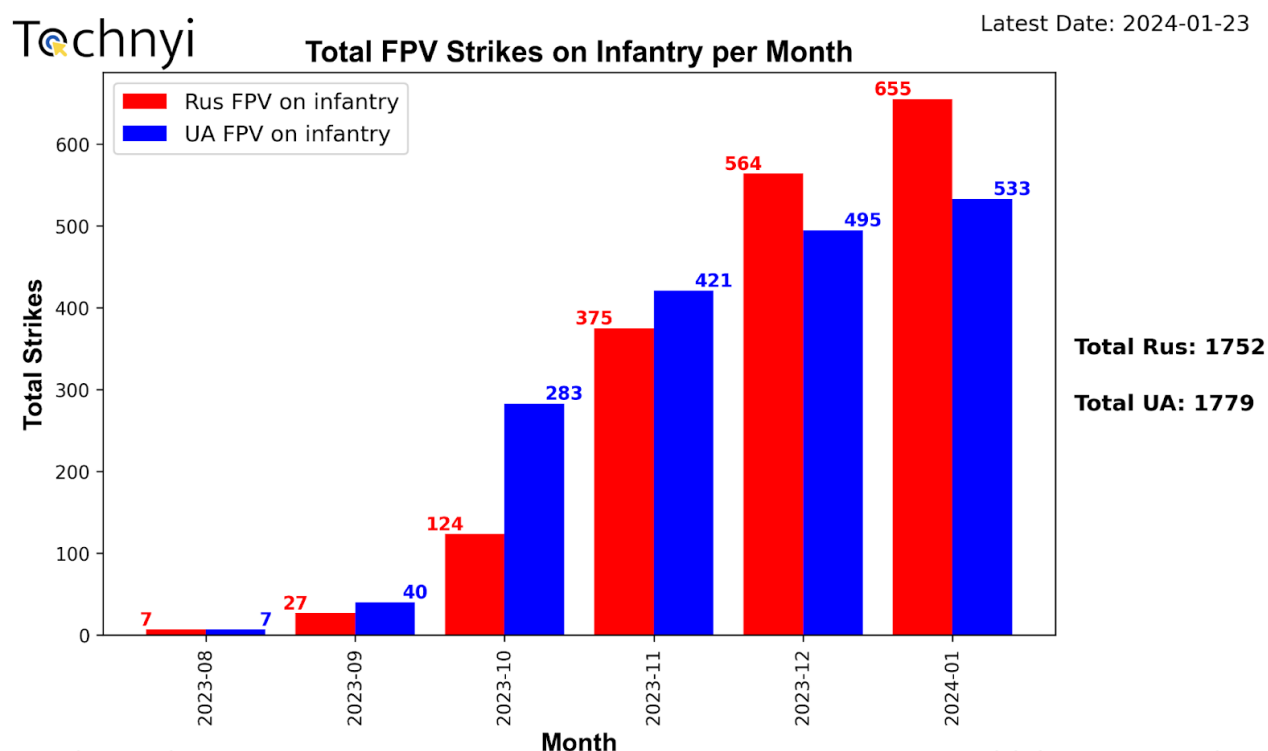


Рис. 1.1 Графік зростання використання дронів із ШІ

Це результати відстеження випадків використання дронів з штучним інтелектом у 2023–2024 роках. Можемо бачити що за 5 місяців використання

таких дронів зросло у майже 10 разів. Інтерес до ШІ постійно зростає, що обумовлено ефективністю алгоритмів у вирішенні сьогоденних завдань, особливо там, де автоматизація та збереження життя людини мають важливе значення. У цій роботі акцент зроблено на застосуванні AI для автономного, або напів автономного управління дронами. Завдяки цьому дрони здатні здійснювати виявлення об'єктів, їх захоплення, слідкувати за положенням та переміщенням цілі, а також адаптувати свої дії у реальному часі в залежності від умов. Це відкриває нові перспективи у військовій, рятувальній та промисловій галузях.

### **1.1.1 Методи навчання штучного інтелекту**

Способи та підходи до навчання штучного інтелекту поділяються на кілька основних підходів, кожен із яких має власну специфіку та сферу застосування. До найбільш поширених методів належать контрольоване, неконтрольоване та підкріплювальне навчання. Кожен з цих методів розроблений для певних типів завдань, що визначають специфіку навчання та структуру алгоритму. Розглянемо детальніше кожен з цих методів.

**Контрольоване навчання** (supervised learning) є найпоширенішим методом навчання, який використовує підписані дані, тобто кожен зразок у наборі даних має відповідну мітку чи вихідне значення. Основна ідея полягає в тому, що модель навчається, порівнюючи свій вихід із правильними відповідями, а потім коригує свої параметри, щоб мінімізувати похибку у майбутньому. Контрольоване навчання широко застосовується у задачах класифікації та регресії. Однією з класичних моделей контрольованого навчання є логістична регресія, яка використовується з 1960-х років для бінарної класифікації, наприклад, у фінансовій та медичній сферах. Розглянемо її далі для кращого розуміння.

**Логістична регресія** — це базовий алгоритм контрольованого навчання, що використовується для класифікації об'єктів між двома класами. Алгоритм будує

функцію, яка приймає значення від 0 до 1 і на основі цього значення передбачає, до якого класу належить об'єкт.

Розглянемо дію алгоритму:

1. Для кожного об'єкта в наборі даних алгоритм обчислює значення зважених параметрів (тобто коефіцієнтів) для його характеристик, що дозволяє визначити наскільки кожна характеристика впливає на результат.
2. Потім використовується функція сигмоїди, яка переводить отримане значення у ймовірність від 0 до 1.
3. На основі порогового значення (нехай це буде 0.5), алгоритм визначає клас об'єкта: значення вище 0.5 вважається належністю до одного класу, нижче 0.5 — відповідно до іншого.
4. Алгоритм оновлює свої коефіцієнти, мінімізуючи помилки для всіх об'єктів у наборі даних, поступово наближаючись до оптимальних значень.

Логістична регресія проста у використанні й добре підходить для задач класифікації, де об'єкти мають бінарні класи (наприклад, "так" чи "ні").

Наведемо ще декілька важливих прикладів. Перший це Support Vector Machine (SVM), розроблений у 1995 році Володимиром Вапником, який показав високу ефективність у розв'язанні задач класифікації з невеликою кількістю параметрів. Другий це модель AlexNet розроблена у 2012 році Алексом Крижевським, яка продемонструвала новий підхід до класифікації зображень, використовуючи згорткові нейронні мережі (CNN) і здобула перемогу на змаганнях ImageNet, що стало важливою подією в історії контрольованого навчання.

**Неконтрольоване навчання** (unsupervised learning) є методом, що використовується для роботи з даними без міток. Тобто система намагається самостійно знаходити структури, закономірності або кластери в даних. Цей метод корисний у завданнях кластеризації та зменшення розмірності, де основною метою є знайти структуру у даних або згрупувати їх на основі спільних ознак. Одним із популярних алгоритмів неконтрольованого навчання є k-means кластеризація, що була представлена ще у 1950-х роках Стюартом Ллойдом. Застосування такого методу може бути корисним для виявлення аномалій, або пошуку схожих об'єктів у великих масивах зображень, що важливо наприклад у сфері відеоспостереження та охорони.

**K-means кластеризація** – це один із найпростіших алгоритмів неконтрольованого навчання, який дозволяє розбити дані на групи або "кластери" на основі їхньої схожості. Цей алгоритм не вимагає підписаних даних і самостійно знаходить структури в даних.

Дія алгоритму:

1. Спочатку визначається кількість кластерів ( $k$ ), на які потрібно розділити дані.
2. Алгоритм випадково вибирає  $k$  точок, які будуть початковими центрами кластерів.
3. Далі для кожного об'єкта обчислюється відстань до кожного з центрів кластерів, і об'єкт призначається до найближчого кластеру.
4. Центри кластерів оновлюються як середні значення об'єктів у кожному кластері.
5. Ці кроки повторюються до тих пір, поки центри кластерів більше не змінюються, або не досягнуто максимальну кількість ітерацій.

K-means кластеризація добре підходить для аналізу груп даних, таких як сегментація клієнтів, або розпізнавання патернів у великих наборах даних. Іншим

прикладом послугує алгоритм головних компонент (РСА), запропонований у 1901 році Карлом Пірсоном і широко використовуваний для зниження розмірності даних у таких сферах як аналіз зображень та фінанси. У 2014 році була створена модель Word2Vec розробником Томасом Міковим, яка також базується на неконтрольованому навчанні. Алгоритм Word2Vec дозволяє знаходити семантичні зв'язки між словами, і це стало серйозним досягненням у обробці.

**Підкріплювальне навчання** орієнтоване на взаємодію агента із середовищем. У цьому методі модель навчається на основі відгуків, що отримуються після виконання дій, які призводять до певних змін у середовищі. За успішні дії система отримує позитивну винагороду, а за помилки – негативну. Підкріплювальне навчання широко використовується для автономного управління, і для навчання дронів уникати перешкод та оптимізувати маршрути. Підкріплювальне навчання застосовується у сфері робототехніки (дрони), автономного водіння (автопілоти автомобілів), ігор та автоматизованого управління (медичинські операції). Цей тип навчання ШІ найкраще підходить для досягнення мети цієї магістерської роботи.

Одним із перших алгоритмів у цій галузі є Q-навчання, розроблене Крісом Уоткінсом у 1989 році. Цей алгоритм став основою для багатьох сучасних алгоритмів підкріплювального навчання. Таким чином у 2013 році компанія DeepMind, що є дочірньою компанією Google, представила Deep Q-Network (DQN), яка використала глибокі нейронні мережі для гри у класичні відеоігри Atari, де агент зміг досягти високих результатів.

**Q-навчання** – один із найпростіших алгоритмів підкріплювального навчання, що дозволяє агенту (алгоритму), навчитися приймати оптимальні рішення в середовищі, щоб досягнути максимальної винагороди. Цей алгоритм базується на концепції таблиці Q-значень, де кожне значення в таблиці визначає, наскільки вигідно виконати певну дію на конкретному етапі.

Опис дії алгоритму:

1. Для кожного можливого стану і дії в цьому стані створюється таблиця Q-значень, яка заповнюється випадковими значеннями.
2. Агент (алгоритм) починає діяти в середовищі, отримуючи при цьому нагороди або покарання за свої дії. Далі алгоритм оновлює значення в таблиці Q-значень за допомогою формули, що враховує "поточне Q-значення", отриману "винагороду" та очікуване "максимальне значення" у наступному стані.
3. Далі алгоритм обдумує та обирає дії, які принесуть максимальну винагороду у наступній ітерації. З часом зменшується ймовірність випадкового вибору дій для подальшого розвитку.
4. Після багатьох ітерацій таблиця Q-значень містить оптимальну стратегію, тобто шлях, для того щоб досягнути максимальної винагороди у поточному середовищі.

Q-навчання використовується для задач, де агенту (алгоритму) потрібно приймати рішення в умовах невизначеності, або браку інформації, наприклад в іграх чи автономному управлінні автомобілем. Він дозволяє системі самостійно визначати оптимальну поведінку через накопичений досвід. У 2017 році DeepMind також представила алгоритм AlphaGo, який вперше переміг професійного гравця в гру "Go", що стало цікавою та важливою подією в підкріплювальному навчанні і привернуло увагу до його потенціалу у розв'язанні складних завдань.

Кожен із методів навчання має свої переваги та обмеження, які визначають його ефективність для конкретного завдання. Контрольоване навчання забезпечує точність у прогнозуванні, неконтрольоване навчання дозволяє досліджувати великі обсяги даних без попереднього внесення даних, а підкріплювальне навчання ідеально підходить для динамічних систем, що самонавчаються і потребують розвитку.

### 1.1.2 Методи вдосконалення штучного інтелекту

Алгоритми покращення штучного інтелекту дозволяють підвищити ефективність і точність алгоритмів, адаптуючи їх до складніших умов та нових задач. Одним із таких методів є оптимізація гіперпараметрів. Налаштування параметрів, таких як швидкість навчання або кількість шарів у нейронній мережі, може суттєво вплинути на якість і продуктивність моделі. Оптимізація допомагає досягти більш стабільних та очікуваних результатів. Також методи вдосконалення дозволяють зменшити обчислювальні витрати, енерговитрати, час на обчислення, покращити стійкість моделей до змінних умов, досягти кращої продуктивності на складних і великих наборах даних.

Оптимізація моделей є процесом налаштування параметрів алгоритму для досягнення максимальної продуктивності. Один із найпоширеніших методів оптимізації – градієнтний спуск (Gradient Descent), який мінімізує функцію втрат шляхом поступового оновлення параметрів у напрямку зменшення помилок. Цей алгоритм багато де має прихильність у світі програмування. Існують різні модифікації градієнтного спуску, в залежності від задачі. Наприклад, адаптивний градієнтний алгоритм (AdaGrad – Adaptive Gradient) та адаптивна моментна оптимізація (Adam – Adaptive Moment), які автоматично налаштовують швидкість навчання для кожного параметра забезпечуючи стабільнішу оптимізацію. Такі підходи дозволяють ефективніше налаштовувати складні нейронні мережі, покращуючи їх продуктивність навіть з великими наборами даних.

Ансамблеві методи поєднують кілька моделей для покращення якості прогнозів. Одним із популярних методів ансамблювання є беггінг (Bagging), запропонований Левом Брейманом у 1996 році, де кілька слабких моделей об'єднуються для зниження дисперсії та підвищення точності. Bagging включає такі відомі алгоритми як Random Forest, що об'єднує дерева рішень для покращення стабільності та узагальнення спільної моделі. Інший ансамблевий метод бустинг (Boosting) вдосконалює кожну наступну модель на основі помилок



попередніх. Алгоритм XGBoost, розроблений у 2016 році є однією з найпопулярніших реалізацій бустингу і часто застосовується у змаганнях з аналізу даних завдяки своїй точності.

Також варто згадати про трансферне навчання, яке дозволяє використовувати знання, здобуті моделлю на одному завданні, для вирішення інших пов'язаних завдань. Це особливо корисно, коли новий набір даних є малим, або ресурси для навчання недостатні, обмежені. За допомогою трансферного навчання можна адаптувати великі попередньо натреновані моделі, наприклад BERT, для обробки природної мови, або VGGNet для розпізнавання зображень чи силуетів. Краще всього використовувати такі методи навчання для специфічних задач, що забезпечить більшу точність при значно менших обчислювальних витратах, в тому числі енергетичних.

**Регуляризація** – це метод, що дозволяє знизити ризик перенавчання моделі, тобто її адаптації до шуму (сміття) в навчальних даних. Найпоширеніші методи регуляризації – L1 і L2 регуляризація, які штрафують великі значення параметрів, сприяючи їх зменшенню та підвищуючи узагальненість моделі. Інший підхід до регуляризації – Dropout, який запропонований у 2014 році Джеффрі Гінтоном. Він випадковим чином відключає деякі нейрони під час навчання. Dropout широко використовується у нейронних мережах для зниження ризику перенавчання та підвищення стійкості моделі до змінних вхідних даних.

Метод аугментації даних є ще одним ефективним методом вдосконалення ШІ, особливо у випадках, коли кількість навчальних даних обмежена. Вона включає генерування нових даних на основі наявних даних шляхом їх обертання, масштабування, зміни кольорів тощо. Цей спосіб значно підвищує стійкість і точність моделей ШІ, особливо у задачах комп'ютерного зору, бо обсяги даних є ключовими для продуктивності. Вона широко застосовується у таких моделях як

ResNet та Inception, які досягли гарних результатів на змаганнях з класифікації зображень.

Завдяки цим методам вдосконалення моделі штучного інтелекту стають ефективнішими, стійкішими та точнішими. Кожен із зазначених методів вдосконалення дозволяє розширити можливості ШІ. Завдяки вдосконаленню алгоритмів можливо підвищити ефективність дронів у задачах, що вимагають високої точності та стабільності.

## **1.2 Поняття та огляд комп'ютерного зору**

Комп'ютерний зір (Computer Vision, CV) – це галузь штучного інтелекту, що займається автоматичним аналізом та інтерпретацією зображень та відео. Завдяки алгоритмам комп'ютерного зору, машини можуть "бачити" і розпізнавати об'єкти, що дозволяє використовувати ці системи у різних сферах, наприклад безпілотні технології, медицина, робототехніка та промисловість.

У дронах комп'ютерний зір надає можливість розпізнавати навколишні об'єкти, оцінювати їхні розміри за потреби, відстань та потенційну траєкторію руху. Це особливо важливо для автономного управління, де CV дозволяє дрону уникати перешкод, слідувати за об'єктами і виконувати маневри без втручання оператора. Основними задачами комп'ютерного зору є виявлення об'єктів, класифікація об'єкту, сегментація зображення та відстеження.

Загалом, основна концепція комп'ютерного зору полягає у перетворенні візуальної інформації з різних сенсорів (камер, тепловізорів) у зрозумілі для машини формати. Це включає обробку зображень і подальше розпізнавання певних об'єктів або особливостей на них. Отже, при інтеграції CV з алгоритмами ШІ, дрони та інші автономні системи отримують здатність самостійно орієнтуватися у просторі та приймати рішення, що робить їх використання ефективним у реальних умовах.

Комп'ютерний зір включає кілька ключових підзадач, серед яких:

- **Виявлення об'єктів** (*object detection*): процес ідентифікації конкретних об'єктів на зображеннях, наприклад, автомобілів, людей, дорожніх знаків тощо. Виявлення об'єктів є корисним для автономного управління машиною, де система повинна своєчасно розпізнавати перешкоди.
- **Класифікація зображень** (*image classification*): процес, у якому алгоритм визначає, до якого класу належить зображення або його фрагмент. Моделі, такі як AlexNet та ResNet, показали чудові результати у класифікації, заклавши основу для більш точного розпізнавання візуальних об'єктів.
- **Сегментація зображень** (*image segmentation*): процес розбиття зображення на окремі області або об'єкти. Сегментація дозволяє виділити конкретні частини зображення, наприклад, розділити фон від основного об'єкта, що є критично важливим для медичної діагностики та аналізу зображень із супутників.
- **Відстеження об'єктів** (*object tracking*): процес відстеження руху об'єктів у відеопотоці. Цей метод широко використовується в системах відеоспостереження, де потрібно слідкувати за певним об'єктом протягом тривалого часу.
- **Фіксація об'єкту** (*object no-loss*): процес відстеження форми об'єкта у режимі реального часу. Цей метод вузькоспеціалізований, зокрема у робототехніці та військовій справі. Його задача утримувати заданий об'єкт, вести його у разі пересування, та головною задачею є "розуміння" методу змінення форми чи вигляду відстежуваного об'єкту.

Застосування алгоритмів глибинного навчання у computer vision в наші часи значно покращило здатність розпізнавати та передавати візуальні дані комп'ютеру у зрозумілому йому вигляді.

**CNN (Convolutional Neural Networks)** алгоритми забезпечують високу точність розпізнавання завдяки автоматичному виділенню головних ознак через згорткові шари, що є основою для обробки зображень. Хоча вони і досягають гарної точності, але все ж потребують значних обчислювальних ресурсів, що робить їх менш придатними для задач, де критичною є швидкість та енергоефективність.

**Архітектура YOLO (You Only Look Ones)** запропонована Джозефом Редмоном у 2016 році, дозволяє одночасно виявляти та класифікувати об'єкти у режимі реального часу. Завдяки методу "одноразового перегляду" (one look), YOLO може обробляти зображення і визначати координати об'єктів, що робить його ідеальним для застосувань, де важлива висока швидкість, наприклад, для автономних транспортних засобів.

**SSD (Single Shot MultiBox Detector)** поєднує швидкість і точність, але має менші вимоги до обчислювальних ресурсів порівняно з CNN. Використовуючи багаторівневі особливості, SSD дозволяє ефективно розпізнавати об'єкти різних розмірів.

Комп'ютерний зір активно використовує методи оптимізації, такі як аугментація даних, для підвищення точності та стійкості моделей. Це дозволяє системам адаптуватися до змінних умов об'єктів і працювати на результат, де візуальна інформація є основним джерелом даних. Нижче представлено порівняння основних алгоритмів CV за їхніми характеристиками:

Таблиця 1.1

Основні характеристики досліджуваних алгоритмів

| Алгоритм | Точність (%) | Швидкість (FPS) | Обчислювальні ресурси |
|----------|--------------|-----------------|-----------------------|
| CNN      | 95           | 10              | Високі                |
| YOLO     | 90           | 45              | Середні               |
| SSD      | 85           | 30              | Низькі                |

### 1.2.1 Методи створення комп'ютерного зору

Взаємодія із CV системами потребує алгоритмів здатних обробляти, аналізувати та інтерпретувати візуальні дані, перетворюючи їх на зрозумілу для машини інформацію. Основні методи створення комп'ютерного зору включають обробку зображень, використання згорткових нейронних мереж та алгоритмів розпізнавання силуетів чи об'єктів, що надають машині здатність автоматично визначати й класифікувати об'єкти на зображеннях чи відео. Створення CV відбувалося поступово, починаючи з фундаментальних досліджень у галузі штучного інтелекту та обробки зображень. Ідея полягала у розробці систем, здатних сприймати візуальні дані та перетворювати їх у зрозумілу інформацію для ПК, як це робить людський зір. Розробка CV пройшла декілька ключових етапів, кожен із яких вплинув на сучасні підходи та технології.

**Обробка зображень** – фундаментальний етап, це база. Воно включає попередню обробку візуальних даних для покращення якості та підготовки до подальшого аналізу. Обробка зображень може включати такі операції як фільтрація шуму, зміна контрастності, нормалізація зображення, та його вирівнювання. Ці операції дозволяють зменшити зайві елементи у вхідних даних для алгоритмів CV і підвищити їхню придатність для аналізу. Наприклад, фільтри Гауса або Соба використовуються для виявлення контурів об'єктів, і це є основою для їх подальшого розпізнавання.

Перші дослідження комп'ютерного зору розпочалися у 1960–х роках, коли вчені почали експериментувати з автоматичною обробкою зображень. Початковою метою було зрозуміти, як комп'ютери можуть «бачити» та інтерпретувати базові об'єкти, такі як геометричні форми. На цьому етапі дослідники використовували прості методи обробки зображень, зокрема виділення контурів і сегментацію, які допомагали розбивати зображення на окремі частини. У цей період відбувалися також експерименти з розпізнавання символів та оптичного розпізнавання тексту (OCR).

Далі через 20 років дослідники почали застосовувати машинне навчання для аналізу візуальних даних. Цей підхід дозволив навчати комп'ютери розпізнавати об'єкти на основі великих тренувальних наборів даних, а не за жорстко закладеними правилами. Алгоритми, як-от метод опорних векторів (SVM) і нейронні мережі, почали знаходити застосування у CV. Цей період ознаменувався переходом від суто програмних методів до інтеграції з апаратними рішеннями, наприклад CCD-камери та спеціальні сенсори, які могли фіксувати зображення з високою роздільною здатністю (dpi). Ці камери дозволяли покращити якість початкових даних, що слугували основою для навчання систем.

**Згорткові нейронні мережі** це базовий метод комп'ютерного зору, який підвищує точність розпізнавання об'єктів. CNN складаються з шарів що обробляють вхідне зображення, прямо на рівні пікселів, витягуючи з нього основні ознаки. Згорткові шари об'єднують локальні характеристики такі як текстури, краї, тоді як шари субдискретизації знижують розмірність даних, що дозволяє заощадити обчислювальні ресурси. CNN вперше були використані в задачах розпізнавання рукописних символів у 1980-х роках, але їх популярність зросла тільки після успіху моделі AlexNet у 2012 році, яка продемонструвала високу точність у класифікації зображень.

Звісно що з розвитком комп'ютерного зору виникла потреба у покращенні обладнання для збору даних. Тому розробники почали застосовувати стереокамери, тепловізори та лідари, які дозволяли системам отримувати більше даних про навколишнє середовище. Лідар, наприклад, використовується для створення тривимірних моделей об'єктів. Вони стали настільки корисними, що тепер лідари присутні навіть у смартфонах. Використання датчиків разом із CV є основою для автономних транспортних засобів, наприклад електромобілів, де потрібне точне та швидке сприйняття простору навколо, і відповідних дій у разі безпеки.

Алгоритми розпізнавання об'єктів, які ми розглянули раніше, є ключовими для виконання задач комп'ютерного зору. YOLO працює за принципом "одноразового погляду", що забезпечує високу швидкість обробки, необхідну для задач у реальному часі. SSD є високопродуктивним алгоритмом, який працює за принципом "одиначного проходу", проте його архітектура орієнтована на обробку об'єктів різних розмірів і масштабів, і це робить його універсальним для застосувань у різних середовищах. Лідари сприяють створенню тривимірних моделей і кращому визначенні відстаней.

Використання глибинних моделей з багатоетапною обробкою є ще одним методом для CV. Наприклад, архітектура R-CNN (Region-based Convolutional Neural Networks) спочатку виділяє області на зображенні, що потенційно містять об'єкти, і лише потім аналізує їх за допомогою згорткових мереж. Модель R-CNN була запропонована у 2014 році і стала базою для подальшого розвитку більш точних алгоритмів. Це Fast R-CNN та Faster R-CNN, які використовуються для виявлення об'єктів зі складними ознаками та патернами.

Методи створення Computer Vision варіюються від простих, наприклад операції обробки зображень, до складних, таких як глибинні нейронні мережі, які здатні адаптуватися до різних задач. Кожен метод має свою специфіку і застосовується в залежності від вимог до точності та доступних ресурсів. Це робить комп'ютерний зір гнучким інструментом для автоматизації та аналізу візуальної інформації.

### **1.2.2 Методи навчання комп'ютерного зору**

Покращення систем CV включає спеціалізовані методи, що дозволяють обробляти й аналізувати зображення або відео з максимальною точністю і швидкістю. На відміну від інших галузей ІІІ, комп'ютерний зір вимагає технік, які

враховують просторові взаємозв'язки між пікселями, відстані між об'єктами, масштабування об'єктів і змінні умови освітлення.

Сегментація зображень дозволяє моделі розділяти зображення на окремі частини або об'єкти, виділяючи межі та області. Наприклад, семантична сегментація дозволяє кожному пікселю зображення присвоїти певну категорію (наприклад, дорога, пішохід, автомобіль), і це є важливим для автономного водіння транспортним засобом, де машина повинна чітко відрізнити дорожні знаки від перешкод.

Методи класифікації зображень у комп'ютерному зорі часто включають спеціальні типи CNN нейронних мереж, які фокусуються на витягуванні контурів, текстур та кольорів. Однією з ключових моделей у цій сфері є ResNet, представлена у 2015 році. Вона використовує глибоку архітектуру з "залишковими зв'язками" для ефективного навчання навіть дуже складних мереж. Цей підхід дозволяє нам уникати проблеми згасаючого градієнта.

Алгоритми розпізнавання об'єктів (YOLO, SSD, CNN тощо) забезпечують можливість одночасного розпізнавання та локалізації об'єктів у режимі реального часу. Це дозволяє використовувати CV у завданнях де швидкість є важливою, в тих же військових цілях, в системах безпеки, моніторингу тощо.

Трансферне навчання дозволяє використовувати попередньо навчені моделі для нових задач з меншими обсягами даних. Наприклад, модель VGGNet, попередньо навчена на ImageNet для класифікації зображень, була навчена на наборі даних для класифікації зображень, тож її можна адаптувати для специфічного завдання, такого як виявлення конкретних об'єктів на зображеннях з дронів. Це значно скорочує час навчання, оскільки модель вже має початкові знання про загальні об'єкти.



Система навчання з аугментацією даних забезпечує додаткову гнучкість комп'ютерного зору. Використовуючи техніки обертання, зміни масштабу, відзеркалювання та інші трансформації, можна суттєво розширити навчальний набір. В цьому є користь, коли система повинна функціонувати в умовах змінного освітлення, різних кутів огляду або наявності перешкод, що підвищує стійкість моделі до непередбачених умов.

### **1.2.3 Методи тестування комп'ютерного зору**

Проведення тестових робіт комп'ютерного зору, особливо для застосування в автономних дронах, є головним етапом для забезпечення усіх очікуваних потреб, таких як точність, стабільність та надійність системи в реальних умовах. Сучасні методи тестування включають використання симуляційних середовищ, полігонні випробування та тренування на основі отриманих даних. Перевірка у реальних польотах потрібна для оцінки здатності дрону виявляти цілі, тримати їх у "lock" режимі, навігувати у складних умовах та адаптуватися до змін середовища.

Процес тестування починається з симуляційних середовищ, де алгоритми комп'ютерного зору попередньо тестується, використовуючи метод інтеграції з хардверними компонентами та системами дрона. Симуляції дозволяють оцінити ефективність алгоритмів виявлення та наведення в умовах, наближених до реальних, при цьому мінімізуючи ризики для обладнання. Отримані попередні результати можна використовувати для аналізу та вдосконалення для повторного тестування. Також завдяки симуляційним середовищам можна протестувати функціональність системи у випадках обмеження GPS, наявністю перешкод, втратою з станцією керування, або змінними умовами середовища польоту. На цьому етапі інтерфейси ПО забезпечують обмін даними між компонентами системи в реальному часі, що є користо та важливо для плавної інтеграції всього необхідного функціоналу у кінцевий продукт.

Наступним етапом є тестування на реальних дронах у польових умовах. Це дозволяє оцінити працездатність і надійність розробленої системи у реальних умовах, а також виявити слабкі місця, які не завжди проявляються у симуляціях, зокрема в умовах зміни погоди, ландшафту, чи при появі раптових перешкод. Таким чином система, створена на основі алгоритмів підкріпленого навчання, демонструє високу адаптивність до непередбачуваних змін – нейронна мережа може самостійно адаптуватись під змінні умови, і завдяки накопиченому досвіду вдосконалюватися для наступних випробувань.

Серед основних аспектів тестування комп'ютерного зору на дронах можемо виділити оцінку здатності системи до автономних прийнятих рішень щодо обходу перешкод і коректного розпізнавання цілей. Умови з обмеженням, або повністю відсутнім GPS сигналом, та іншими сигналами, створюють особливий виклик для систем навігації, на які також покладається комп'ютерний зір. Подібні складні ситуації дозволяють протестувати ефективність розроблених алгоритмів у складних реальних умовах і оцінити адаптаційні можливості ШІ.

## 2 ОПИС МОДЕЛІ

### 2.1 Загальна архітерктура штучного інтелекту

Архітектура ШІ для автономного управління БПЛА, БпАК, дроні тощо, базується на багаторівневій системі алгоритмів, який включає в себе збір даних, їх обробку, аналіз що відбувається та прийняття рішень. ШІ у цьому контексті інтегрує компоненти, що відповідають за виявлення цілей, оцінку ситуації та автоматичне коригування траєкторії польоту, чи енергоспоживання, відповідно до поточних умов і задач. В основі архітектури лежить нейронна мережа, яка була згадана раніше, і яка вчиться на великих наборах даних та приймає рішення на їх основі, і на основі свого попереднього досвіду.

Структура архітектури ШІ для дрона включає такі основні блоки:

- **Сенсорний блок** – збирає інформацію із зовнішнього середовища, наприклад з камер, GPS модулів, гіроскопів, горизонту, тепловізорів або інших встановлених сенсорів. Отримані дані слугують вхідними даними для системи, такі як зображення, відеопотоки та просторові координати.
- **Попередня (підготовча) обробка даних** – цей блок готує вхідні дані для основної нейронної мережі, очищаючи їх від шуму, нормалізуючи та виділяючи ключові ознаки. Обробка зображень може робити виділення контурів, патернів, фільтрацію шуму або вирівнювання контрасту, що допомагає у точності розпізнаванні об'єктів.
- **Аналіз і розпізнавання** – на цьому етапі вхідні дані (з попередніх пунктів) обробляються за допомогою згорткових нейронних мереж (CNN), або рекурентних мереж (RNN), залежно від задачі. CNN використовуються для аналізу зображень і відео, а RNN застосовуються для обробки послідовностей, наприклад, для відстеження об'єктів у реальному часі.

- **Блок (операції) прийняття рішень** – базується на принципах підкріплювального навчання, де система визначає оптимальну дію на основі нагород чи покарань (тобто свого досвіду), отриманих у попередніх епізодах. На основі даних з аналізу дрон приймає рішення про зміни траєкторії, захоплення цілі, відстеження або супроводження, або рішення про ураження цілі. Алгоритм підкріплювального навчання використовує спеціальну таблицю Q–значень або глибоку Q–мережу, які було описано раніше. Таблиця оновлюється після кожної дії залежно від отриманого результату (нагорода чи покарання). Це допомагає системі вчитися приймати кращі рішення з часом, і бути більш ефективною з кожною версією.



Рис. 2.1 Діаграма активностей (поведінка штучного інтелекту)

Як можемо бачити на рисунку 2.1, система штучного інтелекту працює як інтегрована структура взаємодії між сенсорними блоками, центральною обробкою даних та базою даних. Головна її мета – прийняття оптимальних рішень на основі аналізу поточних даних, попереднього досвіду та алгоритмів обробки інформації.

Алгоритми прийняття рішень штучним інтелектом можна описати як послідовність взаємодії між блоками, вони вказані на діаграмі, які можна описати наступним чином:

- **Отримування сигналу:** з датчиків або камери надходять тригери, що змушують розпочати процес роботи.
- **Збір даних і обробка вхідного сигналу:** сенсори передають дані в систему, де вони обробляються для усунення шумів і виділення ключових ознак.
- **Аналіз даних і розпізнавання:** оброблені дані передаються у CNN мережу, яка аналізує зображення чи відео для виявлення об'єктів або цілей.
- **Прийняття рішення на основі попереднього досвіду:** результати аналізу та попередні дії зберігаються у таблиці, або нейронній мережі, що дозволяє системі визначити оптимальну дію.
- **Збереження результатів у базу даних:** кожна дія має результат, який записується у базу даних як досвід. Далі він використовується при наступних прийняттях рішень, що отримати кращий результат. Також це впливає на швидкість прийняття рішення, оскільки якщо воно вже існує, то час не витрачається на аналіз, розрахунки та роздуми.
- **Навчання на основі результатів:** після кожного циклу система отримує зворотний зв'язок про успішність дії, що оновлює її знання для наступних ітерацій.

Завдячуючи такій архітектурі III та взаємодії між блоками забезпечується здатність дрона до автономного прийняття рішень на основі накопиченого досвіду. Це дозволяє йому ефективніше з кожним разом виконувати задачі у різних умовах.

Як згадувалось раніше у підрозділі 1.1.1, підкріплювальне навчання є найбільш ефективним у контексті дронів та штучного інтелекту для них, оскільки воно передбачує взаємодію агента із середовищем. Розглянемо діаграму з логікою навчання штучного інтелекту методом підкріплювального навчання:

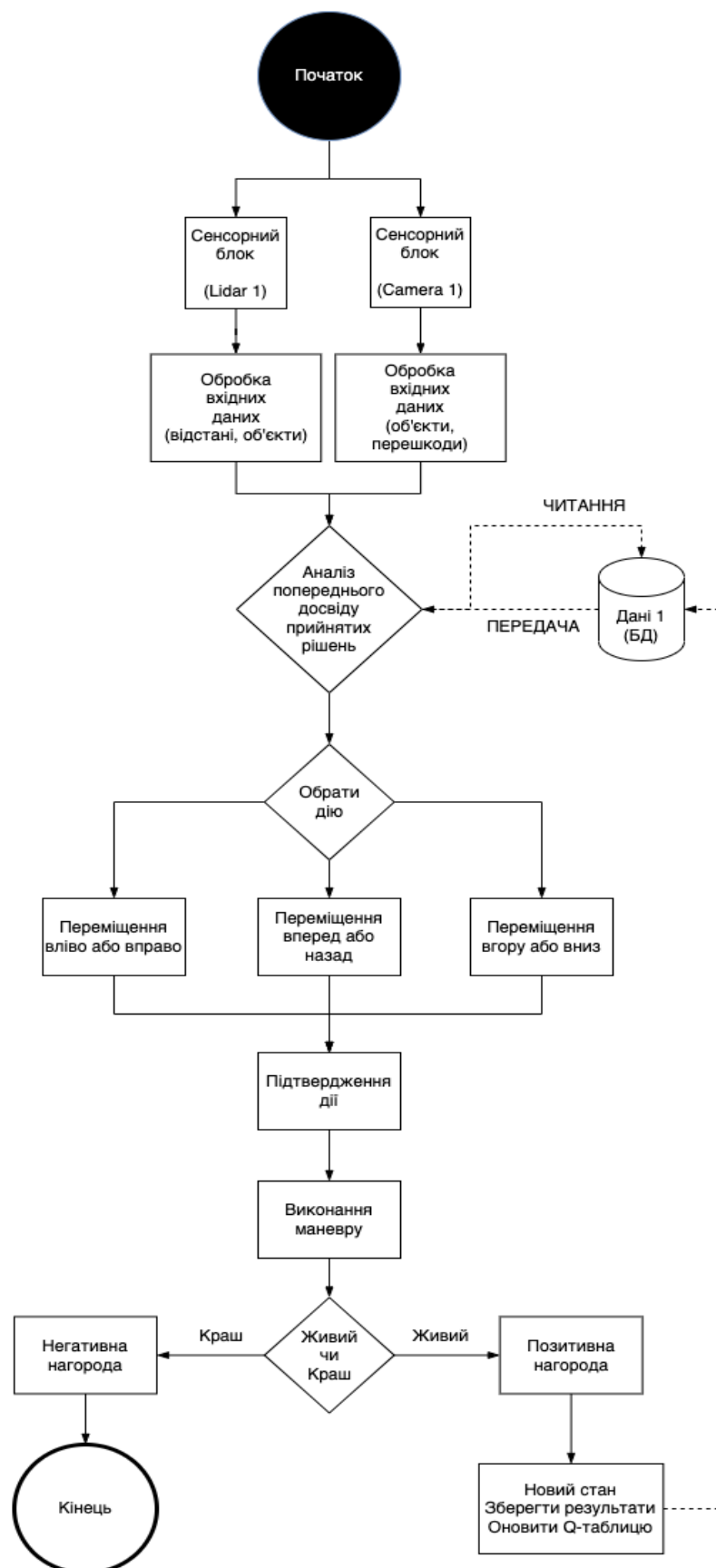


Рис. 2.2 Діаграма навчання через підкріплювальне навчання

Алгоритм навчання через підкріплювальне навчання складається з деяких етапів, які представлені на діаграмі, а описати їх ми можемо так:

- **Дані з датчиків:** інформація про навколишнє середовище надходить із блоків сенсорів, наприклад з lidar або камери. Вони надають актуальні дані про відстані, об'єкти і перешкоди.
- **Обробка вхідних даних:** сенсори збирають дані про відстані до об'єктів та бачать наявність перешкод. Це забезпечує систему обробки необхідною інформацією для передавання далі, та прийняття кінцевого рішення. Ці дані проходять попередню обробку для виділення ключових особливостей.
- **Попередній досвід рішень:** після обробки даних, система звертається до історії попередніх рішень та результатів. Аналізуючи досвід, система дивиться чи були вже такі сценарії, і може прогнозувати результат можливих дій.
- **Вибір дії:** після аналізу попереднього досвіду, і обрахунків ймовірного поточного рішення, система обирає оптимальну дію з доступних варіантів, щоб уникнути перешкод і досягти поставлених цілей.
- **Підтвердження дії:** обрана дія підтверджується, і система виконує дію.
- **Спостереження за новим станом:** після виконання дії система визначає, чи призвела ця дія до негативного результату ("dead end"), чи позитивного результату. На цьому етапі спостерігається новий стан системи після обраної дії.
- **Отримання винагороди та оновлення Q-таблиці:** далі система отримує зворотний зв'язок у вигляді позитивної чи негативної винагороди. Якщо дія була вдалою, система отримує позитивну винагороду, якщо ж ні, то надається негативна винагорода. Ця інформація зберігається і оновлює Q-таблицю, покращуючи знання системи для майбутніх рішень.
- **Продовження навчання:** відтак, якщо дія призвела до завершення процесу (зіткнення), програма зупиняється. Інакше, цикл повторюється з метою подальшого навчання та покращення точності прийняття рішень. Ці дані можуть передаватися та зберігатися віддалено, оскільки у випадку знищення

дрона локальні дані втрачаються. Це є також елементом захисту інформації, а також це надає перевагу для подальшого навчання системи, оскільки нові дані можна додати до БД і використовувати у симуляторі для навчання майбутніх моделей.

## 2.2 Загальна архітектура комп'ютерного зору

Computer vision є ключовою фішкою систем автономного управління, оскільки саме він дозволяє дрону аналізувати навколишнє середовище, розпізнавати об'єкти і приймати відповідні рішення в режимі реального часу. Загальна архітектура складається з кількох основних компонентів, кожен з яких виконує специфічні функції для захоплення та обробки зображень, ідентифікації об'єктів, які були внесені у базу знань ШІ, і забезпечення безпечного польоту дрона для виконання кінцевої задачі.

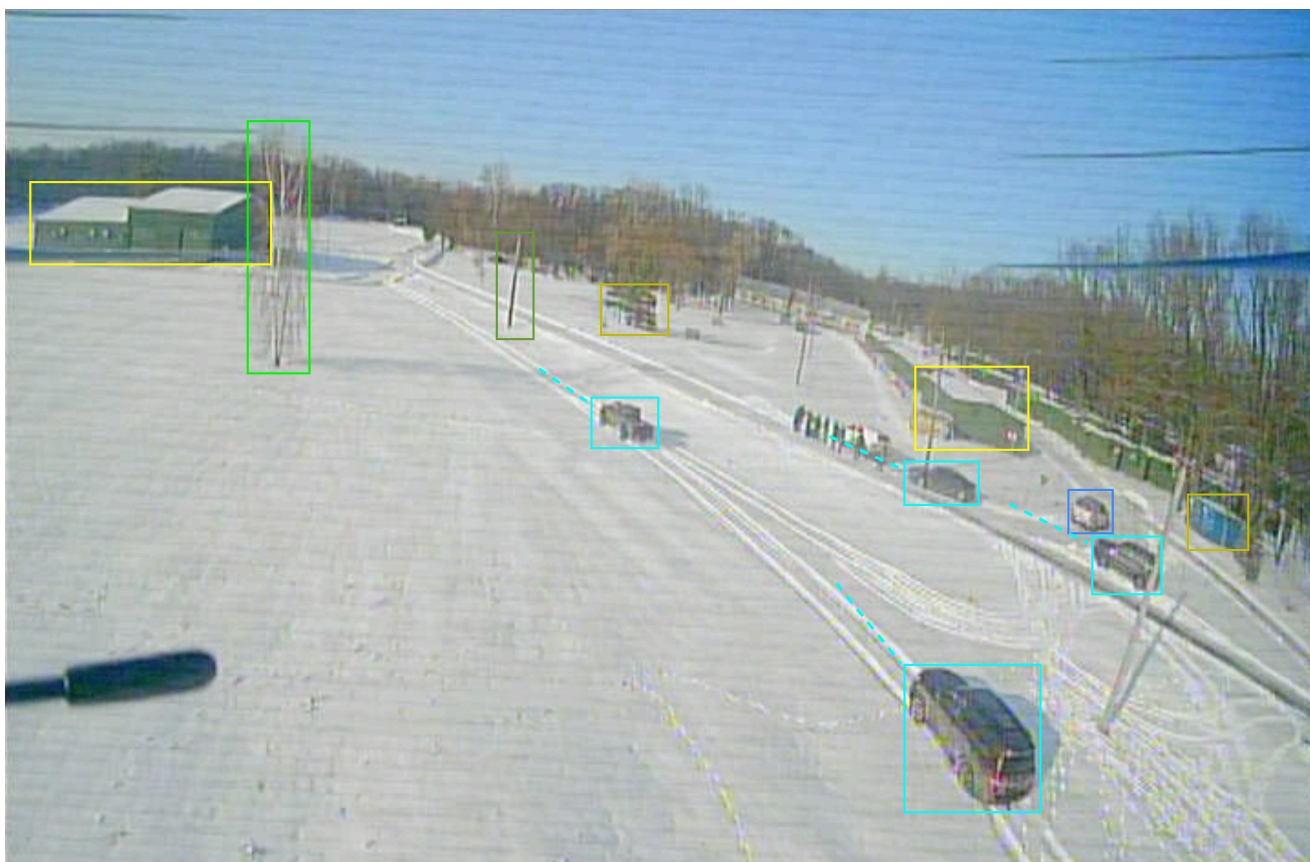


Рис. 2.3 Розпізнавання різних об'єктів комп'ютерним зором



На рисунку 2.3 як комп'ютерний зір "бачить" і виділяє об'єкти на відео, одразу призначаючи їм категорії на основі бази даних. Розпізнавання має відбуватися дуже швидко, відповідно іноді це негативно впливає на точність. У разі сумнівів об'єкт виділяється тим самим кольором, але в темнішому відтінку. Наприклад, на зображенні дрон розпізнав дерево зліва, позначивши його яскраво-зеленим кольором. Правіше він виявив об'єкт, схожий на дерево, але через недостатню впевненість позначив його темно-зеленим відтінком. Такий підхід дозволяє пілоту швидко орієнтуватися: яскраві кольори привертають увагу до чітко ідентифікованих об'єктів, тоді як менш яскраві не відволікають зайвої уваги, залишаючись помітними на екрані.

Для отримання якісних даних про навколишнє середовище дрони оснащуються сучасними камерами з різними технічними характеристиками. Як правило використовуються легкі моделі, які не навантажують конструкцію дрона, але мають високу роздільну здатність і широкий кут огляду. Тобто ми отримуємо максимум з пропорції "якість – легкість". Такі камери, як правило, мають тепловізійний сенсор та режими для роботи в умовах низької видимості або вночі. Наприклад, камери Foxeer FT256 (Analog CVBS Thermal Camera) або TVC 384CA підтримують передачу тепловізійного зображення, оскільки мають відповідний сенсор. Це робить їх придатними для використання в різних операціях та завданнях. Розташування камер на дроні залежить від завдання: фронтальні камери підходять для аналізу шляху або навчання для швидкісних польотів через перешкоди, бічні камери — для моніторингу об'єктів у декількох зонах, а нижні — для посадки (рідко) або вивчення земної поверхні.

Розглянемо дві тепловізійні камери, які підходять для наших цілей: "Foxeer FT256 Analog CVBS Thermal Camera" та "Аналогова тепловізійна камера IRC-384CA". Нижче представлена порівняльна таблиця, яка дозволяє оцінити їхні характеристики.

## Результат порівняння наведених варіантів камер для CV

| Характеристика                                 | Foxeer FT256 Analog<br>CVBS Thermal Camera                                           | IRC-384CA                       |
|------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------|
| Тип детекції                                   | —                                                                                    | Неохолоджений оксид ванадію     |
| Роздільна здатність                            | 256×192                                                                              | 384×288                         |
| Частота кадрів                                 | 50 кадрів/сек                                                                        | 50 кадрів/сек                   |
| Об'єктив                                       | Atherma, фіксований фокус,<br>7 мм                                                   | F1.0/4.0 мм                     |
| Кути огляду (по горизонталі/вертикалі)         | 25° / 19°                                                                            | —                               |
| Частота оновлення                              | 25 Гц                                                                                | 50 Гц                           |
| Тип детектора                                  | 12 μм                                                                                | 12 μм                           |
| Діапазон довжин хвиль                          | 8 ~ 14 μм                                                                            | 8 ~ 14 μм                       |
| NETD (Noise Equivalent Temperature Difference) | <50 мк                                                                               | ≤40 мк @ F1 25°C                |
| Дисплей зображення                             | Гаряче біле                                                                          | —                               |
| Алгоритм зображення                            | Корекція нерівномірності (NUC), 3D шумопоглинання (3DNR), цифрове поліпшення деталей | —                               |
| Напруга живлення                               | 4.5 – 18 В                                                                           | 5 – 16 В                        |
| Споживання живлення                            | —                                                                                    | До 1.5 Вт                       |
| Відеовихід                                     | CVBS                                                                                 | PAL                             |
| Вібрація                                       | 6.06 г, 3 осі                                                                        | —                               |
| Удар                                           | Напівсинусоїда, 80 г/4 мс, 3 осі, 6 прямих                                           | —                               |
| Розміри (мм)                                   | 25.8 × 25.8 × 31.8                                                                   | 27 × 37 × 40                    |
| Вага (г)                                       | —                                                                                    | 74.0 (без кабелів)              |
| Робочий діапазон температур                    | —                                                                                    | –20°C до +60°C                  |
| Комплектація                                   | За запитом                                                                           | Камера, кабель 3–pin JST–SH1.25 |

Як бачимо, камера Foxeer FT256 має перевагу у мінімальній вазі, компактних розмірах та оптимальною роздільною здатністю для задач. Також вона має в собі деякі вбудовані алгоритми обробки зображення для корекції нерівномірностей та цифрового покращення деталей. Вона добре підходить для невеликих дронів, таких як First Person View.



Рис. 2.4 Камера Foxeer для CV задач

Про камеру IRC-384CA можемо сказати, що вона має значно вищу роздільну здатність, яка дозволяє отримувати більш детальне зображення. Її низьке споживання енергії також є перевагою, а широкий температурний діапазон робить її придатною та надійною для застосування в різних температурних умовах.



Рис. 2.5 Камера 384CA для CV задач

Вага IRC-384CA більша за Foxeer FT256, що збільшує навантаження на роторну систему дрона та підвищує енергоспоживання. Тому при виборі камери важливо враховувати не лише технічні характеристики, а й її вплив на енергоефективність та тривалість польоту.

## **3 СТВОРЕННЯ ІНСТРУМЕНТУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ТА КОМП'ЮТЕРНОГО ЗОРУ**

### **3.1 Засоби розробки**

Для розробки методів, які базуються на методах artificial intelligence та computer vision, важливо правильно обрати засоби розробки та середовища розробки. Це в подальшому буде впливати на ефективність реалізації та роботи алгоритмів, інтеграції з різним апаратним забезпеченням, можливістю тестувати софт, а також можливість вдосконалення системи. Вони мають відповідати як загальним вимогам, так і специфіці завдань у яких будуть використовуватись frv дрони. Раніше у цій роботі було описане загальне призначення дронів у наш час, та їх актуальні задачі. Засоби розробки у наш час покривають усі стихії, де використовуються дрони – водні операції, наземні та повітряні операції, тому людьми були створені інструменти розробки з метою покрити всі ці кейси застосування безпілотних систем.

#### **3.1.1 Загальні вимоги до засобів розробки та мови програмування**

Для створення інструменту, який базується на artificial intelligence та computer vision, необхідно забезпечити відповідність засобів розробки всім етапам роботи системи. Це включає не лише розробку алгоритмів і забезпечення їхньої продуктивності, але й інтеграцію з апаратними модулями дрона, стабільний обмін даними між компонентами, а також можливість масштабування та вдосконалення програмного забезпечення відповідно до нових викликів і потреб.

Сумісність із апаратними частинами дрона також є важливою характеристикою. Методи розробки повинні забезпечувати адекватну інтеграцію з різними компонентами встановленими на дрон, наприклад такими як сенсори (ультразвукові, або lidar системи), камери, включно із тепловізійним режимом, а також підтримувати протоколи обміну даними, які використовуються в цілосній

системі, зокрема MAVLink. Протокол MAVLink (Micro Air Vehicle Link) – стандартний протокол для обміну даними між дронами різного базування, та наземними станціями управління і прийому даних. Його універсальність дозволяє працювати з широким спектром апаратних комплексів, наприклад морського базування, повітряного та наземного, забезпечуючи передачу команд, телеметрії, стану модулів системи та інформації про загальний стан. Завдяки цьому дрони різного базування можуть коректно передавати необхідні дані і виконувати завдання, отримуючи інструкції та передаючи зворотний зв'язок від оператора у реальному часі.

Наявність широкого спектра бібліотек та інструментів є ще одним важливим аспектом при виборі засобів розробки, оскільки ми маємо мати вибір між інструментами щоб охопити усі потреби поставленої задачі кінцевого продукту. Наприклад, OpenCV – це потужна бібліотека з багаторічною історією розвитку та доповнень, яка забезпечує достатній набір функцій для роботи з візуальними даними, зокрема для аналізу зображень та патернів, відеопотоку, та побудови алгоритмів розпізнавання. Бібліотека NumPy використовується для швидкої обробки масивів даних і виконання числових розрахунків, це основа для навчання моделей комп'ютерного зору. Обидві ці бібліотеки є відкритими для використання та модифікацій, тож активно підтримуються спільнотою розробників та ентузіастів, що робить їх зручними для інтеграції в різні проекти, або прототипи.

Оскільки розробка методик управління дронами ведеться для вузькоспеціалізованих завдань, кросплатформеність не є обов'язковою вимогою. Ба більше, цим можна знехтувати, оскільки розробка адаптивності і кросплатформеності потребує часу. Основний фокус спрямований на забезпечення ефективності та стабільності роботи у конкретному середовищі, з конкретним обладнанням та софтовими рішеннями, які відповідають специфіці FPV дронів та задач, розрахованих на них.

Варто також сказати про вибір та використання мов програмування, які найкраще підходять до специфіки проєкту розробки методик для дронів, враховуючи використання штучного інтелекту та комп'ютерного розу. У межах цього проєкту основними мовами програмування було обрано Python та C++. Мова Python забезпечує простоту реалізації алгоритмів III та CV, а також багату екосистему бібліотек, що дозволяє нам безперешкодно створювати прототипи та тестувати їх. Мова C++ застосовується для роботи з низькорівневим апаратним (hardware) забезпеченням, виконання обчислювально інтенсивних операцій і створення графічного інтерфейсу користувача (UI). Ця мова буде забезпечувати нам оптимальну продуктивність і дозволяє об'єднати різні компоненти системи в єдину кінцеву функціональну структуру.

### **3.1.2 Середовища розробки Visual Studio Code та PyCharm**

Для забезпечення ефективного та комфортного процесу розробки додатків обраними мовами, важливо використовувати оптимальне середовище розробки. У нашому випадку вибір середовищ для розробки обмежився двома популярними IDE: PyCharm та Visual Studio Code.

PyCharm – це потужне інтегроване середовище розробки для Python, створене компанією JetBrains. Воно пропонує багатий набір функцій, включаючи підтримку керування версіями, інтеграцію з системами контролю версій (Git, Mercurial), автоматичну перевірку коду, автодоповнення, а також вбудовану підтримку обмеженого тестування, рефакторингу та злагодження коду. Завдяки цим можливостям PyCharm є ідеальним середовищем для розробки додатків на Python, особливо при роботі з алгоритмами штучного інтелекту та комп'ютерного зору. Оскільки це середовище розроблено тими ж людьми, що і сама мова програмування, воно оптимізоване для Python і має потужну підтримку численних бібліотек і інструментів.

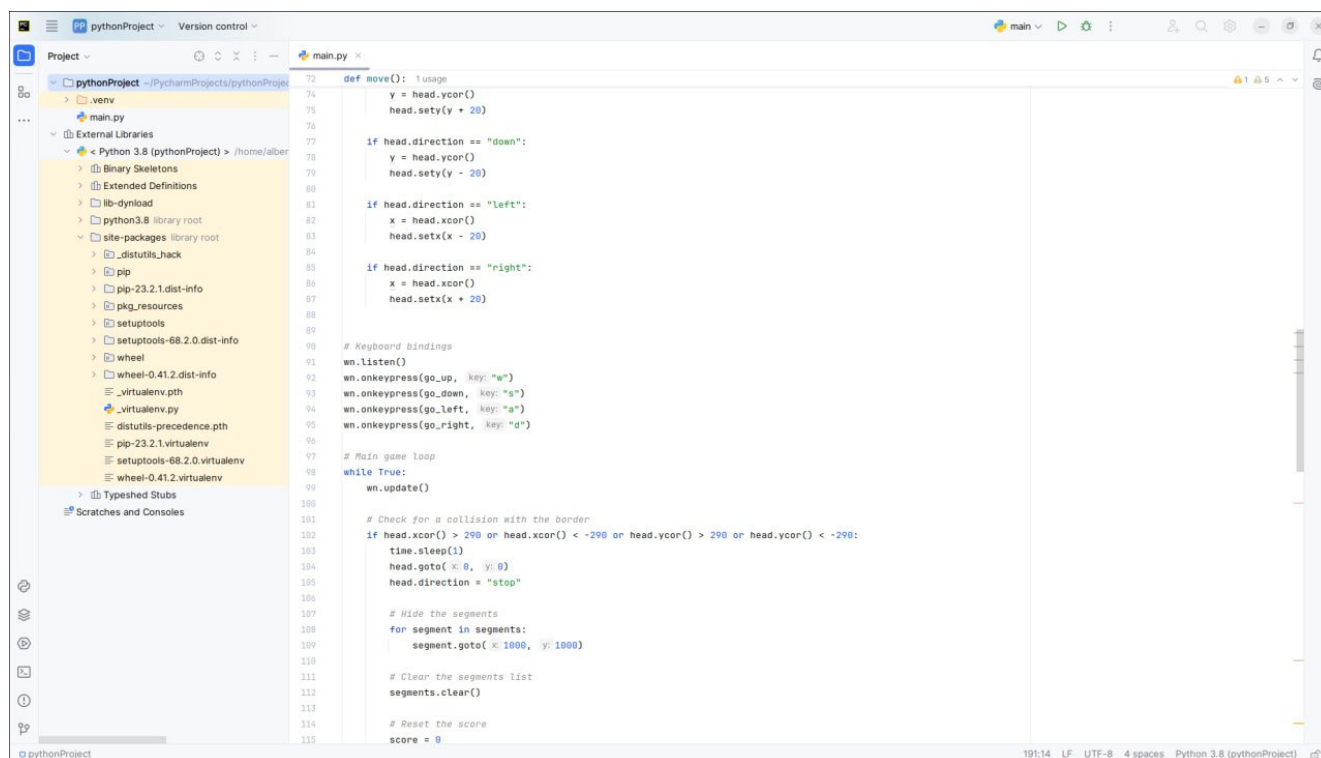


Рис. 3.1 Робоче середовище в PyCharm

Розглянемо Visual Studio Code – це легке, але потужне середовище розробки, яке підтримує безліч мов програмування, включаючи C++, Python, JavaScript та інші. VS Code розроблено компанією Microsoft і є безкоштовним і відкритим продуктом для користувачів. Його головною перевагою є модульність: юзер може додавати плагіни та розширення для роботи з будь-якою технологією, яку використовує у своєму проєкті. Для роботи з C++ та Python доступні плагіни для автодоповнення коду, налагодження, тестування та інтеграції з Git, що як і у випадку з PyCharm значно полегшує процес розробки. Це середовище є ідеальним варіантом для проєктів, де важлива легкість, потужність та швидкість роботи. Нижче наведено скріншот робочого середовища програми.

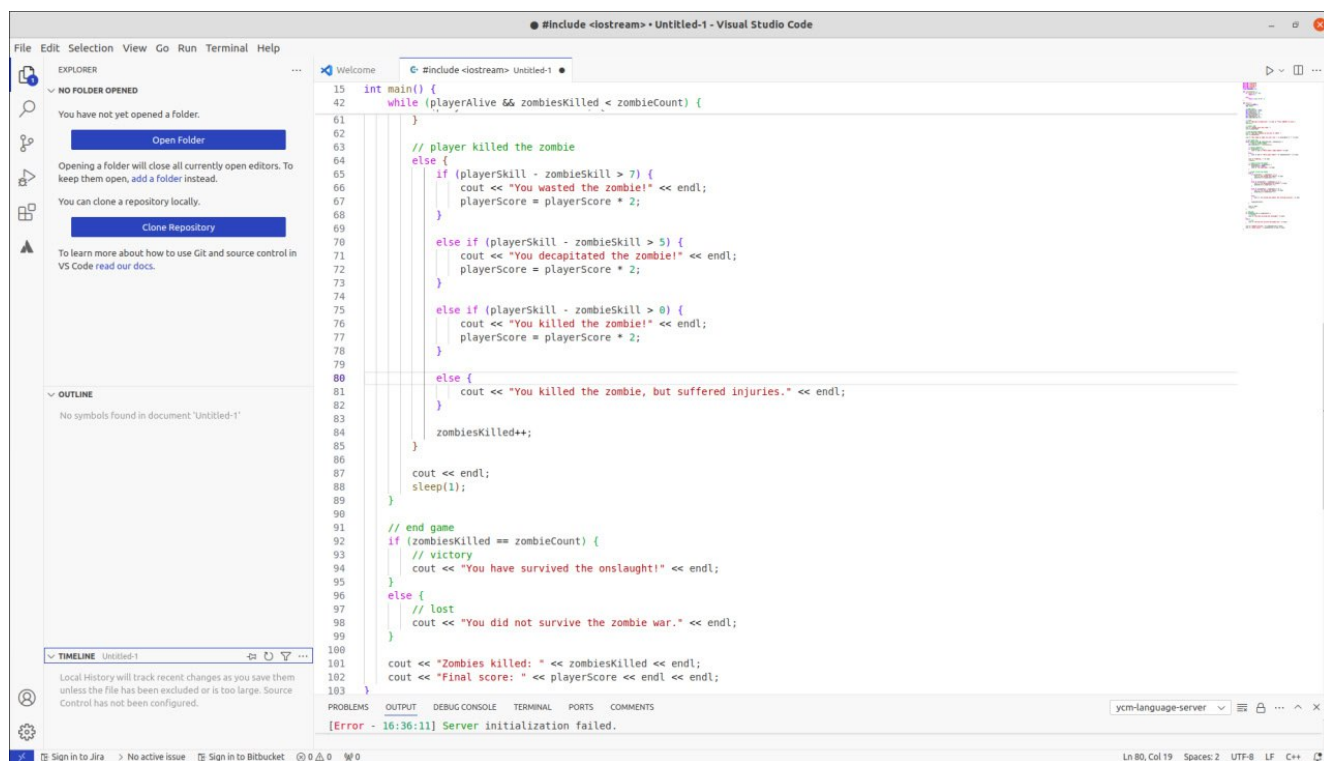


Рис. 3.2 Робоче середовище в Visual Studio

Обидва середовища розробки мають свої переваги: PyCharm ідеально підходить для розробки з використанням Python завдяки своїй глибокій інтеграції з мовою, тоді як Visual Studio Code пропонує універсальність і підтримку багатьох мов програмування та інструментів. Наш вибір саме цих IDE зумовлений специфікою проєкту та потреб кінцевого продукту. Обидва середовища підтримують різні плагіни та бібліотеки для роботи з AI і CV, що дозволяє ефективно працювати з даними та алгоритмічними методами.

### 3.1.3 Інструмент для симуляції AirSim

Процес налагодження та відпрацювання алгоритмів, що використовуються для керування дронами, потребує інструменту, який може моделювати реальні або наближені до реальних умови. AirSim – це відкритий симулятор від Microsoft, спеціально розроблений для віртуальних дронів та автомобілів. Він дозволяє тестувати алгоритми штучного інтелекту та комп'ютерного зору, ідеально підходячи для дослідження різноманітних сценаріїв, що наближені до реальних умов.



Симулятор має можливість налаштовувати параметри середовища, такі як погодні умови, час доби, тип місцевості (місто, сільська місцевість, пустеля, водні простори) та додавати перешкоди, які будуть міняти поведінку дрона. Все це дозволяє створювати різноманітні сценарії для тестування, включаючи ситуації з відсутнім GPS сигналом, відсутнім управлінням, втратою точки "Home", маневри в обмеженому просторі, або при несприятливих погодних умовах. Наприклад, для тестування та налагодження методик computer vision будуть корисними такі налаштування погоди як дуже яскраве сонце, чи сильні зливи і туман. Чим складніше – тим краще, бо штучний інтелект навчається і адаптується, тож при виконанні реального завдання шанси на успіх будуть значно підвищені.

Симулятор AirSim інтегрується з такими середовищами як Unity та двигуном Unreal Engine, що дає змогу створювати реалістичні графічні та фізичні моделі для симуляцій. Це забезпечує високу якість зображення та більш реалістичну поведінку дронів на об'єкти, як от автомобілі, катери, рухомі цілі, що дає можливість тестувати потрібні аспекти, такі як планування траєкторії, обхід перешкод, виявлення та супровід цілі, а також одночасне навчання нашого підкріплювального алгоритму для автоматичного управління.

Для створення та налаштування симуляцій в AirSim використовуються мови програмування Python та C++. Python використовується для написання скриптів та автоматизації процесів, завдяки своїй простоті та широкій підтримці бібліотек. C++ застосовується для реалізації складних та ефективних алгоритмів і інтеграції з хардовим забезпеченням.

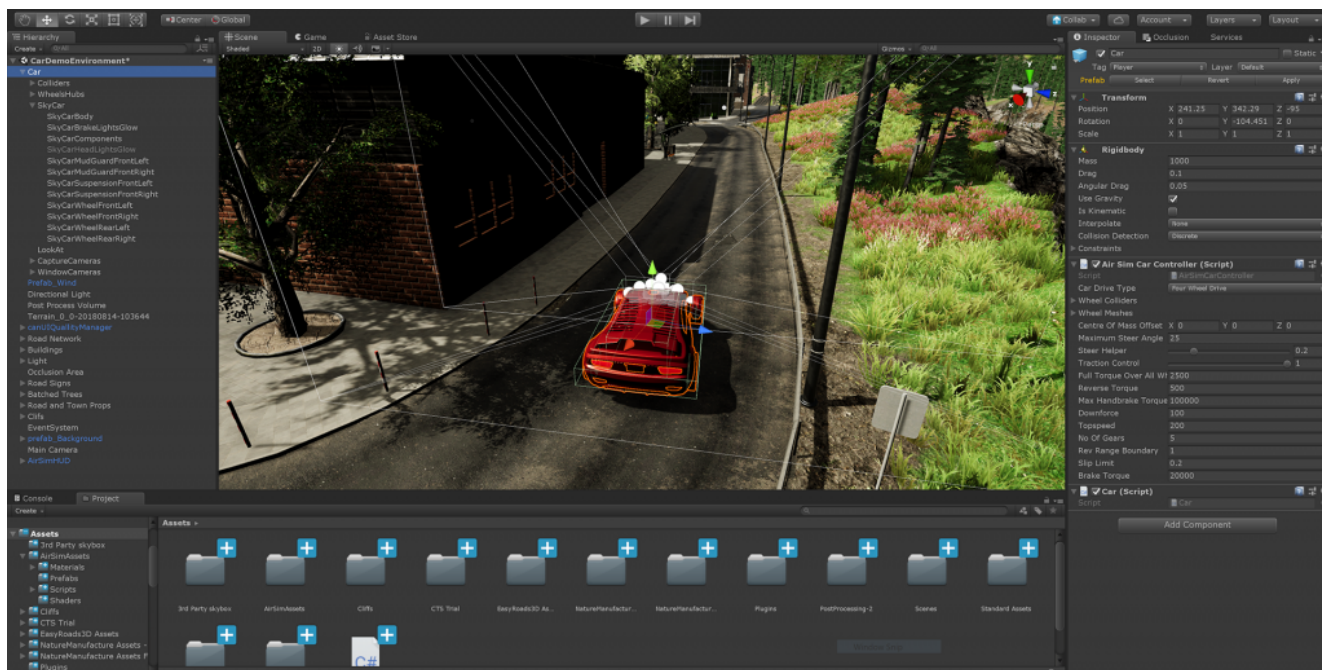


Рис. 3.3 Робоче середовище в AirSimulator

Середовище симулятора доволі схоже на IDE Visual Studio, тож проблем із роботою у досвідченого користувача виникати не має. Зліва все так само маємо провідник нашої файлової системи, проект в цілому та його окремі файли. У правій частині знаходяться параметри, які міняються в залежності від того який об'єкт на даний момент обрав користувач. Знизу можемо бачити файли та об'єкти, які складають автомобіль в нашому випадку. По центру бачимо робочий простір та сам об'єкт уваги, його структуру, трасерні промені тощо.

### 3.2 Бібліотеки та інструменти

Для досягнення оптимально ефективності у розробці таких систем корисно використовувати спеціалізовані бібліотеки та інструменти. Вони забезпечують нам зручність розробки, спрощують створення складних алгоритмів і якість роботи системи. Таким чином програмісти мають змогу ефективно та вчасно виконувати задачі для досягнення загального результату. Як ми знаємо, час – найцінніший ресурс, який не оглядається на помилки або успіхи. Він просто йде, і вміти ним користуватися дуже важливо.

### 3.2.1 Бібліотеки обробки зображень

Однією з головних бібліотек для реалізації завдань комп'ютерного зору є OpenCV. Вона надає потужний набір функцій для роботи з візуальними даними, наприклад обробка зображень та відео, розпізнавання рухомих об'єктів, аналіз траєкторій руху, побудова 3D-моделей. OpenCV активно використовується розробниками завдяки гнучкості, потужності та підтримці багатьох мов програмування, зокрема наших Python та C++. Бібліотека дозволяє інтегрувати алгоритми комп'ютерного зору у проєкт, при цьому підтримуючи високу продуктивність навіть якщо обчислювальні ресурси обмежені.

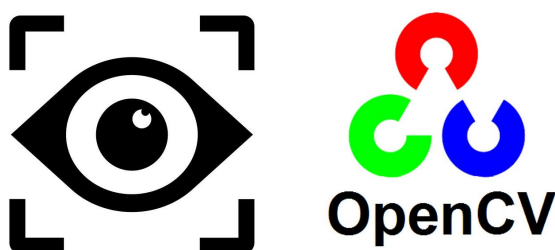


Рис. 3.4 Логотипи бібліотек OpenCV

Для базової, тобто попередньої обробки зображень ми будемо використовувати Pillow, це бібліотека мови Python, яка забезпечує прості методи роботи з візуальними даними. Вона може змінювати розмір, конвертувати зображення у різні формати, фільтрувати шуми та робити інші операції. Python Pillow зручний для попередньої обробки даних перед подачею їх на обробку алгоритмами комп'ютерного зору.



Рис. 3.5 Логотип бібліотеки Python Pillow

### 3.2.2 Бібліотеки для роботи з даними

Ефективна взаємодія з даними також важлива частина, тим паче якщо вона стосується розробки систем на основі artificial intelligence та computer vision. Для цього часто використовується бібліотека NumPy. Вона забезпечує оптимізацію обчислень із багатовимірними масивами даних. Завдячуючи її можливостям, розробники можуть виконувати математичні операції з високою точністю та швидкістю, що є важливим при обробці великих обсягів інформації. Бібліотека NumPy прискорює і спрощує процес обробки даних, дозволяючи ефективно використовувати ресурси при розробці.



Рис. 3.6 Логотип бібліотеки NumPy

Ще одним корисним та потужним інструментом для роботи з даними є бібліотека Pandas. Вона спеціалізується на діях із табличними даними (наприклад, Q-tables, які ми використовуємо). Вона надає зручні методи для створення звітів, аналізу результатів та їх валідації. Pandas використовується для оцінки ефективності алгоритмів, а також для аналізу їх роботи на етапі тестування. Її функціональність дає змогу швидко обробляти великі масиви вже структурованих даних, забезпечуючи при цьому точність результатів.



Рис. 3.7 Логотип бібліотеки Pandas

### 3.2.3 Інструменти для оптимізації

Етап оптимізації продуктивності системи теж потребує інструментів, і тому розробниками використовуються спеціалізовані інструменти, які дозволяють покращити алгоритми і підвищити їхню ефективність. Розглянемо Nuitka – компілятор Python коду, що перетворює його у високопродуктивні програми. За допомогою цього інструмента, при можливості, знижується навантаження на обчислювальні ресурси, що є важливим при роботі з масивними даними на апаратному забезпеченні дрона. Насправді, цей інструмент не користується широкою популярністю через те, що у створенні дронів першочерговим пріоритетом є кількість і стабільність, а не якість і продуктивність. Nuitka використовується при вже створених і налагоджених процесах, як потужне доповнення.



Рис. 3.8 Логотип бібліотеки Nuitka

Ще одним використовуваним інструментом є Line Profiler, який надає можливість аналізувати продуктивність написаного коду, визначаючи "вузькі місця", де оптимізації не буде зайвою. Цей інструмент незамінний у процесі налаштування системи, тому що дає змогу виявляти й усувати потенційні проблеми, що впливають на загальну швидкість роботи алгоритмів.

Використання усіх перелічених бібліотек та інструментів дозволяє розробнику створювати високоефективні системи для автономного управління дронами, залишаючи як бонус оптимальну продуктивність і точність роботи.

### 3.3 Розробка методів

Розробка методів, що інтегрують алгоритми штучного інтелекту та комп'ютерного зору, базується на модульному підході процесу. Такий підхід дозволяє створити гнучку систему, яка легко адаптується до змін середовища, оновлення алгоритмів або змін обладнання. Основною ідеєю архітектури є поділ функціоналу на незалежні модулі, кожен з яких відповідає за певний етап обробки даних, прийняття рішень і взаємодії з дронами.

#### 3.3.1 Архітектура створеного інструменту

Система складається з п'яти ключових модулів:

- **Модуль збору даних:** відповідає за отримання інформації з сенсорів та камер дрона, включаючи попередню обробку даних для усунення шумів і підготовки до подальшого аналізу.
- **Модуль *computer vision*:** виконує розпізнавання об'єктів (патернів) і аналіз отриманих візуальних даних.
- **Модуль *artificial intelligence*:** реалізує алгоритми прийняття рішень на основі даних, отриманих із *computer vision*, і раніше накопиченого досвіду.
- **Модуль навігації:** відбудовує маршрути для дрона, враховуючи інформацію від CV і AI.
- **Модуль *user interface*:** забезпечує зручний зворотний зв'язок із оператором і надає візуалізацію стану систем.

Нижче ми розглянемо код методів з елементами штучного інтелекту та комп'ютерного зору, який забезпечує автономне управління дроном. Основна увага приділена реалізації ключової логіки системи, яка включає алгоритми обробки зображень, аналіз даних, прийняття рішень та інтеграцію з хардверним забезпеченням дрона. На кожному етапі буде представлений код функцій системи, а також використання бібліотек, що оптимізують роботу.

### 3.3.2 Інтеграція та налаштування бібліотек у проект

**Збір даних.** Модуль збору даних отримує сигнали з відеопотоку камери і сенсорів. Для коректної роботи computer vision ми використовуємо камеру з підтримкою тепловізійного режиму, як було описано раніше. Ці дані попередньо оброблюються для усунення шумів і конвертуються у формат, більш зручний для подальшого аналізу. Цей етап важливий, оскільки він впливає на подальші операції та точності всієї системи.

```
# Імпортуємо cv бібліотеки
import global cv2
import opencv.h
import python.headless
import contrib.python

# Підключення до камери
if (global.connect = true && MAVLink.videoReciver = true) {
    ComVis.MAVLink.videoManager.Camera = true;
    init camera = cv2.VideoCapture(0);
    camera → check python.headless && check contrib.python;
    cv2.VideoCaptureTimer(start);
}

VideoCaptureTimer() {
# Читання кадру кожні 0.5 секунд
    interval: 500
    repeat: true
    setResult: NULL
    var framePerSec → camera.getFrame(500);

# Обробка ймовірної помилки
    if (!noResponse) {
        wait(2000);
        call MAVLink.videoReciver;
        framePerSec = request;
        processed_frame = cv2.GaussianBlur(frame, (5, 5), 0)
    } else {
        cvCritical << ("Camera not found!") << framePerSec;
    }
}
```



### 3.3.3 Реалізація алгоритмів комп'ютерного зору

**Обробка зображень.** Модуль CV виконує обробку отриманих даних, виділяє ключові об'єкти, та аналізує їхнє розташування у просторі. У разі низької якості зображення модуль застосовує алгоритм розмивання Гауса для усунення шумів, що дозволяє покращити розпізнавання об'єктів. Таким чином, камера може виявити перешкоду на шляху дрона, або "побачити" ціль, використовуючи алгоритми, і тоді на основі цих даних штучний інтелект зможе приймати подальші рішення.

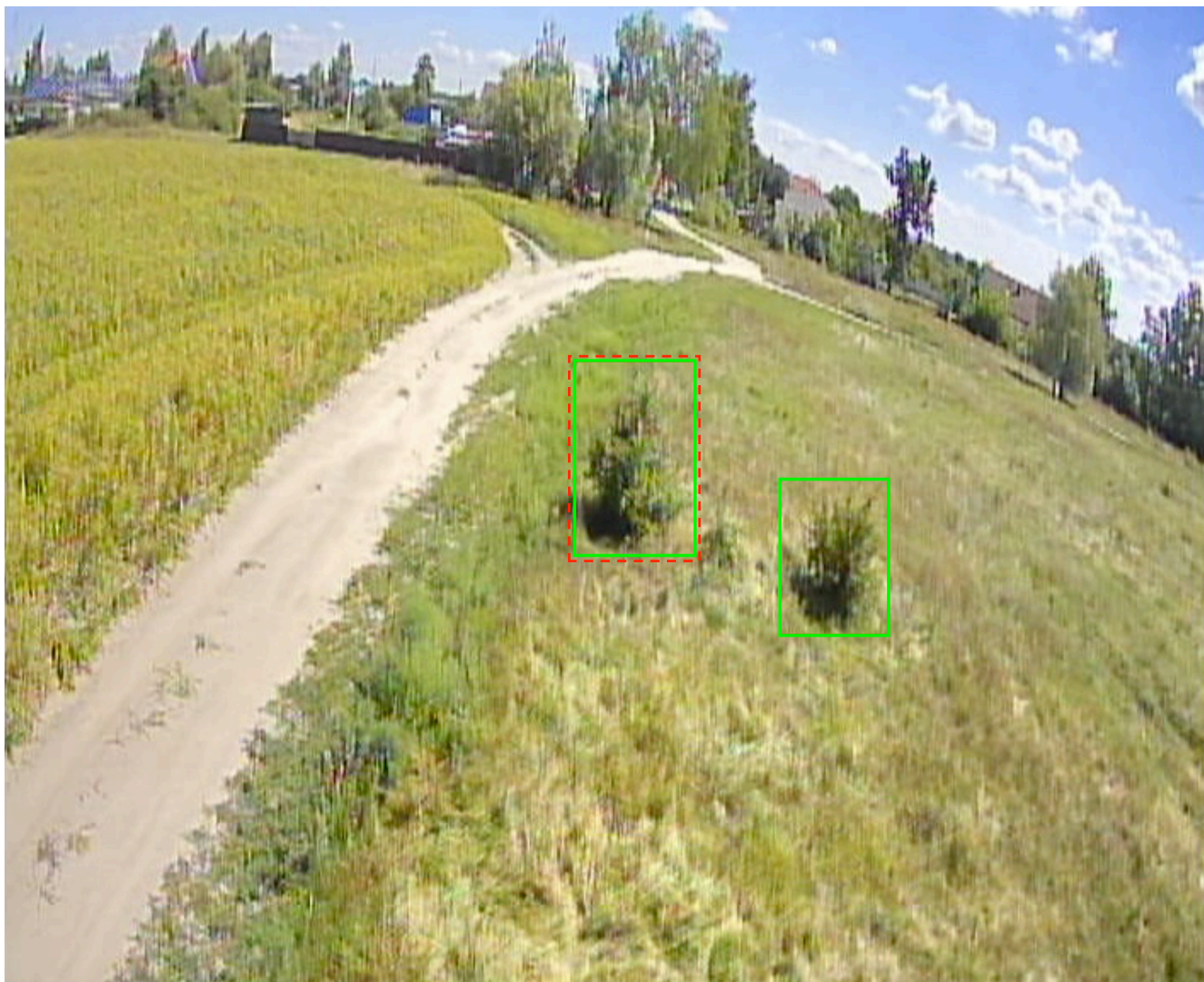


Рис. 3.9 Розпізнавання та обробка перешкод комп'ютерним зором

Як бачимо на світлинці вище computer vision, отримавши опрацьовану картинку з камери, розпізнав для патерна об'єктів, а саме невисокі дерева, і обвів їх у зелену рамку яскравого кольору. Це означає підтверджене розпізнавання



об'єкту, і для цього була задіяна функція "cv2.boundingRect". Взаємодіючи з алгоритмами штучного інтелекту метод CV зрозумів поточну та ймовірну майбутню траєкторію руху дрона, і виходячи з цих даних зробив висновок, що при подальшому такому русі зіткнення із перешкодою гарантоване. Тому, як ми бачимо, метод CV обвів у додаткову червону пунктирну рамку перешкоду, яка є потенційно фатальною.

Також нижче наведено приклад, як комп'ютерний зір розпізнав рухомий об'єкт у нічний час, який виявився зайцем. Пілот вирішив прослідкувати за твариною, підтвердивши дроні дію через UI. Алгоритм ШІ виділили тваринку у помаранчевий бокс і навіть намагався прогнозувати траєкторію руху.



Рис. 3.10 Захоплення рухомої цілі та спостереження за нею

Завдяки інтеграції комп'ютерного зору та штучного інтелекту дрон виділив зайця як "об'єкт інтересу" і виконує його супроводження, адаптуючись до змін у швидкості та напрямку руху об'єкта. Це демонструє можливості системи працювати навіть у складних умовах, таких як обмежена видимість, використовуючи інфрачервону камеру для отримання необхідних даних.



Рис. 3.11 Зрив захоплення цілі, сповіщення пілота

Через деякий час спостереження умови стали несприятливими для роботи алгоритмів штучного інтелекту та комп'ютерного зору. Наприклад, з'явилися значні шуми на зображенні, а праворуч за курсом видно ліс, який у нічній камері виглядає як суцільна темна пляма. Заєць, намагаючись сховатися, рухається у напрямку дерев. Сукупність цих факторів призвела до того, що дрон втратив здатність утримувати ціль, що викликало зрив захоплення. На екрані у помаранчевому прямокутнику з'явився напис "target loss", який повідомляє пілота

про втрату об'єкта. Червоною стрілкою на зображенні позначено поточне місце знаходження зайця, і варто зазначити, що алгоритм штучного інтелекту доволі точно спрогнозував напрямок його руху.

**Прийняття рішень.** Модуль AI отримує оброблену інформацію від CV і на її основі визначає оптимальну дію. Алгоритм підкріплювального навчання враховує поточну ситуацію, а також раніше накопичений досвід, щоб зробити найкращий вибір.

Як бачимо на Рисунку 3.9, дрон почав виконувати маневр ухилення вліво, враховуючи відстань до критичної перешкоди, швидкість свого руху, занижку висоту польоту та іншу перешкоду, яка знаходиться праворуч. Система проаналізувала всі доступні дані щоб обрати найкращий варіант дії, зважаючи на відсутність інших перешкод у напрямку ухилення (зліва).

```
# Вхідні дані з cv
detected_object = {
    "type": "tree",
    "distance": 11, # метри
    "size": "small",
    "direction": "straight"
}

# Дані для AI
drone_speed_now = 5; # м/с
reaction_time_possible = 1.2; # секунди
save_me.distance = 9; # метри

# Обчислення часу до зіткнення
time_to_collision = detected_object["distance"] / drone_speed;
```

```

# Визначення напрямку ухилення
def save_me.calculate(direction, distance) {
    if (detected_object.direction == "left") {
        save_me.trajectory = "right";
    }
    else if (detected_object.direction == "right") {
        save_me.trajectory = "left";
    } else {
        trajectory = "up"; # Ухилення вгору як останній варіант
    }
    return trajectory;
}

# Виконання маневру ухилення
if (time_to_collision <= reaction_time) {
    evasion_trajectory = calculate_evasion(detected_object["direction"],
detected_object["distance"]);
    aiCritical << ("Manoeuvring {evasion_trajectory} to avoid collision with
{detected_object['type']}.");
}

# Симуляція успішного маневру
time.sleep(1)
    aiCritical << ("Successfully avoided {detected_object['type']}. Logging
success...");

# Зберігання результату
result = {
    "event": "evasion",
    "object": detected_object["type"],
    "trajectory": evasion_trajectory,
    "status": "success"
} else {
    aiCritical << ("No actions required.");
}

```



**Автономна навігація.** Модуль навігації забезпечує стабільне позиціонування дрона, використовуючи дані, які надходять з GPS та GNSS, і обробляються AI та CV. У разі втрати зв'язку зі станцією керування, дрон може діяти автономно завдяки глобальному позиціонуванню, інтегрованому штучному інтелекту та наповненій базі даних.

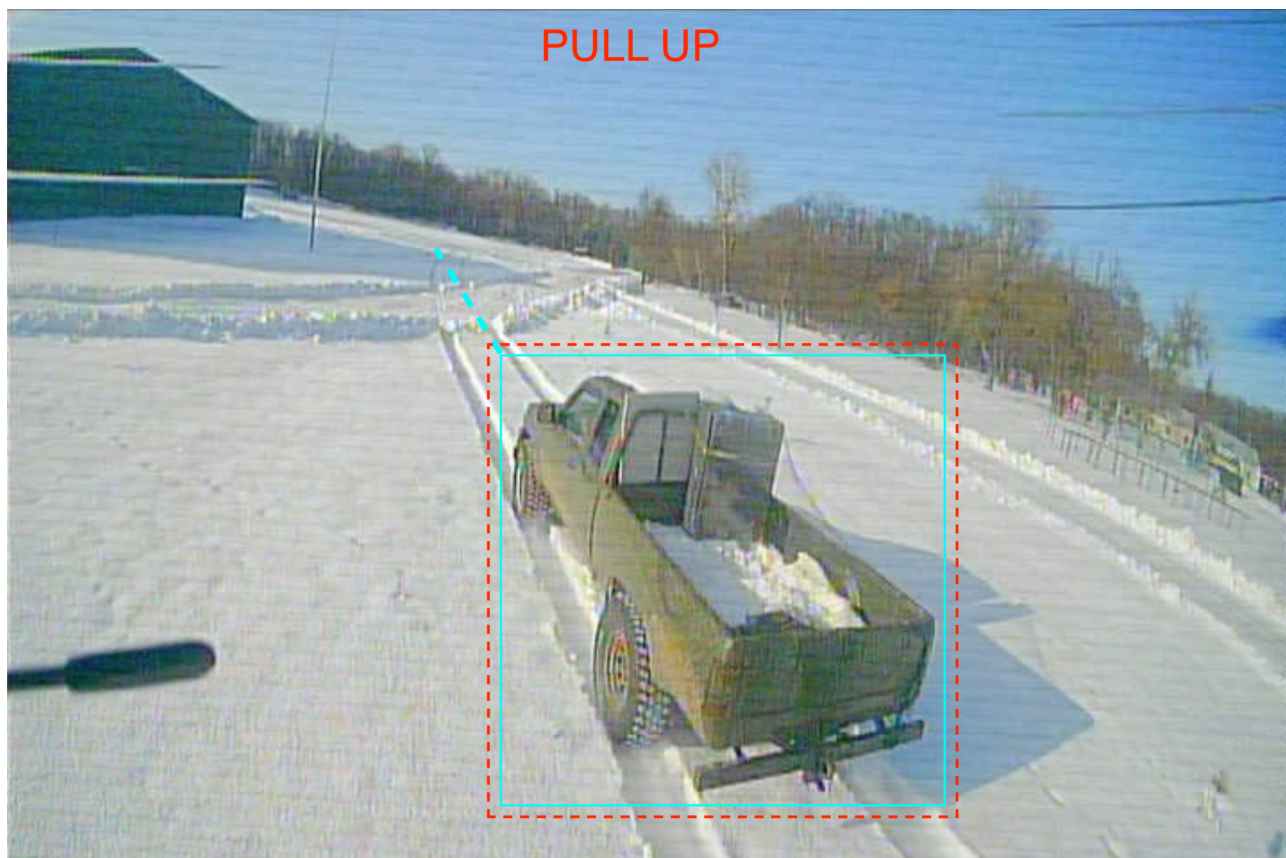


Рис. 3.12 Автономне виконання маневру ухилення за допомогою ШІ

При відсутності зовнішнього зв'язку ШІ дрона здатний аналізувати ситуацію в режимі реального часу, використовуючи дані з сенсорів, камер та інших джерел які передбачені конструкцією, для ухвалення рішень. Це означає що дрон буде визначати найбільш оптимальний маршрут, аналізувати перешкоди, і виконувати задачі ураження або спостереження, якщо вони були призначені заздалегідь. Наприклад, AI обчислює можливі траєкторії для обходу перешкод або прямування до цілі, враховуючи погодні умови, залишок напруги в АКБ, швидкість вітру та інші змінні середовища.



Рис. 3.13 Успішне автономне виконання маневру ухилення від перешкоди

Як видно на світлинах дрон виділив об'єкт, що перебуває на його шляху, тобто рухомий автомобіль і визначив, що для уникнення зіткнень необхідно виконати маневр вгору, оскільки рухаючись по глісаді втрачається висота. Таке рішення було прийняте штучним інтелектом через постійну оцінку дроном оточення протягом усього польоту – наразі висота занижка, збоків зустрічаються стовпи, а зверху перешкод виявлено не було. У процесі автономної роботи модуль AI також періодично намагається відновити зв'язок зі станцією керування, намагаючись забезпечити обмін даними і передати управління пілоту, якщо це можливо.

**Зворотний зв'язок.** Пілот отримує всю необхідну інформацію про стан дрона та його взаємодію із середовищем через інтерфейс у FPV-окулярах, якщо мова йде про frv дрони.



Рис. 3.14 Інтерфейс користувача при управлінні FPV дроном

Як показано на світлині, інтерфейс включає ключові параметри, такі як напруга акумулятора, висота польоту, кількість супутників GPS, рівень сигналу RSSI, залишковий час польоту та індикатор горизонту. Ці дані дозволяють пілоту оцінити ситуацію і приймати необхідні рішення. Всі дії пілот виконує за допомогою пульта керування, оснащеного кнопками, тумблерами та роликами. Це забезпечує йому зручне управління при будь-яких умовах. Показники на інтерфейсі оновлюються в реальному часі, дозволяючи відстежувати поточний стан дрона та результати виконання алгоритмів AI та CV.

## 4 ПРОЦЕС ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

### 4.1 Проведення тестування

Тестування розроблених методів проводилося у два основні етапи: у симуляторі AirSim та на реальному полігоні. Для кожного з етапів використовувалися всі можливі інструменти, щоб максимально покрити всі можливі кейси і сценарії.

**Симулятор AirSim.** Симулятор дозволяє змодельовати різноманітні ситуації та змінювати різні параметри, такі як стан модулів дрона, середовища та інших зовнішніх факторів. Під час тестування проводився аналіз роботи методів в умовах обмеженої видимості, відсутності GPS сигналу, низького рівня заряду АКБ, а також при імітації раптової втрати зв'язку з оператором. За допомогою інструментів програми було протестовано поведінку дрона при різних погодних умовах симуляції, час доби із різним ступенем освітлення, швидкість вітру і траєкторія руху деяких об'єктів. Для перевірки методів комп'ютерного зору на точність та швидкість виявлення цілей використовувалися статичні та рухомі об'єкти. Наприклад, у симулятор було додано автомобілі, які рухалися по заданим траєкторіям, і які дрон має уникати, супроводжувати або влучити у них. Крім того, були створені сценарії з перешкодами, такими як дерева, будівлі чи навіть інші дрони. Ці такі тест кейси дозволили нам оцінити наскільки ефективно система адаптується та справляється до різних змінних умов середовища і поведінки цілі.

Оскільки симулятор є гнучким інструментом, вдалося також перевірити попередню поведінку штучного інтелекту у випадку втрати зв'язку з пілотом, тобто наземною станцією. Дрон реагував очікувано, ніяких негативних відхилень не було помічено.





Рис. 4.1 Процес тестування методів у програмі AirSim

Як бачимо інтерфейс програмидоволі простий і зрозумілий для користувача, що дозволяє ефективно користуватися програмою. Маємо відкритий світ, мапу, деякі інформаційні дані дрона, а також перегляд поточної картини у різних фільтрах: звичайний режим, тепловізійний режим та CV режим. Нижче представлена таблиця результатів тестування дрона з розробленими методами ШІ та CV у симуляторі AirSim.

Таблиця 4.1

Результат поведінки дрона у симуляційному середовищі

| Умови тестування        | Точність наведення (%) | Надійність (%) | Автономність (хв) |
|-------------------------|------------------------|----------------|-------------------|
| День, ясна погода       | 95                     | 98             | 21                |
| Ніч, обмежена видимість | 88                     | 93             | 20                |
| Відсутність GPS         | 72                     | 85             | 17                |

**Полігон.** Після успішного тестування в симуляторі та отримання позитивних результатів для необхідних тест кейсів, дрон перевіряється у реальних умовах на полігоні. Було проведено два тестових виїзди: влітку та взимку. Це було необхідно для перевірки системи та комп'ютерного зору за різних погодніх умов.

Під час літніх випробувань акцент робився на аналіз поведінки системи в умовах відкритих просторів, де рухалися автомобілі, а також на взаємодію з перешкодами, таких як дерева, кущі, стовпи, деякі будівлі та птахи. Зимові тести включали складні погодні умови, такі як сніг, мороз, відбивання світла від ліхтарів, погане освітлення та сильний вітер. Це дозволило нам перевірити стійкість роботи штучного інтелекту та комп'ютерного зору за низьких температур і складніших обставинах. Також варто зазначити, що стан акумуляторної батареї теж є важливим у дроні.

Влітку температура навколишнього середовища та повітря була високою, і доходила до 35°C, проте завдяки постійному контакту АКБ з повітрям вона не перегрівалася. Напочатку зими, коли випадав перший сніг та вдарили перші морози, стан АКБ значно погіршився, оскільки він банально замерзав. Допомогало попередньо перед польотами тримати акумуляторну батарею в теплому місці, але все ж низькі температури впливали на час польоту дрона.

Таким чином було відпрацьовано всі тест кейси та сценарії польоту, які потребувалися щоб надати розробленим методам статус "passed" і забезпечити подальше їх використання користувачами.



Рис. 4.2 Випробувальні польоти біля полігону (літо)

У цей важкий та небезпечний час польоти на полігонах не можуть виконувати всі бажаючи. Процедура отримання дозволу на дані дії доволі складна і вирішується та координується з відповідними відповідальними назначеними особами. Також полігон має свої обмеження та правила, тож ми як відповідальні пілоти маємо дотримуватись їх і неухильно виконувати вказівки назначених відповідальних осіб. Як бачимо на рисунку 4.2, в об'єктив камери потрапило декілька хатин, але відстань до них безпечна і звук роторів не заважає людям, які там проживають. Полігони обмежені у своїх розмірах та мають зазначені шляхи використання повітряного простору. У випадку якщо у локації знаходження оголошується повітряна тривога – ми негайно маємо виконати приземлення дрона.

і сповістити про це відповідальну особу. Таким чином ми не заважаємо роботі Повітряних Сил України та не наражаємо на небезпеку себе та інших. За невиконання правил користування визначеними полігонами, або їх нехтуванням, гарантовано можна отримати проблеми. Нижче розглянемо таблицю результатів полігонних випробувань.

Таблиця 4.2

## Результат поведінки дрона на полігоні

| Умови тестування          | Пора року | Точність наведення (%) | Надійність (%) | Автономність (хв) |
|---------------------------|-----------|------------------------|----------------|-------------------|
| День, ясна погода         | Літо      | 92                     | 96             | 42                |
| Ніч, обмежена видимість   | Літо/Зима | 87/85                  | 91/89          | 38/29             |
| Сніг, сильна відбиваність | Зима      | 70                     | 85             | 34                |

## 4.2 Аналіз результатів

Проведені тестування в умовах симулятора та на реальному полігоні показали гарну ефективність розробленої системи автономного управління дроном. Алгоритми ІІІ та CV продемонстрували надійність та точність виконання поставлених задач, здатність дронів адаптуватися до змінюваних умов середовища та автономно приймати рішення у різних сценаріях. Використання розроблених методів штучного інтелекту та комп'ютерного зору дозволяє дрону з високою швидкістю аналізувати навколишнє середовище та виконувати складні маневри.

Проте виявлено певні обмеження, зокрема в умовах сильної відбиваності світла, що характерно для зимових умов з снігом. Це може призвести до виникнення фальшивих розпізнавань, що потребує подальшого вдосконалення алгоритмів обробки зображень і покращення адаптації системи до складних

погодних умов. Також важливим досвідом є те, що система добре показала себе у різних ситуаціях, де необхідно було ухилятися від перешкод і коригувати траєкторію, навіть коли дрон працює в автономному режимі та без GPS зв'язку.

Таблиця 4.3

## Результати порівняння методу Blind Bird із аналогами

| Умови тестування                   | Точність наведення (%) | Надійність (%) | Автономність (хв) |
|------------------------------------|------------------------|----------------|-------------------|
| You Only Looks Once                | 75                     | 88             | 9                 |
| Single Shot MultiBox Detector      | 80                     | 90             | 16                |
| <b>Розроблений метод BlindBird</b> | <b>85</b>              | <b>92</b>      | <b>до 21</b>      |

Враховуючи результати тестів можемо зробити висновок, що розроблені методи III та CV є перспективними для подальшої реалізації в реальних умовах. Наступними кроками можуть бути оптимізація алгоритмів штучного інтелекту та вдосконалення роботи алгоритмів комп'ютерного зору, а також інтеграція нових методик тестування для перевірки дії системи в інших сценаріях.



## ВИСНОВКИ

Метою даної магістерської роботи було розроблення інструменту для автономного управління дронами на основі методів штучного інтелекту та комп'ютерного зору. Основною задачею було створення загальної системи, яка поєднує необхідні технології для виконання поставлених задач з виявлення об'єктів та різними взаємодіями з ними, включаючи ухилення від перешкод, та виконання завдання без участі оператора. В процесі роботи було проведено аналіз доступних існуючих підходів та технологій у сфері автономного управління дронами, зокрема із застосуванням штучного інтелекту та комп'ютерного зору. Було виявлено, що більшість існуючих рішень не достатньо забезпечують адаптивність в умовах змінного середовища та високих вимог до точності в реальному часі. Крім того, багато з цих рішень страждають від низької швидкості обробки даних та не здатні ефективно працювати у складних погодних умовах. Рішенням цих проблем стало використання бібліотек з відкритим кодом та власні напрацювання, а також інтеграція з MAVLink для забезпечення надійного зв'язку з дронним обладнанням. Вибір таких інструментів та технологій дозволив реалізувати високу продуктивність та точність, а також ефективно інтегрувати ШІ з наявними системами. Розроблені методи, узагальнені локальною назвою Blind Bird, успішно пройшли тестування в симуляційному середовищі, а пізніше і у реальних умовах на полігоні, де були перевірені різні сценарії використання. Результати тестування показали, що система добре адаптується до змінюваних умов середовища, ефективно обробляє дані та забезпечує можливість автономного виконання завдань. Водночас, важливо зазначити, що проблема сонячної погоди і снігу продовжує залишатися складною для комп'ютерного зору, оскільки відображення світла може призводити до помилок у розпізнаванні об'єктів. Всі ці досягнення дозволяють створити надійне та ефективне рішення для автономного управління дронами, яке може бути використано в ряді застосувань, включаючи військові, рятувальні та комерційні завдання.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Doe, J. (2021). Autonomous Target Detection for UAVs Using Computer Vision. *Journal of Artificial Intelligence Research*, 24(4), 145-167.
2. Smith, A. (2020). Machine Learning Techniques for UAV Navigation. *Proceedings of the International Conference on Unmanned Aerial Systems*, 12(3), 98-112.
3. Brown, R. (2019). Advances in Computer Vision for Autonomous Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6), 1234-1248.
4. Lee, H., & Kim, J. (2020). Deep Learning Approaches for UAV Path Planning and Target Recognition. *Journal of Field Robotics*, 37(8), 1256-1275.
5. Zhang, Y., & Li, S. (2018). Real-Time Object Detection and Tracking for Autonomous Drones. *International Journal of Computer Vision*, 126(9), 1123-1140.
6. Wang, T., & Chen, L. (2019). Reinforcement Learning for Autonomous UAV Navigation. *Neural Networks*, 110, 116-123.
7. Johnson, M. (2018). Enhancing UAV Autonomy with Integrated AI Systems. *Journal of Robotics and Automation*, 35(2), 145-159.
8. Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*.
9. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 6105-6114.
10. Jacob, B., et al. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704-2713.
11. Sandler, M., et al. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510-4520.

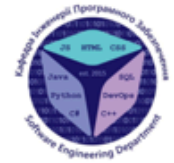
12. Rastegari, M., et al. (2016). XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. Proceedings of the European Conference on Computer Vision, 525-542.
13. Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.
14. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
15. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
16. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).
17. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
18. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.
19. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... & Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.



## ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО–  
КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ



НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ  
ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### МАГІСТЕРСЬКА РОБОТА

РОЗРОБКА МЕТОДИКИ АВТОМАТИЗОВАНОГО ВИЯВЛЕННЯ ЦІЛІ ТА НАВЕДЕННЯ ДРОНА  
З ВИКОРИСТАННЯМ КОМП'ЮТЕРНОГО ЗОРУ ТА МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент групи ПДМ–61 Диндар Альберт Валентинович

Керівник роботи: к.т.н, доц, доцент кафедри ІТ Трінтіна Наталя Альбертівна

Київ – 2025

### МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – вдосконалення процесу автономного наведення дронів за допомогою розробленої методики виявлення цілей на базі комп'ютерного зору та штучного інтелекту.
- **Об'єкт дослідження** – процес автономного управління дроном на основі методик комп'ютерного зору та штучного інтелекту.
- **Предмет дослідження** – методи виявлення цілей та наведення для дронів, що дозволяють мінімізувати участь оператора в управлінні.

## ПОРІВНЯННЯ АНАЛОГІВ

| Алгоритм   | Переваги                                              | Недоліки                                                  |
|------------|-------------------------------------------------------|-----------------------------------------------------------|
| CNN        | Висока точність у розпізнаванні об'єктів              | Висока потреба в обчислювальних ресурсах                  |
| YOLO       | Виявлення об'єктів у реальному часі                   | Обмежена точність для дрібних або накладених об'єктів     |
| SSD        | Баланс між точністю та швидкістю                      | Менша ефективність для складних сцен                      |
| Blind Bird | Здатність тримати ціль при поганих навколишніх умовах | Велика кількість навчальних епізодів, довгий час навчання |

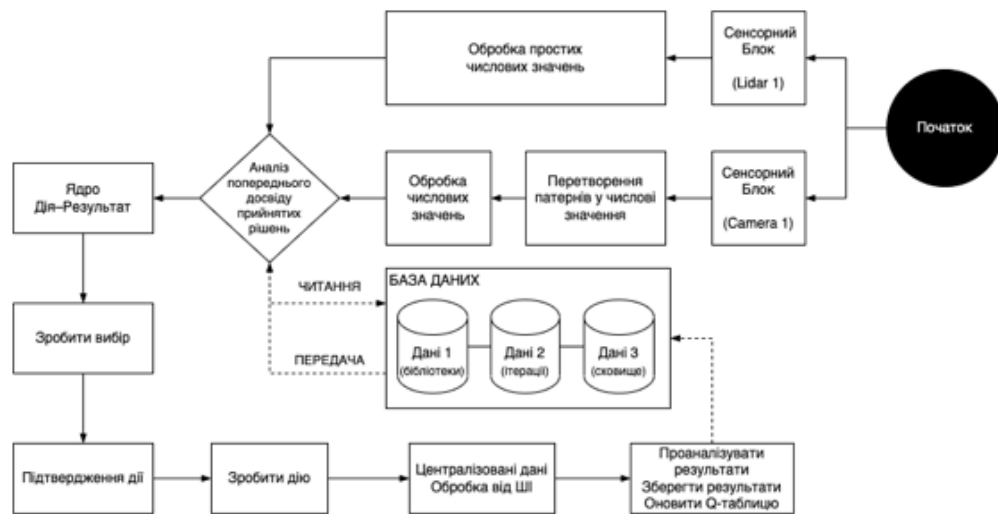
X

## ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



X

## ДІАГРАМА ПОВЕДІНКИ ШТУЧНОГО ІНТЕЛЕКТУ



X

## ДІАГРАМА НАВЧАННЯ ШТУЧНОГО ІНТЕЛЕКТУ



X

## МОДЕЛЬ МЕТОДУ ІМПОРТУ ТА ЗАДІЯННЯ CV



X

## ПОРІВНЯЛЬНИЙ АНАЛІЗ

| Умови тестування           | Точність наведення (%) | Надійність (%) | Автономність (хв) |
|----------------------------|------------------------|----------------|-------------------|
| Нормальні умови            | 92 / 85                | 95 / 85        | 20 / 16           |
| Обмежений GPS сигнал       | 85 / 40                | 90 / 50        | 18 / 13           |
| Відсутній GPS сигнал       | 75 / 10                | 82 / 34        | 15 / 10           |
| Наявність перешкоди        | 80 / 80                | 88 / 56        | 17 / 15           |
| Різноманітні погодні умови | 78 / 69                | 85 / 79        | 16 / 14           |

X

## **РЕЗУЛЬТАТ РОБОТИ РОЗРОБЛЕНИХ МЕТОДІВ**



X

## **ВИСНОВКИ**

1. Проведено аналіз поточних методів CV та AI, які застосовуються для автономного управління дронами.
2. Визначено та обрано оптимальні середовища розробки, мови програмування, бібліотеки й інструменти для реалізації поставлених завдань.
3. Розроблено методи та алгоритми для покращення виявлення цілей та автономного наведення дронів із використанням алгоритмів комп'ютерного зору та підкріплювального навчання.
4. Проведено тестування розроблених методів в симуляційному середовищі та оцінено ефективність роботи алгоритмів в полігонних умовах.

X

## АПРОБАЦІЯ МАГІСТЕРСЬКОЇ РОБОТИ

### Тези:

1. Диндар А. В. Методи автономного виявлення цілей та наведення дрона за допомогою комп'ютерного зору та штучного інтелекту // Міжнародний електронний науково-практичний журнал «WayScience», 07.11.2024, online конференція, с. 83.

2. Диндар А. В. Вибір нейронної мережі для виявлення цілі з урахуванням енергоспоживання дрона // Міжнародний електронний науково-практичний журнал «WayScience», 07.11.2024, online конференція, с. 85.

