### НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО

В.о. завідувача кафедри

Олександр КОВАЛЬ

« » 202\_р.

Дипломна робота

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»**

**спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Автоматична система визначення координат та типу рухомого об’єкту з квадрокоптера»**

Виконав:

студент IV курсу, групи ТІ-01

Красільнік Іван Павлович

(прізвище, ім’я, по батькові) (підпис)

Керівник:

ст. викл. Сарибога Г.В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Київ – 2024

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики Кафедра інженерії програмного забезпечення в енергетиці Рівень вищої освіти перший (бакалаврський) Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер - фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Олександр КОВАЛЬ

(підпис)

« » 202\_р.

**ЗАВДАННЯ**

**на дипломну роботу студенту**

Красільнік Іван Павлович

(прізвище, ім’я, по батькові)

1. Тема роботи

«Автоматична система визначення координат та типу рухомого об’єкту з квадрокоптера»

керівник роботи

ст. викл. Сарибога Г.В.

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ ” 202\_ року №

1. Строк подання студентом роботи
2. Вихідні дані до роботи\_мова програмування Pythont, алгоритм визначення об’єктів YOLOv8, середовище розробки PyCharm.
3. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити)\_проаналізувати існуючі аналогічні системи; розробити датасет для моделі розпізнавання об’єктів з квадрокоптеру; натренувати модель для розпізнавання об’єктів, розробити алгоритм для визначення відстані до об’єкту.
4. Перелік ілюстративного матеріалу\_інтерфейси аналогічних систем,\_приклади реалізації архітектури моделі, приклади інструментів роботи з датасетом, приклади роботи методів визначення дистанції, приклади роботи

системи

1. Дата видачі завдання « » 202\_ р.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання дипломної роботи | Строки виконання етапів роботи | Примітка |
| 1 | Отримання завдання | 30.10.2023 | Виконано |
| 2 | Дослідження предметної  області | 01.03.2024-  15.03.2024 | Виконано |
| 3 | Постановка вимог до  проєктування системи | 18.03.2024-  22.03.2024 | Виконано |
| 4 | Дослідження існуючих рішень | 25.03.2024-  05.04.2024 | Виконано |
| 5 | Розробка програмного продукту | 15.04.2024-  26.04.2024 | Виконано |
| 6 | Тестування | 15.05.2023-  19.05.2023 | Виконано |
| 7 | Захист програмного продукту | 13.05.2023 | Виконано |
| 8 | Оформлення дипломної  роботи | 22.05.2023-  06.06.2023 | Виконано |
| 9 | Передзахист | 04.06.2023 | Виконано |
| 10 | Захист | 17.06.2023 | Виконано |

|  |  |  |
| --- | --- | --- |
| Студент |  | Іван КРАСІЛЬНІК\_ |
|  | ( підпис ) | ( ім’я, прізвище) |
| Керівник роботи |  | Ганна САРИБОГА\_ |
|  | ( підпис ) | (ім’я, прізвище) |

# РЕФЕРАТ

**Структура та обсяг дипломної роботи.** Робота містить 55 сторінок, 17 рисунків, 2 додатки та 13 посилань.

**Метою роботи** є розробка системи автоматичного визначення координат і типу рухомого об'єкта з квадрокоптера на базі сучасних технологій машинного навчання, зокрема використання моделі YOLOv8 та алгоритмів комп'ютерного зору. Система призначена для військового застосування, забезпечуючи точне та ефективне виявлення різних типів об'єктів у бойових умовах.

Розроблена інформаційна система надає користувачам зручний інтерфейс для автоматизованого виявлення та класифікації об'єктів, таких як люди, танки, транспортні засоби, вантажівки та автомобілі. Система використовує квадрокоптери для збору даних, що забезпечує високу мобільність та оперативність. Інтеграція з моделлю YOLOv8 та алгоритмом визначення відстані методом подібності трикутників дозволяє точно визначати координати та тип об'єктів на основі зображень, знятих у різних умовах.

**Практичне значення отриманих результатів** полягає у створенні готового продукту, який може бути використаний у військових операціях для розвідки, моніторингу та ідентифікації загроз. Система забезпечує точне виявлення та класифікацію об'єктів, що дозволяє оперативно приймати стратегічні рішення. Використання сучасних інструментів, таких як Roboflow для анотації даних та Google Colab для тренування моделі, забезпечує високу точність і ефективність системи. Крім того, інтуїтивно зрозумілий інтерфейс та можливість віддаленого доступу дозволяють легко керувати системою та отримувати необхідні дані в реальному часі.

**Ключові слова:** комп'ютерний зір, машинне навчання, YOLOv8, квадрокоптер, військова розвідка, Roboflow, Google Colab, Python, PyTorch, OpenCV, Ultralytics, анотація даних, алгоритм подібності трикутників.

# ABSTRACT

**Structure and volume of the thesis.** The thesis contains 55 pages, 17 figures, 2 appendices, and 13 references.

**The goal of the work.** The purpose of the work is to develop a system for automatic determination of coordinates and the type of moving objects from a quadcopter based on modern machine learning technologies, specifically using the YOLOv8 model and computer vision algorithms. The system is designed for military applications, ensuring precise and efficient detection of various types of objects in combat conditions.

The developed information system provides users with a convenient interface for automated detection and classification of objects such as people, tanks, vehicles, trucks, and cars. The system uses quadcopters for data collection, ensuring high mobility and operational efficiency. Integration with the YOLOv8 model and the distance determination algorithm based on the similarity of triangles method allows accurate determination of the coordinates and type of objects based on images captured in various conditions.

**Practical significance of the results.** The practical significance of the results lies in the creation of a ready-to-use product that can be employed in military operations for reconnaissance, monitoring, and threat identification. The system ensures precise detection and classification of objects, enabling prompt strategic decision-making. The use of modern tools such as Roboflow for data annotation and Google Colab for model training ensures high accuracy and efficiency of the system. Additionally, the intuitive interface and remote access capability allow for easy system management and real-time data acquisition.

**Keywords:** Computer vision, machine learning, YOLOv8, quadcopter, military reconnaissance, Roboflow, Google Colab, Python, PyTorch, OpenCV, Ultralytics, data annotation, triangle similarity algorithm.

# ЗМІСТ

[ВСТУП 9](#_bookmark0)

1. [ЗАДАЧА ПОБУДОВИ АВТОМАТИЧНОЇ СИСТЕМИ ВИЗНАЧЕННЯ](#_bookmark1) [КООРДИНАТ ТА ТИПУ РУХОМОГО ОБ’ЄКТУ З КВАДРОКОПТЕРА](#_bookmark1)

[. 10](#_bookmark1)

* 1. [Постановка задачі 10](#_bookmark2)
  2. [Аналіз існуючих рішень визначення типу об’єкта 11](#_bookmark3)
     1. [YOLOv8 11](#_bookmark4)
     2. [Faster R-CNN 14](#_bookmark5)
     3. [Mask R-CNN 15](#_bookmark6)
     4. [RetinaNet 17](#_bookmark7)
  3. [Аналіз існуючих рішень визначення відстані до об’єкту 20](#_bookmark8)
     1. [Стереозір (Stereo Vision) 20](#_bookmark9)
     2. [Монокулярна Візія (Monocular Vision) 21](#_bookmark10)
     3. [Тривимірні Камери (3D Cameras) 23](#_bookmark11)
     4. [Лідар (LiDAR) 24](#_bookmark12)
  4. [Вибір рішення для побудови автоматичної системи визначення координат та](#_bookmark13) [типу рухомого об’єкту з квадрокоптера 26](#_bookmark13)

[Висновки до розділу 1 28](#_bookmark14)

1. [АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ](#_bookmark15) [АВТОМАТИЧНОЇ СИСТЕМИ 30](#_bookmark15)
   1. [Вибір платформи розробки та мови програмування 30](#_bookmark16)
      1. [Мова програмування Python 31](#_bookmark17)
      2. [Бібліотека Ultralytics 32](#_bookmark18)
      3. [Бібліотека OpenCV 32](#_bookmark19)
   2. [Середовище розробки PyCharm 33](#_bookmark20)
   3. [Використання Roboflow для підготовки датасетів 34](#_bookmark21)
   4. [Використання Google Colab для тренування моделі 36](#_bookmark22)

[Висновки до розділу 2 37](#_bookmark23)

1. [ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 39](#_bookmark24)
   1. [Підготовка набору даних 39](#_bookmark25)
   2. [Тренування, валідація та тестування моделі у Google Colab 43](#_bookmark26)
   3. [Розробка алгоритму визначення дистанції методом подібності трикутників](#_bookmark27)

[. 45](#_bookmark27)

* 1. [Інтеграція моделі визначення типу рухомого об’єкту та алгоритму](#_bookmark28) [визначення дистанції 48](#_bookmark28)

[Висновок до розділу 3 49](#_bookmark29)

1. [РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ 51](#_bookmark30)
   1. [Системні вимоги 51](#_bookmark31)
   2. [Розробка нових версій датасету 51](#_bookmark32)

[Висновок до розділу 4 52](#_bookmark33)

[ВИСНОВКИ 54](#_bookmark34)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55](#_bookmark35)

[ДОДАТОК А 56](#_bookmark36)

[ДОДАТОК Б 66](#_bookmark37)

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ**

YOLOv8 (англ. You Only Look Once version 8) – Ви лише один раз дивитесь, версія 8.

CNN (англ. Convolutional Neural Network) – згорткова нейронна мережа. R-CNN (англ. Region-based Convolutional Neural Network) – регіонально-

згорткова нейронна мережа.

GPU (англ. Graphics Processing Unit) – графічний процесор. TPU (англ. Tensor Processing Unit) – тензорний процесор.

RPN (англ. Region Proposal Network) – мережа пропозицій регіонів. ROI (англ. Region of Interest) – регіон інтересу.

МРТ (англ. Magnetic Resonance Imaging) – магнітно-резонансна томографія. FPN (англ. Feature Pyramid Network) – мережа піраміди ознак.

ToF (англ. Time of Flight) – час польоту.

SBC (англ. Single Board Computer) – одноплатний комп'ютер.

API (англ. Application Programming Interface) – прикладний програмний інтерфейс.

IDE (англ. Integrated Development Environment) – ітегроване середовище розробки

# ВСТУП

Технологічний розвиток не стоїть на місці, велика кількість наукових досліджень розширює можливості та дає нам змогу вирішувати все складніші завдання. За допомогою програмування людина має можливість створювати різноманітні програмні системи та засоби, що забезпечує автоматизацію процесів. Це сприяє підвищенню продуктивності та точності у виконанні завдань, знижує ризик виконання помилок та пришвидшує робочі процеси.

Розробка системи для визначення типу рухомого об’єкту та його координат з квадрокоптера є актуальною задачею, оскільки така система може вирішувати завдання у різноманітних галузях, таких як військова та цивільна авіація, розпізнавання небезпеки, транспортування, екстрена медична допомога.

Автоматизація системи розпізнавання об’єктів та їх координат має великий потенціал для розвитку безпілотних літальних апаратів та їх застосування у військових місіях. Однак на даний час існує недостатньо рішень, які також мають певні недоліки, що ускладнює їх застосування на практиці.

У зв'язку з цим, метою даної роботи є розробка автоматичної системи визначення координат та типу рухомого об’єкту з квадрокоптера. Для досягнення мети, було проведено аналіз існуючих рішень та обрано оптимальний варіант побудови системи визначення типу рухомого об’єкту та дистанції до нього. Було розглянуто інструменти розробки, описана архітектура проекту та розроблені алгоритми для визначення координат та дистанції від квадрокоптера на основі даних з камери.

Також описано роботу з набором даних (dataset) та представлено результати роботи системи. Розроблена система може бути застосована відповідно до потреб військової сфери.

# ЗАДАЧА ПОБУДОВИ АВТОМАТИЧНОЇ СИСТЕМИ ВИЗНАЧЕННЯ КООРДИНАТ ТА ТИПУ РУХОМОГО ОБ’ЄКТУ З КВАДРОКОПТЕРА

## Постановка задачі

Задача передбачає створення автоматичної системи комп’ютерного зору для квадрокоптера, яка здатна автоматично визначати клас об'єкта та дистанцію до нього на місцевості за допомогою камери, що встановлена на квадрокоптері. Для цього необхідно розробити алгоритми для визначення координат об’єкту та розрахування відстані, а також розробити систему технічного зору для класифікації об’єктів.

Для розробки системи технічного зору для класифікації об’єктів з квадрокоптера, потрібно зібрати датасет зі знімками отриманими з дрона. Система повинна правильно визначати клас рухомого об’єкта при різноманітних факторах. Для забезпечення найкращих результатів може виявитися необхідним проведення серії тестів та валідаційних перевірок.

Окрім того, система повинна мати можливість відстежувати рухомий об’єкт та розраховувати дистанції до нього під час руху, забезпечуючи стабільність та точність. Для коректної роботи системи потрібно знати параметри камери, таких як фокус, оскільки це дозволяє адаптувати систему до різних моделей камер. Це робить систему універсальною та здатною працювати у різних умовах та на різних моделях дронів.

По завершенню цієї задачі буде розроблено автоматичну систему, яка забезпечує визначення класу рухомого об’єкту та дистанції до цілі. Така система може знайти широке застосування у багатьох сферах, включно з військовою галуззю, пошуково-рятувальними операціями, сільськогосподарською діяльністю та науковими дослідженнями.

## Аналіз існуючих рішень визначення типу об’єкта

Розробка автоматичної системи визначення типу об'єктів, що рухаються, з квадрокоптера, починається з вибору найбільш відповідної технології для її реалізації. Одним з найкращих способів визначення типу рухомого об’єкта є використання систем комп’ютерного зору, які можуть використовувати штучний інтелект і алгоритми машинного навчання для вирішення проблем візуального розпізнавання. Однак використання цього методу може мати недоліки: складнощі обробки та залежність від якості зображення, особливо для складних сцен.

### YOLOv8

Одним із можливих рішень є використання системи комп’ютерного зору YOLOv8. Вважається однією з найпопулярніших архітектур для виявлення об’єктів у реальному часі на зображеннях і відео. YOLOv8 є добре відомою системою завдяки високій швидкості обробки відеоданих і легкій інтеграції в різні системи та програми. Ця система характеризується високою швидкістю і високою точністю розпізнавання об'єктів. Варто зазначити, що YOLO розроблено як програмне забезпечення з відкритим вихідним кодом, тому користувачі можуть вільно використовувати та змінювати його для адаптації до своїх конкретних завдань.

Основні Особливості:

* + - * YOLOv8 забезпечує високу точність виявлення об'єктів за рахунок використання нових методів та оптимізацій.
      * Завдяки своїй архітектурі, модель також працює дуже швидко, що робить її придатною для застосувань в реальному часі.
      * В YOLOv8 використовуються покращені блоки конволюційних нейронних мереж та інші вдосконалення, які підвищують здатність моделі виявляти об'єкти різного розміру та форми.
      * Застосовуються методи агрегації та об’єднання, які покращують контекстуальне розуміння сцени.
      * Модель підтримує навчання на різних наборах даних і використовує сучасні методи оптимізації для підвищення якості навчання.
      * YOLOv8 може бути налаштована для різних апаратних платформ, від потужних серверів до мобільних пристроїв, що дозволяє використовувати її в різних середовищах.
      * Модель легко інтегрується з популярними фреймворками машинного навчання та бібліотеками, такими як PyTorch або TensorFlow.
      * Підтримуються інструменти для розгортання на серверах, мобільних пристроях та в хмарних середовищах.

Переваги YOLOv8:

* + - * Висока Точність: YOLOv8 використовує покращені методи виявлення об’єктів для забезпечення високої точності навіть у складних умовах. Завдяки вдосконаленій архітектурі та методикам навчання модель показує чудові результати на різних наборах даних.
      * Швидкість: Ця модель оптимізована для роботи в режимі реального часу, що особливо важливо для програм, чутливих до затримки. Швидкість обробки зображень дозволяє використовувати YOLOv8 у програмах, які вимагають негайної реакції, наприклад: Приклади: відеоспостереження та безпілотні автомобілі.
      * Масштабованість: YOLOv8 є універсальним, оскільки його можна налаштувати для різноманітних апаратних платформ, від високопродуктивних серверів до мобільних пристроїв. Моделі підтримують різні масштаби, тому їх можна використовувати для завдань різної складності та розміру даних.
      * Інтеграція з Популярними Фреймворками: Інтеграція з популярними фреймворками: Ця модель легко інтегрується з такими фреймворками, як PyTorch

і TensorFlow, що полегшує її впровадження в існуючі системи машинного навчання. Підтримка сучасних кросплатформних засобів розгортання.

* + - * Гнучкість: Параметри та функції тонкого налаштування дозволяють YOLOv8 адаптуватися до конкретних завдань і умов зйомки. Підтримка доповнення даних і різних методів попередньої обробки може покращити якість навчання моделі.

Недоліки YOLOv8:

* + - * Високі вимоги до обчислювальних ресурсів: Навчання YOLOv8 вимагає потужних графічних процесорів (GPU) і великої кількості обчислювальних ресурсів, що може бути дорогим. Великі набори даних також вимагають великих обсягів пам'яті та дискового простору.
      * Складність налаштування та оптимізації: Оптимізація моделі для конкретного завдання може бути складною та вимагає глибоких знань машинного навчання. Оптимізація гіперпараметрів і архітектури моделі для досягнення максимальної продуктивності може зайняти значну кількість часу та зусиль.
      * Вимоги до даних: Досягнення високої точності вимагає великих і різноманітних навчальних наборів даних, які може бути важко зібрати. Якість анотацій і різноманітність даних безпосередньо впливають на продуктивність моделі.

YOLOv8 — це потужний інструмент виявлення об’єктів із багатьма перевагами, такими як висока точність, швидкість і гнучкість. Однак є недоліки, такі як висока вартість обчислювальних ресурсів і складне налаштування та вимоги до даних. Тим не менш, YOLOv8 залишається одним з провідних рішень для багатьох завдань комп'ютерного зору..

### Faster R-CNN

Faster R-CNN — це вдосконалена версія архітектури згорткової нейронної мережі на основі регіону (R-CNN), яка використовується для завдань виявлення об’єктів. Ця модель показує значні покращення продуктивності та точності порівняно з попередніми версіями R-CNN і Fast R-CNN.

Основні Особливості:

* + - * Головна інновація Faster R-CNN полягає у введенні мережі пропозицій регіонів (RPN), яка генерує пропозиції об'єктів.
      * RPN є повністю конволюційною мережею, яка працює безпосередньо на виході з базової конволюційної мережі, генеруючи пропозиції об'єктів значно швидше, ніж попередні методи.
      * Модель використовує спільне навчання RPN і детектора об'єктів, що дозволяє підвищити ефективність та точність обох компонентів.
      * Спільне використання базової конволюційної мережі для обох задач значно зменшує обчислювальні витрати.
      * Faster R-CNN досягає значного підвищення швидкості обробки порівняно з попередніми версіями завдяки інтеграції RPN.
      * Модель здатна обробляти зображення в реальному часі або близько до нього на сучасних графічних процесорах.
      * Faster R-CNN складається з базової конволюційної мережі (наприклад, ResNet), RPN, а також регіональної мережі на основі ознак (Region of Interest, RoI) для класифікації та регресії об'єктів.

Переваги Faster R-CNN:

* + - * Faster R-CNN демонструє високу точність об'єктного детектування завдяки ефективному використанню RPN та глибоких конволюційних мереж.
      * Висока якість пропозицій регіонів забезпечує більш точне виявлення об'єктів.
      * Завдяки інтеграції RPN, модель значно швидша за попередні версії R- CNN та Fast R-CNN.
      * Модель здатна працювати в режимі реального часу при використанні сучасних апаратних ресурсів.
      * Faster R-CNN може бути налаштована для різних завдань об'єктного детектування та класифікації.
      * Модель підтримує різні архітектури базових конволюційних мереж, що дозволяє налаштувати її для специфічних застосувань.
      * Спільне використання базової мережі для RPN та детектора об'єктів зменшує обчислювальні витрати та час обробки.

Недоліки Faster R-CNN:

* + - * Автономні Транспортні Засоби: Виявлення об'єктів на дорозі, таких як пішоходи, інші автомобілі та перешкоди.
      * Системи Відеоспостереження: Моніторинг та аналіз відео для виявлення підозрілої активності або об'єктів.
      * Медична Діагностика: Виявлення патологій на медичних зображеннях, таких як рентгенограми або МРТ.
      * Розпізнавання Обличчя: Використання в системах безпеки та ідентифікації.

Faster R-CNN залишається одним із найпопулярніших методів виявлення об’єктів завдяки своїй точності та ефективності, але він також має обмеження, такі як високі вимоги до обчислювальних ресурсів і складність впровадження.

### Mask R-CNN

Mask R-CNN (Mask Region-Based Convolutional Neural Network) є подальшим розвитком архітектури Faster R-CNN, призначеним для задач сегментації об'єктів. Вона може не лише виявляти об'єкти та класифікувати їх, але й визначати точні контури (маски) кожного об'єкта на зображенні.

Основні Особливості:

* + - * Mask R-CNN розширює Faster R-CNN додатковою гілкою для сегментації об'єктів.
      * Крім виявлення об'єктів і їх класифікації, модель також виконує піксельну сегментацію.
      * До мережі додається паралельна гілка для прогнозування бінарної маски для кожного виявленого об'єкта.
      * Ця маска використовується для визначення точних контурів об'єкта на зображенні.
      * Mask R-CNN вводить нову операцію під назвою RoIAlign (Region of Interest Align), яка покращує точність вирівнювання ознак.
      * RoIAlign зменшує помилки, викликані квантовими округленнями під час вирівнювання ознак, що значно підвищує точність сегментації.
      * Модель складається з базової конволюційної мережі (наприклад, ResNet), Region Proposal Network (RPN), та двох гілок: одна для класифікації і регресії об'єктів, і друга для сегментації масок.

Переваги Mask R-CNN:

* + - * Mask R-CNN забезпечує високу точність як виявлення об'єктів, так і їх сегментації завдяки використанню RoIAlign та паралельної гілки для масок.
      * Модель дозволяє отримувати точні контури об'єктів, що є критично важливим для багатьох застосувань.
      * Модель може бути використана для різних задач комп'ютерного зору, таких як семантична сегментація, інстанційна сегментація, та навіть keypoint detection (виявлення ключових точок).
      * Підтримка різних архітектур базових конволюційних мереж.
      * Mask R-CNN може бути застосована до різних типів об'єктів і сцен, що робить її корисною для широкого спектра завдань від автономних транспортних засобів до медичної діагностики.

Недоліки Mask R-CNN:

* + - * Навчання та інференція Mask R-CNN вимагають значних обчислювальних ресурсів, включаючи потужні графічні процесори (GPU).
      * Великий обсяг пам'яті та часу обробки, особливо для великих зображень з багатьма об'єктами.
      * Реалізація та налаштування Mask R-CNN є складною через багатокомпонентну архітектуру.
      * Налаштування гіперпараметрів і оптимізація моделі для досягнення найкращих результатів можуть зайняти багато часу.
      * Незважаючи на високу точність, Mask R-CNN може бути не підходящою для задач, де важлива мінімальна затримка, через значний час обробки.

Mask R-CNN є потужною та гнучкою архітектурою для задач об'єктного детектування та сегментації. Вона забезпечує високу точність та універсальність, що робить її придатною для широкого спектра застосувань. Однак, вона також має свої обмеження, зокрема високі вимоги до обчислювальних ресурсів та складність реалізації. Незважаючи на це, Mask R-CNN залишається одним із провідних методів для складних задач комп'ютерного зору.

### RetinaNet

RetinaNet є популярною архітектурою для задач об'єктного детектування, розробленою для вирішення проблеми дисбалансу між класами об'єктів, яка часто зустрічається в таких задачах. Основною інновацією RetinaNet є використання Focal Loss, яка значно покращує продуктивність моделі при виявленні рідкісних об'єктів.

Основні Особливості:

* + - * Основна новація RetinaNet - це Focal Loss, яка змінює стандартну функцію втрат для того, щоб приділяти більше уваги складним для класифікації об'єктам.
      * Focal Loss зменшує вагу легко класифікованих прикладів і збільшує вагу важких, що дозволяє моделі краще виявляти рідкісні та важкі для розпізнавання об'єкти.
      * На відміну від двоступеневих моделей, таких як Faster R-CNN, RetinaNet є одноступеневою моделлю, що робить її швидшою та простішою в реалізації.
      * Використання Feature Pyramid Network (FPN) для побудови багатомасштабних ознак забезпечує ефективне виявлення об'єктів різного розміру.
      * RetinaNet використовує FPN для створення багатомасштабних ознак, що дозволяє моделі ефективно працювати з об'єктами різних розмірів.
      * FPN покращує здатність моделі до виявлення дрібних об'єктів, що є важливим для багатьох задач комп'ютерного зору.

Переваги RetinaNet:

* + - * Використання Focal Loss дозволяє RetinaNet досягати високої точності при виявленні рідкісних та важких для класифікації об'єктів.
      * Це особливо важливо для задач, де дані сильно дисбалансовані.
      * Як одноступенева модель, RetinaNet працює швидше за двоступеневі моделі, такі як Faster R-CNN.
      * Це робить її придатною для застосувань, де важлива швидкість обробки, наприклад, в реальному часі.
      * Використання FPN забезпечує ефективну роботу з об'єктами різних розмірів, що робить модель універсальною для різних типів завдань.
      * Модель може бути налаштована для різних архітектур базових мереж (наприклад, ResNet), що забезпечує гнучкість у виборі компромісу між швидкістю та точністю.

Недоліки RetinaNet:

* + - * Як і інші сучасні моделі для об'єктного детектування, RetinaNet вимагає значних обчислювальних ресурсів для навчання.
      * Використання складних архітектур базових мереж може призвести до підвищених вимог до пам'яті та обчислень.
      * Налаштування Focal Loss і вибір гіперпараметрів можуть бути складними та вимагати експериментів для досягнення найкращих результатів.
      * Оптимізація моделі для специфічних задач може потребувати значних зусиль.
      * Хоча RetinaNet швидша за двоступеневі моделі, вона може бути повільнішою за деякі інші одноступеневі моделі, такі як YOLO, у задачах реального часу.
      * Для задач, де затримка є критично важливою, можуть знадобитися додаткові оптимізації.

RetinaNet є потужною архітектурою для об'єктного детектування, яка вирішує проблему дисбалансу між класами об'єктів завдяки використанню Focal Loss. Вона забезпечує високу точність при виявленні рідкісних об'єктів та має переваги одноступеневої моделі, такі як швидкість та простота реалізації. Незважаючи на високі вимоги до обчислювальних ресурсів та складність налаштування, RetinaNet залишається одним із провідних методів для задач об'єктного детектування в різних галузях.

## Аналіз існуючих рішень визначення відстані до об’єкту

Визначення відстані до об'єкту за допомогою камери є важливим завданням у багатьох областях, таких як робототехніка, автономні транспортні засоби, доповнена реальність та системи безпеки. Існують різні методи та технології для розв'язання цієї задачі, кожен з яких має свої переваги та недоліки.

### Стереозір (Stereo Vision)

Стереозір використовує дві камери, розташовані на певній відстані одна від одної (базова лінія), для отримання зображень однієї й тієї ж сцени. Відстань до об'єкта визначається за допомогою триангуляції на основі різниці положень об'єкта на двох зображеннях (диспаратності).

Процес:

1. Збір даних: Камери одночасно знімають одну й ту ж сцену.
2. Калібрування: Визначення точного розташування та орієнтації камер одна відносно одної.
3. Пошук відповідностей: Виявлення відповідних точок на обох зображеннях.
4. Обчислення диспаратності: Визначення різниці положень відповідних точок.
5. Триангуляція: Обчислення відстані до кожної точки на основі відомої базової лінії та диспаратності.

Переваги:

* + Точність на малих відстанях: Стереозір забезпечує високу точність вимірювань на невеликих відстанях.
  + Тривимірна інформація: Можливість отримання повної тривимірної моделі сцени.

Недоліки:

* + Обчислювальна складність: Потребує значних ресурсів для обробки та пошуку відповідностей між зображеннями.
  + Чутливість до умов освітлення та текстур: Стереозір менш ефективний при недостатньому освітленні або на однорідних поверхнях.

Застосування:

* + Робототехніка для навігації та уникнення перешкод.
  + Автономні транспортні засоби для виявлення об'єктів на дорозі.
  + Відеоігри та доповнена реальність для створення тривимірних сцен.

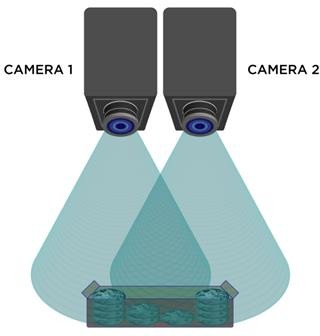


Рисунок 1.1 – приклад роботи стереовізора

### Монокулярна Візія (Monocular Vision)

Монокулярна візія використовує одну камеру для оцінки відстані до об'єктів. Це досягається за допомогою машинного навчання, оптичного потоку, геометричних припущень або методів перспективної проєкції.

Процес:

* + - * Збір даних: Отримання зображення однією камерою.
      * Аналіз зображення: Використання алгоритмів машинного навчання або інших методів для оцінки відстані на основі особливостей зображення.
      * Калібрування: Налаштування камери для точного вимірювання відстаней.

Переваги:

* + - * Простота апаратної частини: Використання лише однієї камери зменшує вартість та складність системи.
      * Можливість використання існуючих камер: Не потребує спеціального обладнання.

Недоліки:

* + - * Менша точність: Відстань до об'єкта оцінюється менш точно, особливо на великих дистанціях.
      * Складність виявлення на великих дистанціях: Зниження точності на великих відстанях або для об'єктів з низьким контрастом.

Застосування:

* + - * Доповнена реальність для розміщення віртуальних об'єктів у реальному світі.
      * Автономні транспортні засоби для додаткової оцінки відстані до об'єктів.
      * Системи безпеки та відеоспостереження для відстеження об'єктів.



Рисунок 1.2 – приклад використання монокулярної візії

### Тривимірні Камери (3D Cameras)

Тривимірні камери використовують технології, такі як Time-of-Flight (ToF) або структуроване світло, для отримання тривимірної інформації про сцену. ToF камери вимірюють час, який потрібно світлу для досягнення об'єкта і повернення до сенсора, тоді як камери зі структурованим світлом проектують відомий візерунок на сцену і аналізують його деформації.

Процес:

* + - * Випромінення лазерного або світлодіодного імпульсу.
      * Вимірювання часу, за який світло повертається до сенсора.
      * Обчислення відстані на основі швидкості світла.
      * Проекція візерунка на сцену.
      * Збір зображення з деформованим візерунком.
      * Аналіз деформацій для визначення глибини.

Переваги:

* + - * Висока точність: Тривимірні камери забезпечують точну інформацію про глибину.
      * Реальний час: Можливість отримання глибинної інформації в реальному часі.

Недоліки:

* + - * Висока вартість: Тривимірні камери коштують дорожче за звичайні камери.
      * Обмеження в умовах освітлення: Ефективність може знижуватися при яскравому сонячному світлі або на великих відстанях.

Застосування:

* + - * Автономні транспортні засоби для точного визначення відстані до об'єктів.
      * Медична діагностика для створення тривимірних зображень анатомічних структур.
      * Робототехніка для маніпулювання об'єктами.

### Лідар (LiDAR)

Лідар використовує лазерне сканування для вимірювання відстані до об'єктів. Він випромінює лазерні імпульси і вимірює час їх повернення, що дозволяє створити високоточну тривимірну карту оточення.

Процес:

* + - * Випромінення імпульсів: Лідар випромінює лазерні імпульси у різних напрямках.
      * Збір відбитих імпульсів: Сенсор приймає відбиті імпульси.
      * Обчислення відстані: Вимірювання часу повернення імпульсів дозволяє обчислити відстань до об'єктів.

Переваги:

* + - * Дуже висока точність: Лідар забезпечує надзвичайно точні вимірювання відстані.
      * Незалежність від освітлення: Працює однаково добре в умовах різного освітлення.

Недоліки:

* + - * Висока вартість: Лідари коштують значно дорожче за інші сенсори.
      * Обмеження в умовах погодних умов: Ефективність знижується під час дощу, туману або пилу.

Застосування:

* + - * Автономні транспортні засоби для високоточного картографування та виявлення об'єктів.
      * Геодезія та картографія для створення детальних тривимірних карт.
      * Робототехніка для навігації та маніпулювання об'єктами в складних середовищах.

Рисунок 1.3– приклад лазерного сканування

Кожен метод визначення відстані до об'єкту за допомогою камери має свої унікальні переваги та недоліки. Стереозір підходить для високоточної тривимірної реконструкції на малих відстанях, але потребує значних обчислювальних ресурсів. Монокулярна візія є найпростішою та найдешевшою, але має обмежену точність. Трівимірні камери забезпечують високу точність у реальному часі, але є дорогими та обмеженими в певних умовах освітлення. Лідар надає найвищу точність і незалежність від освітлення, але є найдорожчим рішенням. Вибір методу залежить від конкретних вимог завдання, таких як точність, вартість, обчислювальні витрати та умови експлуатації.

## 1.3 Вибір рішення для побудови автоматичної системи

**визначення координат та типу рухомого об’єкту з квадрокоптера**

Використання YOLOv8 для автоматичної системи визначення типу об’єкта має багато переваг, які роблять його найбільш оптимальним вибором для проекту [1].

Перш за все, його архітектура є однією з найефективніших для виявлення об’єктів у реальному часі. Його висока швидкість обробки дозволяє виявляти та класифікувати об’єкти швидше, ніж більшість інших систем, що робить його ідеальним для застосування в системам безпеки та відеоспостереження, а також для контролю та автоматизації. Крім того, YOLOv8 забезпечує високу точність виявлення об’єктів навіть за різних умов зйомки, таких як зміна освітлення, кутів огляду або різний фон. Його алгоритми виявлення об’єктів оптимізовані для роботи з різними типами об’єктів і умовами, забезпечуючи надійні результати навіть у складних ситуаціях. YOLOv8 також відрізняється тим, що є досить простим у використанні та інтеграції в існуючі системи, що дозволяє прискорити розробку та розгортання проекту. Загалом використання YOLOv8 забезпечить вашій системі високу швидкість, точність і надійність, необхідні для успішного та ефективного виявлення типу об’єкта.

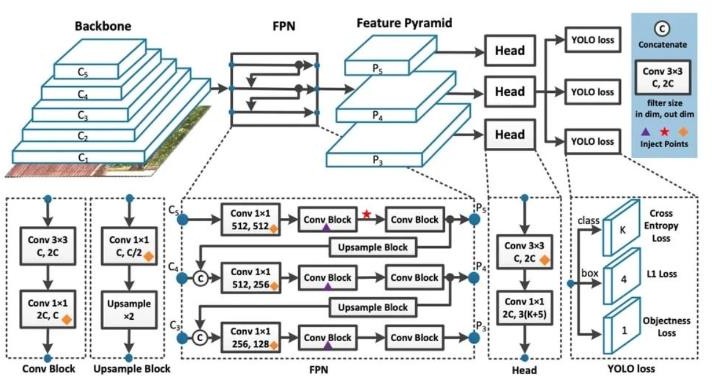


Рисунок 1.4 – архітектура YOLOv8

Використання методу монокулярної візії для визначення відстані до об’єкта також має переваги у застосуванні. Цей метод базується на оптичних принципах із використанням відомої фокусної відстані об’єктива камери. Однією з головних переваг цього методу є його простота і доступність. Він не потребує великої обчислювальної потужності чи складного апаратного забезпечення, що робить його привабливим вибором для різноманітних застосувань, включаючи мобільні системи та вбудовані пристрої. Крім того, метод фокусної відстані може бути ефективним у відносно простих ситуаціях, коли основна увага приділяється вимірюванню відстані до об’єктів на передньому плані. Однак слід зазначити, що точність цього методу може залежати від кількох факторів, таких як роздільна здатність камери, умови освітлення та інші параметри. Таким чином, його використання у більш складних ситуаціях може вимагати додаткового ретельного аналізу та калібрування.

Поєднання YOLOv8 і методу фокусної відстані в системі визначення типу об’єкта може стати чудовим рішенням з кількох причин. По-перше, YOLOv8 забезпечує швидке і точне виявлення об'єктів на зображеннях або відео,

дозволяючи системі швидко реагувати на зміни в навколишньому середовищі. Додавання методу фокусної відстані забезпечить визначення відстані до цих об’єктів за допомогою оптичних властивостей та характеристик камери. Це може бути особливо корисним у ситуаціях, коли вам потрібно знати не лише те, що знаходиться на зображенні, але й визначати наскільки близько вони відносно до системи. Такий підхід може підвищити рівень розуміння середовища і зробити систему більш адаптивною та інтелектуальною. Крім того, поєднання YOLOv8 і фокусного методу може забезпечити більш надійне та точне виявлення відстані та об’єктів, що є важливим для широкого спектру додатків, таких як автономні системи, засоби безпеки та медичні програми. Таким чином, ця комбінація може стати важливим кроком до розробки інтелектуальних та ефективних систем виявлення об’єктів.

## Висновки до розділу 1

У цьому розділі проведено детальний аналіз сучасних рішень для автоматичного визначення типу рухомого об'єкта та розрахунку відстані до нього за допомогою квадрокоптера. Особлива увага приділена методам комп'ютерного зору та різноманітним алгоритмам, які можуть бути використані для вирішення поставлених задач. Окрім цього, розглянуто існуючі підходи до визначення відстані до об'єктів на зображеннях.

Перший важливий висновок полягає у високій ефективності використання YOLOv8 для класифікації об'єктів. Цей метод забезпечує високу точність і швидкість виявлення, що робить його придатним для задач реального часу. Однією з переваг є простота інтеграції та універсальність застосування. Водночас високі вимоги до обчислювальних ресурсів можуть стати обмеженням у певних умовах.

Другий ключовий висновок стосується моделі Faster R-CNN, яка демонструє високу точність завдяки використанню регіональних пропозицій та

глибоких конволюційних мереж. Ця модель є складною в налаштуванні та вимагає значних обчислювальних ресурсів, але забезпечує відмінні результати у виявленні та класифікації об'єктів.

Окрім цього, в розділі було розглянуто методи визначення відстані до об'єктів на зображеннях. Використання стереозору та методів глибокого навчання дозволяє отримати точні вимірювання відстаней. Однак, для досягнення високої точності необхідно враховувати багато факторів, таких як якість зображень та характеристики камери.

На основі проведеного аналізу можна зробити висновок, що поєднання сучасних методів комп'ютерного зору та алгоритмів визначення відстані дозволяє значно підвищити ефективність та точність роботи квадрокоптерів у завданнях моніторингу та аналізу рухомих об'єктів. Проте, для досягнення оптимальних результатів важливо враховувати специфіку кожного застосування та вибирати найбільш відповідні інструменти і методи.

# АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ ТА РЕАЛІЗАЦІЇ АВТОМАТИЧНОЇ СИСТЕМИ

## Вибір платформи розробки та мови програмування

Для ефективної розробки програмного забезпечення визначення координат та типу рухомого об’єкта з квадрокоптера необхідно вибрати платформу розробки та мову програмування, які забезпечать не тільки високу продуктивність і надійність системи, але й будуть відповідати специфіці поставлених завдань. Особливу увагу варто приділити майбутній підтримці та розширенню системи : важливо обирати такі технології, що дозволять легко інтегрувати нові функції та адаптувати систему під зміну вимог.

Вибір платформи розробки та мови програмування також може вплинути на швидкість і ефективність роботи системи. Наприклад, використання потужної та оптимізованої мови програмування може допомогти досягти кращих результатів у роботі системи. Мови програмування низького рівня, такі як C++ та Rust, можуть забезпечити більшу швидкість завдяки контролю над системними ресурсами. Важливо враховувати наявність паралельних обчисленнь, бо вони можуть істотно підвищити продуктивність системи, дозволяючи розподіляти обчислювальні завдання між кількома процесорами або ядрами. Це особливо актуально для алгоритмів обробки зображень і машинного навчання, що є основою для визначення об'єктів у реальному часі.

Також, вибір платформи розробки та мови програмування впливає на зручність та час розробки системи. Використання фреймворку скорочує час, необхідний для розробки додатка за допомогою готових рішень, компонентів та інструментів, які прискорюють та оптимізують процес розробки. Крім того, платформа розробки та мова програмування можуть мати значний вплив на безпеку та стабільність системи, що особливо треба враховувати під час розробки програмного забезпечення для дронів, які можуть виконувати критично важливі завдання. Мови програмування з вбудованими засобами для аналізу коду та

перевірки безпеки, можуть забезпечити додаткові гарантії стійкості програми до помилок і зловмисних атак.

Отже, вибір платформи розробки та мови програмування є стратегічно важливим рішенням, що вимагає глибокого аналізу потенційних технологій, їх відповідності завданням проекту, можливостей до розширення та удосконалення, а також оцінки співвідношення продуктивності та зручності розробки.

### Мова програмування Python

Мова програмування Python також є однією з найпопулярніших мов для розробки програмного забезпечення, особливо в сфері комп'ютерного зору [2]. Його синтаксис простий і зрозумілий, що робить цю мову ідеальним вибором для багатьох розробників, навіть тих, хто не має великих знань у програмуванні. Однією з ключових переваг Python є його гнучкість і велика кількість доступних бібліотек, особливо в області комп’ютерного зору, таких як OpenCV, TensorFlow і PyTorch [3]. Ці бібліотеки роблять розробку систем розпізнавання зображень не тільки практичною, але й дозволяють досягти високої точності та продуктивності. Однією з найбільших переваг Python для систем комп’ютерного зору є простота використання та налагодження коду. Також Python має велику спільноту розробників, а це означає, що є багато документації, навчальних посібників і підтримки, щоб полегшити процес розробки. Хоча Python не такий швидкий, як мови нижчого рівня, такі як C++, він все ще достатньо ефективний для багатьох завдань комп’ютерного зору. Його універсальність і доступність використання роблять його привабливим вибором для швидкого створення прототипів і досліджень у цій галузі. Тому Python став популярним вибором для розробки систем комп’ютерного зору завдяки своїй простоті, ефективності та

різноманітності функцій, які він пропонує розробникам.

### Бібліотека Ultralytics

Ultralytics — це компанія та розробник програмного забезпечення, яка спеціалізується на розробці засобів комп’ютерного зору та обробки зображень. Одним з їхніх основних продуктів є бібліотека PyTorchYOLOv8, потужний інструмент для розпізнавання об’єктів у реальному часі. Бібліотека PyTorchYOLOv8 Ultralytics базується на структурі PyTorch і використовує оновлену версію архітектури YOLO (You Only Look Once) для виявлення зображень і відеооб’єктів. Він має кілька варіантів архітектури (S, M, L, XL), що дозволяє вибрати найбільш підходящий розмір моделі відповідно до ваших потреб у продуктивності та точності. PyTorchYOLOv8 також має велику кількість попередньо навчених моделей для різноманітних сценаріїв використання, що полегшує початок роботи з бібліотекою.

Однією з головних переваг PyTorchYOLOv8 є його висока швидкість і ефективність. Він дозволяє виявляти об’єкти в режимі реального часу, навіть на великих вибірках даних, що робить його ідеальним вибором для додатків, які вимагають швидке отримання результатів, наприклад, для безпілотних автомобілів, систем відеоспостереження тощо.

Крім того, Ultralytics надає детальну документацію та підтримку для своїх продуктів, що дозволяє розробникам швидко вивчати їх і ефективно використовувати у своїх проектах.

В результаті, бібліотека Ultralytics є потужним інструментом для розробки систем комп’ютерного зору та виявлення об’єктів, що допомагає спростити та прискорити розробку таких систем.

### Бібліотека OpenCV

OpenCV (Open Source Computer Vision Library) — це бібліотека комп’ютерного зору та обробки зображень з відкритим кодом, призначена для спрощення завдань у галузі комп’ютерного зору. Вона надає широкий спектр

функцій і алгоритмів для обробки зображень і відео, дозволяючи розробникам створювати різноманітні програми, такі як комп’ютерне бачення, машинне навчання, розпізнавання облич і відеоспостереження.

Однією з ключових особливостей OpenCV є підтримка різних мов програмування, включаючи Python, C++, Java тощо, що робить його дуже популярним серед розробників із різних галузей. Ця бібліотека має розширені функції для виявлення та класифікації об’єктів на зображеннях і відео за допомогою алгоритмів реального часу та машинного навчання.

OpenCV реалізує багато алгоритмів для маніпулювання зображеннями, включаючи фільтрацію, морфологічну маніпуляцію, виявлення країв, виділення ознак і відстеження об’єктів. Він також має функції для керування відео, включаючи обробку в реальному часі, відстеження руху, аналіз руху та виявлення об’єктів.

Завдяки своїй відкритості та потужним функціям OpenCV став стандартом у комп’ютерному зорі та обробці зображень і широко використовується в багатьох галузях, таких як робототехніка, медицина, безпека та автомобільна промисловість.

## Середовище розробки PyCharm

PyCharm — це високоефективне інтегроване середовище розробки, спеціально налаштоване для мови програмування Python. Однією з головних сильних сторін PyCharm є його адаптивне робоче середовище, яке дозволяє розробникам повністю налаштовувати IDE відповідно до своїх потреб. Воно містить широкий спектр функцій, таких як автозавершення, автоматичне виявлення помилок, вбудований налагоджувач і підтримка віртуальних середовищ. Крім того, PyCharm дозволяє інтегруватися з різними інструментами розробки, такими як система контролю версій Git, Docker та багато інших.

Однією з важливих особливостей PyCharm є його підтримка використання машинного навчання та моделей комп’ютерного зору. Завдяки інтеграції з такими бібліотеками, як TensorFlow, PyTorch і scikit-learn, розробники можуть легко створювати, навчати та налаштовувати свої моделі безпосередньо в середовищі PyCharm. Наявність інтегрованих інструментів для аналізу даних і візуального налагодження коду полегшує розробку та дослідження різних алгоритмів машинного навчання. За допомогою вбудованих шаблонів для визначення типів об’єктів, PyCharm служить чудовим інструментом для швидкої та ефективної роботи з цими шаблонами. Завдяки потужним функціям аналізу даних, налагодження візуального коду та підтримці різних фреймворків машинного навчання PyCharm дозволяє розробникам ефективно розгортати та налаштовувати готові до використання моделі, які використовуються для різних завдань комп’ютерного зору.

Таким чином, PyCharm стає ідеальним вибором для розробників, які працюють із готовими до використання моделями об’єктів, завдяки своїм зручним інструментам, широкому набору функцій і підтримці фреймворків машинного навчання.

## Використання Roboflow для підготовки датасетів

Робота з даними є важливою частиною будь-якого проекту машинного навчання. Roboflow надає повний набір інструментів, які дозволяють вам ефективно керувати даними без особливих зусиль. За допомогою цієї платформи ви можете зосередитися на аналізі даних і розробці моделі, зводячи до мінімуму час, витрачений на такі рутинні завдання, як анотації або підготовка даних.

Ще однією перевагою Roboflow є можливість легко розширити свій набір даних за допомогою доповнення.

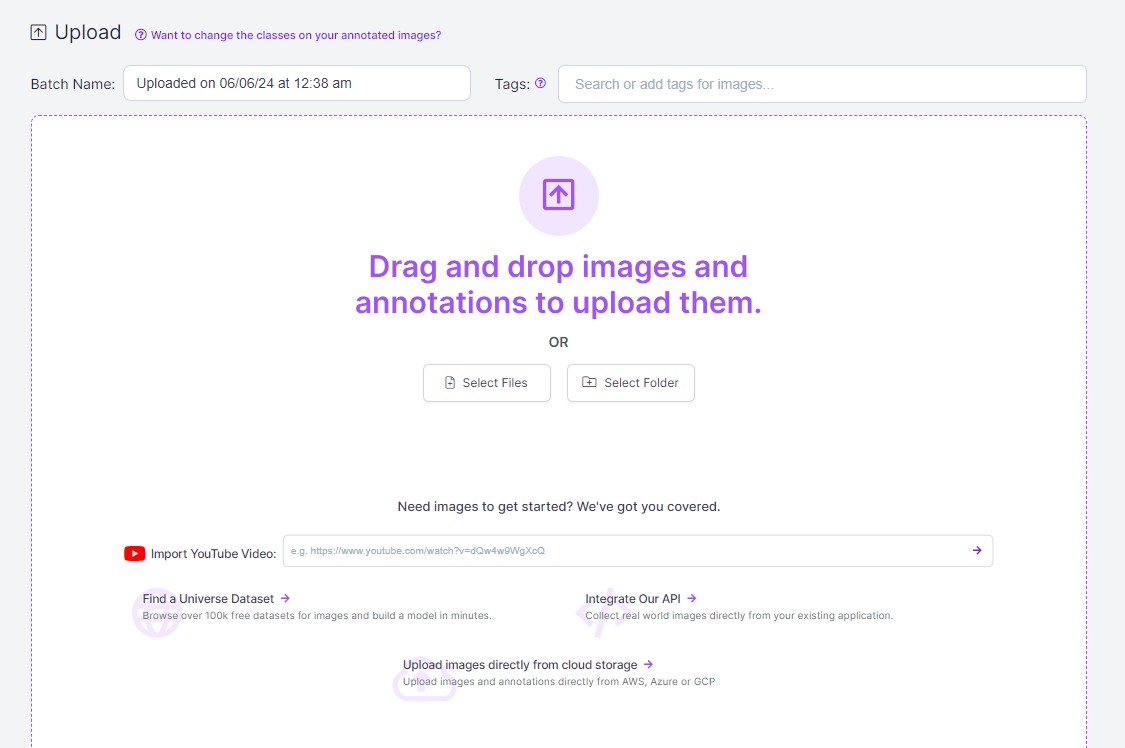


Рисунок 2.1 – сторінка доповнення датасету

Вбудовані інструменти для створення різних варіантів зображень дозволяють збільшити різноманітність набору даних, що допоможе моделі краще навчатися та працювати з реальними даними. Крім того, Roboflow пропонує інтеграцію з різними форматами даних, що дозволяє легко працювати з різними моделями та архітектурами. Це робить платформу популярним інструментом для роботи з даними у сфері комп’ютерного зору та машинного навчання в цілому.



Рисунок 2.2 – формати що підтримуються

Також перевагою Roboflow є його здатність працювати з різними завданнями комп’ютерного зору. Незалежно від того, чи потрібно вам виявити об’єкти на зображеннях, визначити ключові точки чи сегментувати регіони, ця платформа може ефективно задовольнити ваші потреби. Це забезпечує гнучкість під час роботи з різними типами завдань і даних.

Крім того, Roboflow пропонує велику кількість інструментів для керування вашими даними. Розробник можете легко впорядковувати свої набори даних, створювати різні версії та використовувати різні фільтри для аналізу та обробки даних. Це дозволяє підтримувати робочий простір чистим і організованим, сприяючи ефективній роботі та підтримці співпраці команди.

Загалом, Roboflow розроблений, щоб зробити підготовку даних для завдань комп’ютерного зору максимально ефективною та зручною. Його набір інструментів, гнучкість і зручний інтерфейс роблять його потужним інструментом для роботи з даними в області машинного навчання.

## Використання Google Colab для тренування моделі

Використання Google Colab для тренування датасету YOLOv8, створеного у Roboflow, має кілька важливих переваг. Google Colab надає безкоштовний доступ до обчислювальних ресурсів, зокрема графічних прискорювачів GPU(Graphic Processing Unit) та TPU(Tensor Processing Unit). Це дозволяє використовувати потужний інструмент для тренування моделі без необхідності великих витрат на фізичне обладнання або хмарні сервіси.

Використання GPU або TPU у Google Colab дозволяє значно зменшити час тренування моделі, прискорюючи обробку даних. Це особливо важливо для великих датасетів або складних моделей, таких як YOLOv8.

Google Colab має зручний інтерфейс та інтегровані інструменти, що спрощують написання, виконання та спільну роботу з кодом. Це надає можливість ділитися ноутбуками з колегами або працювати над проектами в режимі реального часу.

В цілому, Google Colab є потужним та важливим інструментом для тренування моделей машинного навчання, і його використання спільно з датасетами, створеними у Roboflow, може значно спростити та прискорити процес розробки та налагодження моделей YOLOv8.

## Висновки до розділу 2

У цьому розділі було проведено глибокий аналіз інструментів, що використовуються для розробки та реалізації автоматичної системи визначення координат і типу рухомого об’єкта з квадрокоптера. На основі розглянутих аспектів вибору платформи, мови програмування, бібліотек та середовищ розробки, можна зробити кілька важливих висновків.

По-перше, вибір платформи та мови програмування є критично важливим для ефективної розробки системи. Розглядаючи продуктивність, надійність, підтримку паралельних обчислень, а також можливість легкого розширення та адаптації системи, слід зазначити, що мови низького рівня, такі як C++ та Rust, забезпечують високу швидкість і контроль над системними ресурсами, що особливо важливо для обробки зображень у реальному часі. Водночас, Python завдяки своїй простоті, гнучкості та багатству бібліотек є оптимальним вибором для швидкої розробки та тестування систем комп'ютерного зору.

По-друге, мова програмування Python виявилася однією з найпопулярніших для розробки програмного забезпечення у сфері комп'ютерного зору. Його простий синтаксис, велика кількість доступних бібліотек (таких як OpenCV, TensorFlow, PyTorch) та активна спільнота розробників роблять його привабливим вибором для реалізації проектів різної складності.

По-третє, бібліотека PyTorchYOLOv8 від Ultralytics є потужним інструментом для розпізнавання об'єктів у реальному часі. Її висока швидкість, ефективність, наявність попередньо навчених моделей та детальна документація роблять її ідеальним вибором для задач, що вимагають швидкого отримання результатів, таких як безпілотні автомобілі та системи відеоспостереження.

Крім того, бібліотека OpenCV є стандартом у галузі комп'ютерного зору та обробки зображень завдяки своїй відкритості, широкому спектру функцій та підтримці різних мов програмування. Вона забезпечує можливість вирішення

різноманітних задач, від виявлення та класифікації об'єктів до відстеження руху та аналізу відео в реальному часі.

Середовище розробки PyCharm також має високу ефективність, будучи спеціально налаштованим для Python. PyCharm підтримує роботу з бібліотеками машинного навчання, такими як TensorFlow, PyTorch та scikit-learn, що полегшує створення, навчання та налагодження моделей комп'ютерного зору. Цей інструмент забезпечує зручність та ефективність розробки завдяки своїм потужним функціям та інтеграції з іншими розробницькими інструментами.

Наступним важливим інструментом є Roboflow, який надає повний набір інструментів для ефективної роботи з даними, зокрема для анотації, доповнення та підготовки наборів даних. Використання Roboflow дозволяє значно підвищити якість та різноманітність набору даних, що є критично важливим для навчання моделей комп'ютерного зору.

Нарешті, Google Colab є потужним та важливим інструментом для тренування моделей машинного навчання. Використання Colab разом з датасетами, створеними у Roboflow, забезпечує безкоштовний доступ до потужних обчислювальних ресурсів, таких як GPU та TPU, що дозволяє значно прискорити процес тренування та розробки моделей YOLOv8. Це робить Google Colab цінним інструментом для дослідників та розробників, які працюють над складними задачами комп'ютерного зору.

Таким чином, проведений аналіз демонструє, що правильний вибір інструментів розробки та платформ може значно вплинути на ефективність, швидкість та успішність створення автоматичної системи визначення координат і типу рухомого об’єкта з квадрокоптера.

# ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## Підготовка набору даних

Підготовка датасету є ключовим етапом у розробці системи комп'ютерного зору або машинного навчання. Перш за все, якість даних безпосередньо впливає на якість моделі. Якщо датасет містить недостатньо репрезентативних прикладів або недоліки у якості анотації, це може призвести до погіршення результатів моделі. Репрезентативність даних також важлива, оскільки набір даних повинен відображати різноманітність об'єктів у різних умовах зйомки.

Тому для наповення датасету були здебільшого використані відео атаки українських дронів. Такі відео зняті у бойових умовах, з різними погодними умовами, різними кутами камери та різними дистанціями до об’єктів. Також були використані зображення зі супутників та розвідувальних дронів, для більшої кількості різноманіття об’єктів.

Правильна анотація даних є ще однією ключовою складовою підготовки датасету. Вона повинна бути точною та повністю відображати об'єкти на зображеннях. Недоліки у анотації можуть призвести до навчання моделі на неправильних даних та погіршення результатів її роботи. Анотація зображень була зроблена за допомогою Roboflow.

На платформі Roboflow анотування зображень має декілька переваг, які роблять її вигідним вибором для підготовки датасетів. По-перше, Roboflow пропонує широкий набір інструментів для анотування, включаючи ручне анотування, а також автоматичні та напівавтоматичні методи. Це дозволяє анотувати зображення швидко та ефективно, незалежно від складності завдання.

По-друге, Roboflow забезпечує інтеграцію з різними форматами анотацій, що дозволяє зберігати анотовані дані у форматі, придатному для використання з різними моделями машинного навчання. Це робить підготовку даних більш універсальною та гнучкою. На рисунку 3.1 можна побачити інтерфейс для анотування.

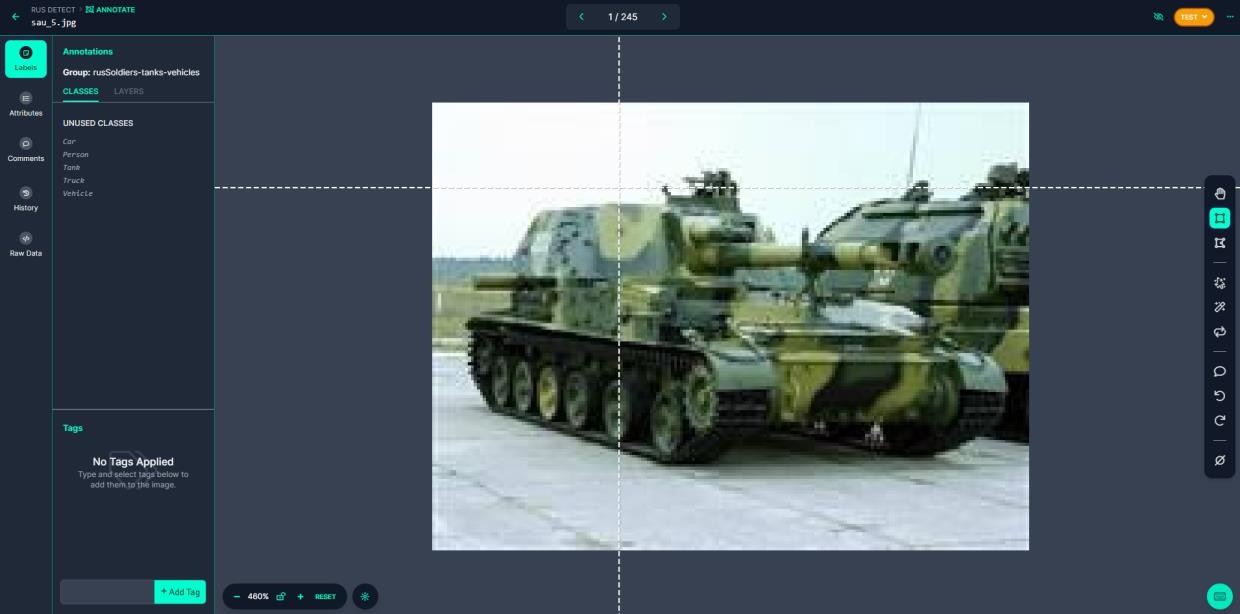


Рисунок 3.1 – інтерфейс для анотування у Roboflow

Крім того, платформа Roboflow пропонує різноманітні інструменти для якісного контролю якості анотацій, що дозволяє впевнитися в точності та повноті анотацій на кожному зображенні. Це особливо важливо при роботі з великими обсягами даних, де навіть невеликі помилки можуть вплинути на якість навчання моделі. Таким чином, при оновленні та збільшенні датасету проблем не буде.

Для моделі було анотовано 5 класів, кожен з яких може представляти певний тип об'єкта, який може бути корисним для військового застосування.

Клас “Person” (особа): цей клас може допомогти в розпізнаванні людей на землі, що може бути важливою інформацією для військового збору розвідки або для виявлення загрози.

Клас “Tank” (танк): розпізнавання танків може бути корисним для визначення місць розташування ворожих танків та для планування військових операцій.

Клас “Vehicle” (транспортний засіб): Виявлення різних військових штурмових засобів, таких як БТР, може допомогти військовим оцінити рух ворожих сил або пересування ресурсів.

Клас “Truck” (вантажівка): Окрім звичайних транспортних засобів, розпізнавання вантажівок може вказувати на перевезення важливого обладнання або матеріалів.

Клас “Car” (автомобіль): Виявлення автомобілів може допомогти військовим контролювати рух на дорогах та виявляти потенційні загрози.

На рисунку 3.2 можна побачити усі класи моделі.



Рисунок 3.2 – класи моделі

Отже, анотування цих класів надає квадрокоптеру важливу інформацію для виконання військових завдань, таких як розвідка, моніторинг та ідентифікація загроз.

Розбиття датасету на набори для тренування, валідації та тестування є важливим етапом в процесі розробки моделі машинного навчання.

Набір тренування (Train set): Це частина датасету, яка використовується для навчання моделі. Тренувальний набір містить дані, на основі яких модель навчається впізнавати патерни та здійснювати прогнози.

Набір валідації (Validation set): Ця частина датасету використовується для налаштування гіперпараметрів моделі та оцінки її продуктивності під час

навчання. Валідаційний набір допомагає виявити перенавчання та підбор оптимальних значень гіперпараметрів.

Набір тестування (Test set): Це незалежна частина датасету, яка використовується для оцінки фінальної продуктивності моделі після її навчання та налаштування. Тестовий набір дозволяє оцінити загальну здатність моделі генералізувати на нових даних, які вона раніше не бачила.

Розбиття зібраного датасету – 291 зображення для тренування, 53 зображення для валідації та 89 зображень для тестування.

На рисунку 3.3 можна побачити як розбито датасет.



Рисунок 3.3 – Розбиття датасету

Для зменшення часу тренування та підвищення працездатності усі зображення були нормалізовані до масштабу 640х640. Нормалізацію можна побачити на рисунку 3.4



Рисунок 3.4 – нормалізація масштабу зображень

Для підвищення варіативності зображень 15 відсотків зображень були скопійовані та переведені у відтінки сірого, після чого були додані до датасету.



Рисунок 3.5 – аугментація датасету

На рисунку 3.5 можна побачити використану аугментацію.

* 1. **Тренування, валідація та тестування моделі у Google Colab** Тренування моделі YOLOv8 з датасету, зібраного у Roboflow, у середовищі Google Colab, може бути ефективним та зручним завдяки доступності

обчислювальних ресурсів та інтегрованим інструментам.

Перше, що потрібно зробити після входу в Google Colab, – налаштувати середовище для роботи. Один з перших кроків - перевірити, чи використовується GPU. Це можна зробити, вибравши "Runtime" в головному меню, а потім "Change runtime type". У вікні, що відкриється, вибираємо "GPU" в розкривному меню "Hardware accelerator". Використання GPU значно прискорить тренування моделі YOLOv8.

Після налаштування середовища Google Colab для використання GPU імпортуємо необхідні бібліотеки для розробки та тренування моделі. Зазвичай це включає в себе бібліотеки машинного навчання, такі як PyTorch, TensorFlow, Ultralytics, а також інші допоміжні бібліотеки для роботи з даними та обробки зображень. Треба переконатися, що всі необхідні бібліотеки встановлені та готові до використання.

Після налаштування середовища завантажуємо датасет, який був підготовлений у Roboflow. Використовуємо API Roboflow або просто завантажуємо датасет безпосередньо в Google Colab. Після завантаження розпаковуємо датасет та підготовлюємо його для подальшої обробки та використання у тренувальному процесі. Треба впевнитися, що дані завантажені правильно та готові до використання у проекті.

Далі можна переходити до тренування. Для цього вказуємо потрібні для тренування моделі параметри: архітектуру моделі, кількість епох та розширення зображень. Модель була тренована з такими параметрами: модель – yolov8,

кількість епох – 25, розширення – 640х640. Під час тренування можна моніторити метрики навчання та спостерігати за візуалізацією результатів, такими як графіки функції втрат та точності. Цей етап дозволяє моделі навчатися розпізнавати патерни в даних та покращувати свою продуктивність з кожною епохою.

Оцінка продуктивності моделі відбувається після завершення тренування, коли використовується валідаційний набір даних для оцінки її точності та відновлення. Аналізуючи ці метрики, можна визначити, наскільки добре модель працює на нових даних та внести корективи в процес тренування для покращення результатів. Результати аналізу можна побачити на рисунку 3.6.

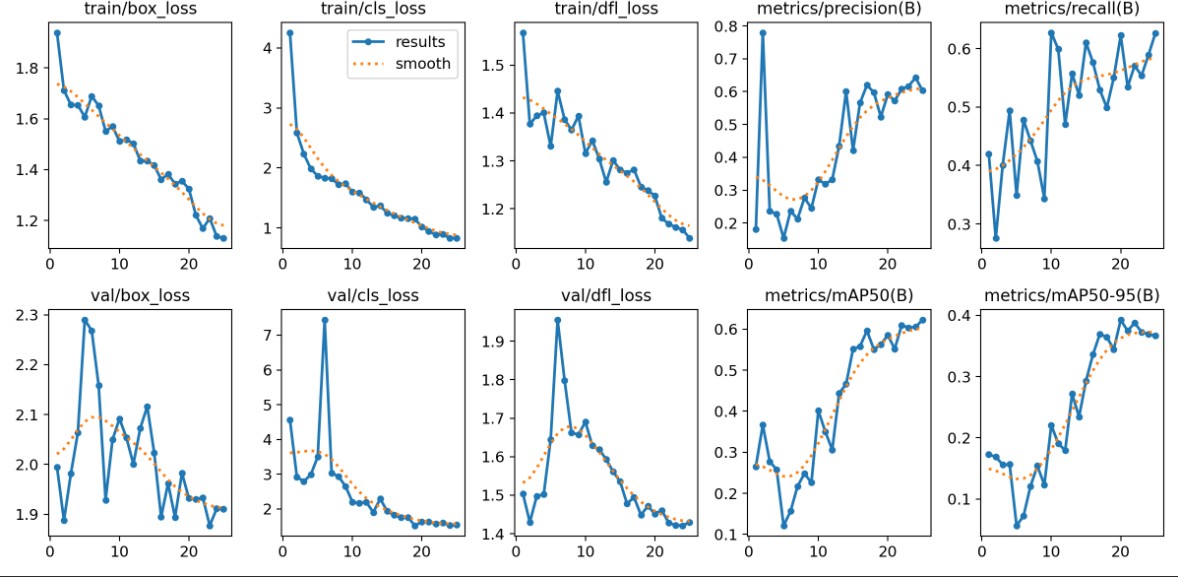


Рисунок 3.6 – оцінка продуктивності

Після успішного тренування та тестування потрібно зберігти навчену модель, щоб використовувати її у подальших застосунках або для розгортання на інших пристроях. Збереження моделі дозволить з легкістю використовувати її у різних проектах або розподілити її з іншими дослідниками та розробниками.



Рисунок 3.7 – результати тестування

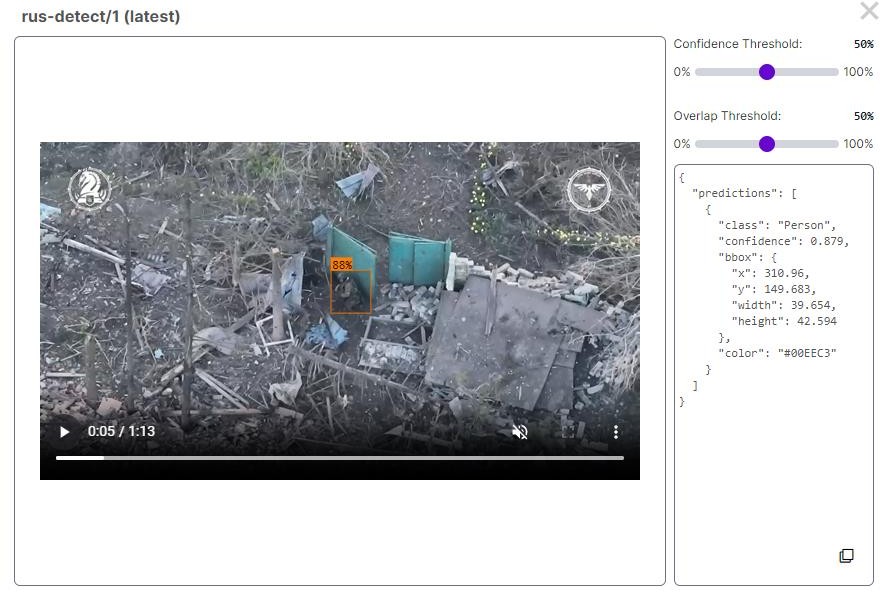


Рисунок 3.8 – результати роботи моделі

Результати моделі та її роботи можно побачити на рисунку 3.7 та рисунку

3.8 відповідно.

## Розробка алгоритму визначення дистанції методом подібності трикутників

Вимірювання відстані між камерою та заданим об'єктом обчислюється за принципами роботи камери. Обчислення цієї відстані є відносно простим, коли відомі реальні розміри об'єкта, оскільки принципи роботи камери легко зрозумілі.

Запропонований метод вимірювання відстані до об'єкта є одним із найефективніших методів в комп'ютерному зорі, який представлений в літературі. На рисунку 3.3 показане одне конкретне зображення відстані між об'єктом і маркером, щоб краще зрозуміти метод подібності трикутників за допомогою наступних різних параметрів:

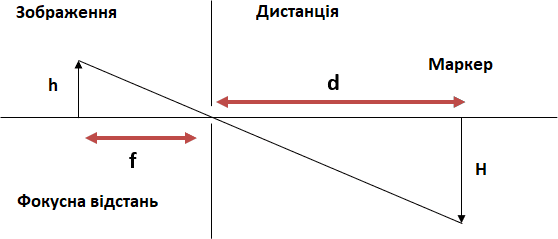


Рисунок 3.9 – візуалізація визначення дистанції методом подібності трикутників за допомогою фокусної відстані

Щоб визначити відстань від камери до відомого об’єкта або маркера, ми будемо використовувати подібність трикутника. Як ми бачимо на рисунку 3.9, ми маємо:

* + - f: фокусна відстань;
    - h: висота маркера в площині зображення;
    - H: висота маркера в площині об’єкта;
    - d: невідомий параметр, відстань від камери до об’єкта.

Вважається, що камера створює взаємозв'язок один-до-одного між об'єктом і зображенням. Тоді ми можемо вивести зв’язок між відомими параметрами:

𝑓/𝑑 = 1/𝐻 (3.1)

𝑓 = 𝑑 1

𝐻

(3.2)

Тепер ми можемо визначити значення d наступним чином:

𝑑 = 𝑓 𝐻

ℎ

(3.3)

Припустімо, ми вимірюємо відстань від камери до об’єкта у відео. Ми знаємо, що фокусна відстань камери f становить 30 мм, а об’єкт, який ми хочемо виміряти, має відому висоту 1 м. Ми також бачимо на відео, що об’єкт має круглий маркер радіусом 10 см.

Використовуючи формулу (3.3) вираховуємо відстань:

1𝑚

𝑑 = 30𝑚𝑚

10𝑐𝑚

Таким чином, відстань від камери до об'єкта становить приблизно 30 см або

0,3 м.



Рисунок 3.10 – результат роботи алгоритму визначення відстані до об'єкту на моделі визначення лиць

Результати роботи алгоритму можна побачити на рисунку 3.10.

## Інтеграція моделі визначення типу рухомого об’єкту та алгоритму визначення дистанції

Для використання алгоритму визначення дистанції на моделі визначення типу рухомого об’єкту треба використати бібліотеку Ultralytics для здобуття усіх необхідних даних. Після тренування моделі у Google Colab, потрібно завантажити ваги з найкращими значеннями для визначення типу об’єкту. Файл з найкращими вагами створюється бібліотекою Ultralytics та завжди називається “best.pt”.

Для написання програми використовується IDE PyCharm та мова програмування Python. Після створення проєкту, додаємо до його структури файл “best.pt” для подальшого використання. У створеному PyCharm файлі “main.py” імпортуємо бібліотеки Ultralytics для роботи моделі та cv2 (OpenCV) для візуалізації результатів.

Спочатку загружаємо модель за допомогою вагів та робимо інференцію за допомогою навченої моделі YOLOv8 на нових зображеннях, знятих камерою. Модель повинна точно виявляти відомий об'єкт та надавати координати обмежувальних рамок. Після виявлення об'єкта на зображенні витягуємо координати обмежувальних рамок та обчислюємо піксельні розміри об'єкта на зображенні. Важливо калібрувати камеру для точного відображення піксельних розмірів на реальні відстані. Це включає визначення параметрів камери, таких як фокусна відстань. Визначаємо параметри відомих розмірів об’єктів для кожного класу. Використовуємо обчислені піксельні розміри об'єкта та відомі реальні розміри для обчислення відстані від камери до об'єкта. Застосовуємо формулу подібності трикутників для перетворення піксельних розмірів на фізичні відстані.

Отримані результати роботи можна вивести на екран за допомогою бібліотеки OpenCV.



Рисунок 3.11 – результат інтеграції моделі визначення типу рухомого об’єкту та алгоритму визначення дистанції

Результат програми можна побачити на рисунку 3.11.

## Висновок до розділу 3

У цьому розділі було детально описано процес програмної реалізації системи автоматичного визначення координат і типу рухомого об'єкта з квадрокоптера. Початковий етап підготовки набору даних продемонстрував важливість якісних та репрезентативних зображень для тренування моделі. Використання відео з бойових дій, а також супутникових зображень та зображень з розвідувальних дронів забезпечило різноманітність даних, що є ключовим для досягнення високої точності моделі. Платформа Roboflow виявилася ефективним

інструментом для анотації зображень, надаючи зручні та різноманітні методи для швидкої та точної підготовки датасету.Процес тренування, валідації та тестування моделі був проведений у середовищі Google Colab, що дозволило ефективно використовувати доступні обчислювальні ресурси. Використання GPU значно прискорило тренування моделі YOLOv8, а інтеграція з Roboflow спростила завантаження та підготовку даних. Завдяки моніторингу метрик навчання та точності моделі вдалося оптимізувати її продуктивність та отримати високі результати.Розробка алгоритму визначення дистанції методом подібності трикутників стала важливим етапом для досягнення точних вимірювань відстаней до об'єктів. Використання фокусної відстані камери та відомих розмірів об'єктів дозволило створити ефективний метод обчислення, що може бути застосований у різних умовах.Інтеграція моделі визначення типу рухомого об'єкта та алгоритму визначення дистанції дозволила створити повноцінну систему для використання квадрокоптера у військових цілях. Завантаження найкращих вагів моделі, виконання інференції на нових зображеннях та обчислення відстані до об'єктів забезпечили точність та надійність системи. Використання PyCharm та мови програмування Python, а також бібліотек Ultralytics та OpenCV, значно спростило процес розробки та візуалізації результатів.Таким чином, проведені дослідження та розробки в цьому розділі доводять ефективність запропонованих методів та інструментів для створення автоматичної системи визначення координат і типу рухомого об'єкта з квадрокоптера. Отримані результати показують високу точність та надійність системи, що робить її цінним інструментом для військових задач, таких як розвідка, моніторинг та ідентифікація загроз.

# РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ

## Системні вимоги

Вибір одноплатних комп'ютерів (SBC) для використання з моделлю YOLOv8 та іншими обчислювальними завданнями може залежати від кількох факторів, таких як потужність обчислювальних ресурсів, доступність роз'ємів для зовнішніх пристроїв, ціна та інші. Ось кілька SBC, які можна врахувати:

Nvidia Jetson Nano: Це досить потужний SBC, який має вбудований GPU із підтримкою CUDA, що робить його ідеальним для роботи з моделями машинного навчання, такими як YOLOv8.Raspberry Pi 4:

Raspberry Pi є популярним вибором для багатьох проектів завдяки своїй доступності та розширеній підтримці. Хоча Raspberry Pi 4 не так потужний, як Nvidia Jetson Nano, він все ще може бути використаний для простих обчислювальних завдань.

Coral Dev Board: Це ще один SBC від Google, який має вбудований аппаратний прискорювач для машинного навчання Edge TPU. Він може забезпечити велику швидкодію для обчислювальних завдань, пов'язаних з машинним навчанням.

Odroid XU4: Цей SBC має відмінні обчислювальні можливості, що робить його привабливим вибором для використання з моделями машинного навчання. При виборі SBC для використання з моделлю YOLOv8 важливо врахувати ваші конкретні потреби та обмеження, такі як бюджет, потужність обчислювальних ресурсів та можливості розширення.

## Розробка нових версій датасету

Створення та покращення датасетів для проектів машинного навчання є важливою частиною процесу розробки моделей. Roboflow є потужним інструментом, який допомагає спростити цей процес та сприяє спільній роботі

над даними між командами. Початок роботи з Roboflow зазвичай включає створення облікового запису та створення нового проекту датасету. Після цього можна завантажити вже існуючі дані або почати з нуля, додаючи нові зображення. Наступним кроком є анотування зображень для позначення об'єктів на них. Roboflow надає зручний інтерфейс для цього, де можна визначити класи об'єктів та виконати маркування. Цей процес може бути ітеративним, оскільки можуть з'являтися нові дані або класи, а також потреби у виправленні помилок.

Одним із ключових переваг Roboflow є можливість спільної роботи над датасетом. Ви можете дозволити іншим користувачам доступ до свого проекту, щоб вони могли вносити зміни та доповнення. Це дозволяє ефективною співпрацювати з колегами або замовниками, а також розподілити навантаження на анотування та відгуки.

Крім того, Roboflow надає інструменти для аналізу даних, виявлення помилок та покращення анотацій. Ви можете переглядати статистику датасету, відстежувати рівень точності анотацій та забезпечувати якість даних для навчання моделей.

В цілому, Roboflow сприяє ефективній розробці та управлінню даними для проектів машинного навчання, забезпечуючи потужні інструменти для створення, покращення та спільної роботи над датасетами.

## Висновок до розділу 4

Розділ "Робота користувача з системою" висвітлює важливі аспекти вибору обладнання та управління даними для забезпечення ефективного функціонування системи визначення координат і типу рухомого об'єкта з квадрокоптера. Важливою складовою є вибір одноплатних комп'ютерів (SBC), таких як Nvidia Jetson Nano, Raspberry Pi 4, Coral Dev Board та Odroid XU4, які забезпечують необхідну обчислювальну потужність для роботи з моделями машинного навчання, зокрема YOLOv8. Кожен із цих комп'ютерів має свої переваги та

обмеження, що дозволяє обирати оптимальний варіант залежно від конкретних потреб та бюджету проекту.Розробка нових версій датасету є критичним етапом для підвищення точності та продуктивності моделі. Використання платформи Roboflow значно спрощує цей процес завдяки зручним інструментам для створення, анотації та управління датасетами. Робота з Roboflow дозволяє ефективно розподілити завдання з анотування та покращення якості даних, забезпечуючи можливість спільної роботи між членами команди та іншими зацікавленими сторонами. Це особливо важливо у великих проектах, де точність анотацій та різноманітність даних є ключовими для успіху.Загалом, забезпечення належних системних вимог та розробка якісних датасетів є основою для успішного впровадження системи автоматичного визначення координат і типу рухомого об'єкта. Використання сучасних інструментів та платформ, таких як Roboflow, сприяє не тільки підвищенню ефективності роботи, але й покращенню результатів моделі, що в кінцевому підсумку підвищує точність і надійність системи в реальних умовах експлуатації.

# ВИСНОВКИ

В даній роботі була поставлена задача побудови автоматичної системи визначення координат та типу рухомого об’єкту з квадрокоптера. Під час виконання роботи було проведено детальний аналіз існуючих рішень і вибрано оптимальне для реалізації системи.

Процес розробки системи включав вибір платформи розробки, мови програмування та інструментів для реалізації програмного забезпечення. Крім того, було вивчено та використано необхідні технології для забезпечення швидкості і точності визначення типу об’єкту та відстані до нього.

Автоматична система була розроблена з метою забезпечення швидкого та ефективного визначання рухомих об’єктів у реальному часі, використані оптимізовані алгоритми, що дозволяють працювати на пристроях з обмеженими ресурсами, таких як одноплатні комп'ютери або мобільні пристрої. Для визначення відстані до об’єкта було використано алгоритм подібності трикутників, Основна ідея полягає в тому, що за допомогою знання фактичного розміру об'єкта і його відображення на зображенні можна визначити відстань до об'єкта за допомогою співвідношення подібних трикутників.

Основною перевагою цього методу є те, що він не вимагає складних обчислень або великої обчислювальної потужності. Він ґрунтується на простих принципах геометрії та може бути застосований з відносно простими алгоритмами, які не потребують значних обчислювальних ресурсів.

Наступні кроки у розвитку цієї системи можуть включати вдосконалення алгоритмів обробки відео та покращення точності визначення координат об’єкту, оптимізація гіпермараметрів моделі, квантизація моделі.

Отже, розроблена система є високопродуктивною та ефективною. Вона може бути використана у військовій та пошукових сферах. Розроблена система відкриває нові можливості для ефективної відстеження об’єктів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Talib M. YOLOv8-CAB: Improved YOLOv8 for Real-time object detection / M. Talib, A. Al-Noori, J. Suad., 2024. – (Karbala International Journal of Modern Science).
2. Guido S. Introduction to Machine Learning with Python / S. Guido, A. C. Müller., 2016. – 398 с.
3. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2nd Edition

/ Aurélien Géron.. – 856 с.

1. PyCharm Documentation. URL: https://[www.jetbrains.com/help/pycharm/getting-started.html](http://www.jetbrains.com/help/pycharm/getting-started.html)
2. Python Documentation. URL: https://docs.python.org/3/
3. Ultralytics Documentation. URL: https://docs.ultralytics.com/
4. OpenCV Documentation. URL: https://docs.opencv.org/4.x/d1/dfb/intro.html
5. PyTorch Faster R-CNN Documentation URL: https://pytorch.org/vision/main/models/faster\_rcnn.html
6. PyTorch Mask R-CNN Documentation. URL: https://pytorch.org/vision/main/models/mask\_rcnn.html
7. PyTorch RetinaNet Documentation. URL: https://pytorch.org/vision/0.16/models/retinanet.html
8. Roboflow. URL: https://roboflow.com/
9. D.Reis. Real-Time Flying Object Detection with YOLOv8 / D.Reis, J. Kupec, J. Hong, A. Daoudi., 2023.
10. N.Ghedia. Moving Objects Detection Using Machine Learning (SpringerBriefs in Electrical and Computer Engineering) / N.Ghedia, C. Vithalani, A. Kothari, R. Thank., 2022. – 92 с.

# ДОДАТОК А

Автоматична система визначення координат та типу рухомого об’єкту з квадрокоптеру

Лістинг програмних модулів

Аркушів 9

Київ 2024

Rus-detect.ipynb

!nvidia-smi

import os

HOME = os.getcwd() print(HOME)

!pip install ultralytics==8.0.196

from IPython import display display.clear\_output()

import ultralytics ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image

!mkdir {HOME}/datasets

%cd {HOME}/datasets

!pip install roboflow --quiet from roboflow import Roboflow

rf = Roboflow(api\_key="API\_KEY")

project = rf.workspace("krasilnik-ivan").project("rus-detect") version = project.version(1)

dataset = version.download("yolov8")

%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=25 imgsz=640 plots=True

!ls {HOME}/runs/detect/train/

%cd {HOME}

Image(filename=f'{HOME}/runs/detect/train/confusion\_matrix.png', width=600)

%cd {HOME}

Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)

%cd {HOME}

Image(filename=f'{HOME}/runs/detect/train/val\_batch0\_pred.jpg', width=600)

%cd {HOME}

!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml

%cd {HOME}

!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True

import glob

from IPython.display import Image, display

for image\_path in glob.glob(f'{HOME}/runs/detect/predict/\*.jpg')[:3]: display(Image(filename=image\_path, width=600))

print("\n")

Main.py

from ultralytics import YOLO import cv2

focal\_length = 8

known\_height = 0.22

class ObjectDetection:

def init (self, capture\_index=0): self.capture\_index = capture\_index

self.device = 'cpu'

print("Using device: ", self.device)

self.model = self.load\_model()

def load\_model(self):

model = YOLO('faces.pt') model.fuse()

return model

def plot\_bboxes(self, results, frame): xyxys = []

confidences = [] class\_ids = [] distances = []

for result in results:

boxes = result.boxes.cpu().numpy()

xyxys.append(boxes.xyxy) confidences.append(boxes.conf) class\_ids.append(boxes.cls)

if len(result.boxes) == 0:

print("No bounding boxes detected.") continue

cord = result.boxes.xywh.tolist()[0] h = cord[3]

image\_object\_height = h

print("Image Object Height:", image\_object\_height)

distance = (focal\_length \* known\_height) / (image\_object\_height / 100) distances.append(distance)

for (xyxy, class\_id, confidence, distance) in zip(xyxys[0], class\_ids[0], confidences[0], distances):

x1, y1, x2, y2 = map(int, xyxy)

class\_name = self.model.names[int(class\_id)]

label = f"{class\_name} {confidence:.2f} {distance:.2f}m" cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 2)

return frame, xyxys, confidences, class\_ids, distances

def call (self):

cap = cv2.VideoCapture(self.capture\_index)

assert cap.isOpened(), "Error: Cannot open camera. Please check if camera is connected."

while cap.isOpened(): ret, frame = cap.read() if not ret:

break

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) results = self.model(frame)

if len(results) == 0:

cv2.imshow('Object Detection', cv2.cvtColor(frame, cv2.COLOR\_RGB2BGR))

if cv2.waitKey(1) & 0xFF == ord('q'): break

continue

plot\_img, xyxys, class\_ids, confidences, distances = self.plot\_bboxes(results,

frame)

cv2.imshow('Object Detection', cv2.cvtColor(plot\_img, cv2.COLOR\_RGB2BGR))

if cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release() cv2.destroyAllWindows()

if name == " main ": object\_detection = ObjectDetection() object\_detection()

Main2/py

import torch.cuda

from ultralytics import YOLO import cv2

import time focal\_length = 20

class ObjectDetection:

def init (self, video\_path, class\_heights=None): self.video\_path = video\_path

self.class\_heights = class\_heights if class\_heights is not None else {}

# Check if CUDA is available and set the device accordingly self.device = 'cuda' if torch.cuda.is\_available() else 'cpu' print("Using device: ", self.device)

# Load the YOLO model self.model = self.load\_model()

def load\_model(self): model = YOLO('best.pt') model.fuse()

return model

def plot\_bboxes(self, results, frame): xyxys = []

confidences = [] class\_ids = [] distances = []

for result in results:

boxes = result.boxes.cpu().numpy()

xyxys.append(boxes.xyxy) confidences.append(boxes.conf) class\_ids.append(boxes.cls)

print("Detected classes:", [self.model.names[int(cls)] for cls in class\_ids[0]]) print("Confidences:", confidences)

for (result, class\_id) in zip(results, class\_ids[0]): cord = result.boxes.xywh.tolist()[0]

y, h = cord[1], cord[3] image\_object\_height = h

class\_name = self.model.names[int(class\_id)].lower() # Convert class name to lowercase

known\_height = self.class\_heights.get(class\_name, 0.0) if known\_height == 0.0:

print(f"Unknown height for class '{class\_name}'. Cannot calculate distance.") continue

distance = (focal\_length \* known\_height) / image\_object\_height \* 100 distances.append(distance)

print("Distances:", distances)

for (xyxy, class\_id, confidence, distance) in zip(xyxys[0], class\_ids[0], confidences[0], distances):

x1, y1, x2, y2 = map(int, xyxy)

class\_name = self.model.names[int(class\_id)]

label = f"{class\_name} {confidence:.2f} {distance:.2f}m" cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 2)

return frame, xyxys, confidences, class\_ids, distances def call (self):

cap = cv2.VideoCapture(self.video\_path)

assert cap.isOpened(), "Error: Cannot open video file. Please check if the file exists."

frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)) frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

while cap.isOpened(): start\_time = time.time()

ret, frame = cap.read() if not ret:

break

frame = cv2.resize(frame, (640, 640)) original\_frame = frame.copy()

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) results = self.model(frame)

if len(results) == 0:

cv2.imshow('Object Detection', cv2.cvtColor(original\_frame, cv2.COLOR\_BGR2RGB))

if cv2.waitKey(1) & 0xFF == ord('q'): break

continue

plot\_img, \_, \_, \_, \_ = self.plot\_bboxes(results, frame) cv2.imshow('Object Detection', cv2.cvtColor(plot\_img,

cv2.COLOR\_RGB2BGR))

if cv2.waitKey(1) & 0xFF == ord('q'): break

end\_time = time.time()

elapsed\_time = end\_time - start\_time fps\_text = f"FPS: {1 / elapsed\_time:.2f}" print(fps\_text) # Print FPS for debugging

cap.release() cv2.destroyAllWindows()

known\_heights = { 'person': 1.7,

'tank': 2.4,

'vehicle': 1.5,

'truck': 3.5,

'car': 1.5

}

# Path to the input video file

video\_path = "C:/Users/user/PycharmProjects/yolo/Video6.mp4"

# Create an instance of ObjectDetection with the known heights dictionary and video path

if name == " main ":

object\_detection = ObjectDetection(video\_path, class\_heights=known\_heights) object\_detection()

# ДОДАТОК Б

