НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: “Система вимірювання відстані до об’єктів на відео”

Виконав: студент 4 курсу, групи ІВ-93
(шифр групи)

Цоколов Максим Володимирович
(прізвище, ім’я, по батькові) (підпис)

Керівник ст.викладач Алєнін О.І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ст.викладач Виноградов Ю.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2023 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Цоколова Максима Володимировича

1. Тема проєкту *“Система вимірювання відстані до об’єктів на відео”*
керівник проєкту ст.викладач Алєнін О.І._____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 31 травня 2023 року №2101-с
2. Термін здачі студентом закінченого проєкту *8 червня 2023 р.*
3. Вихідні дані до проєкту *технічна документація. теоретичні та статистичні дані, патенти на винахід*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)
Розділ 1. Огляд та аналіз існуючих рішень вимірювання відстані на відео.

Розділ 2. Огляд методів та технологій розробки системи вимірювання відстані до об’єктів на відео.

Розділ 3. Деталі розробки системи.

5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) Пристрій вимірювання відстані (Структурна схема), Застосунок вимірювання відстані (Функціональна схема), Алгоритм дії системи вимірювання відстані (Принципова схема).

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Виноградов Ю.М.		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.02.2023-25.02.2023*	
2.	*Вивчення та аналіз завдання*	*25.02.2023-15.04.2023*	
3.	*Розробка архітектури та загальної структури системи*	*25.03.2023-25.04.2023*	
4.	*Розробка структур окремих підсистем*	*07.04.2023-30.04.2023*	
5.	*Програмна реалізація системи*	*20.04.2023-25.05.2023*	
6.	*Оформлення пояснювальної записки*	*17.04.2023-21.05.2023*	
7.	*Захист програмного продукту*	*25.05.2023*	
8.	*Передзахист*	*10.06.2023*	
9.	*Захист*	*22.06.2023*	

Студент-дипломник

(підпис)

Керівник проєкту

(підпис)

АНОТАЦІЯ

Цей дипломний проект має на меті розв'язання проблеми вимірювання відстані до об'єктів на відео. В результаті аналізу та розробки було натреновано нейронну мережу, що може визначати об’єкти на відео. Окрім цього було розроблено алгоритм, який може визначати відстань до об’єктів, використовуючи дані про камеру, об’єкт та дані з нейромережі. Дана система має веб-інтерфейс для взаємодії. Розроблена система може використовуватись не лише для розпізнавання відстані до об’єктів на відео, що вже було записане, але й на відео що транслюється. Система реалізована на мові програмування Python з використанням фреймворків OpenCV, PyTorch і Werkzeug. Для реалізації веб-інтерфейсу використовується бібліотека Flask.

ANNOTATION

This diploma project aims to solve the problem of estimation the distance to objects in a video. As a result of the analysis and development, a neural network was trained that can detect objects in video. In addition, an algorithm was developed that can estimate the distance to objects using camera, object, and neural network data. This system has a web interface for interaction. The developed system can be used not only to recognize the distance to objects in the video that has already been recorded, but also in the video that is being broadcast. The system is implemented in the Python programming language using the OpenCV, PyTorch and Werkzeug frameworks. The Flask library is used to implement the web interface.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.001 ОА***	Система вимірювання відстані	1		
			до об’єктів на відео			
			Опис альбому			
	А4	***ІАЛЦ.467200.002 ТЗ***	Система вимірювання відстані	3		
			до об’єктів на відео			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Система вимірювання відстані	67		
			до об’єктів на відео			
			Пояснювальна записка			
	А1	***ІАЛЦ.467200.004 Д1***	Система вимірювання відстані	1		
			до об’єктів на відео			
			Пристрій вимірювання відстані			
			(Структурна схема)			
	А1	***ІАЛЦ.467200.005 Д2***	Система вимірювання відстані	1		
			до об’єктів на відео			
			Застосунок вимірювання відстані			
			(Функціональна схема)			
	А1	***ІАЛЦ.467200.006 Д3***	Система вимірювання відстані	1		
			до об’єктів на відео			
			Алгоритм дії системи вимірювання відстані			
			(Принципова схема)			
	А1	***ІАЛЦ.467200.007 Д4***	Система вимірювання відстані	1		
			до об’єктів на відео			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***	
Розроб		Цоколов М.В.			*Система вимірювання відстані до об’єктів на відео* Опис альбому
Перев		Алєнін О.І			
					КПІ ім. Ігоря Сікорського ФІОТ ІВ-93

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система вимірювання відстані до об’єктів на відео*»

Київ – 2023 р.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Це технічне завдання поширюється на розробку системи вимірювання відстані до об’єктів на відео. Область застосування: визначення відстані до людей та об’єктів на відео високої роздільної якості, що були зняті з дронів.

2. ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського проєкту по освітньо-професійної програми “Комп’ютерні системи та мережі” спеціальності 123 “Комп’ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка системи вимірювання відстані до об’єктів на відео.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література по нейромережах статті з прикладної фізики та довідники по розробці веб-додатків, публікації в інтернеті з цих питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Робота на зображеннях високої якості – система має змогу працювати із відео високої якоті.
- Визначення відстані до окремих типів об’єктів - система має вимірювати відстань до заданих об’єктів, в даному випадку - людей.
- Система має мати простий інтерфейс взаємодії для демонстрації та тестування її роботи.

5.2. Вимоги до програмного забезпечення

- Вимоги до програмного забезпечення включають використання мови програмування Python 3.9, фреймворків Flask(2.2.3), Werkzeug(2.2.2),

- Ultralytics(8.0.107), OpenCV(4.7.0), сумісність між компонентами та достатні ресурси пам'яті та продуктивності.
- Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

5.3. Вимоги до апаратної частини

- Для реалізації проекту потрібен потужний комп'ютер з багатоядерним процесором, достатньою оперативною пам'яттю та швидким диском. Рекомендується використання відеокарти з підтримкою CUDA або OpenCL для прискорення обчислень. Мінімальні вимоги включають наявність чотирьохядерного процесора, 8 ГБ оперативної пам'яті та швидкого диску.

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	*10.02.2023-25.02.2023*
Вивчення та аналіз завдання	*25.02.2023-15.04.2023*
Розробка архітектури та загальної структури системи	*25.03.2023-25.04.2023*
Розробка структур окремих частин системи	*07.04.2023-30.04.2023*
Програмна реалізація системи	*20.04.2023-25.05.2023*
Виправлення помилок	*25.05.2023-30.05.2023*
Оформлення пояснювальної записки	*17.04.2023-21.05.2023*

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Система вимірювання відстані до об’єктів на відео*»

Київ – 2023 р.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

GPU	(Graphics Processing Unit) Графічний процесор
OpenCL	(Open Computing Language) Відкрита мова обчислень
AVI	(Audio Video Interleave) Чергування аудіо та відео
MPEG	(Moving Picture Experts Group) Експертна група з питань рухомого зображення. Один з форматів стиснення аудіо та відео.
MP4	(MPEG-4 Part 14) MPEG-4 Частина 14
YOLO	(You Only Look Once) Ти дивишся лише раз
YAML	(YAML Ain't Markup Language) YAML не є мовою розмітки
SGD	(Stochastic Gradient Descent) Стохастичний градієнтний спуск

ВСТУП

В сучасному світі швидкого технологічного розвитку візуальна обробка даних займає все більш важливе місце у різних галузях, починаючи від автономних автомобілів до систем безпеки та нагляду. Одним із важливих завдань у цьому контексті є визначення відстані до об'єктів на відео, що є необхідним етапом аналізу та прийняття рішень.

Ця робота присвячена дослідженню та розробці системи вимірювання відстані до об'єктів на відео. Метою роботи є створення ефективного та точного рішення, яке може знайти своє застосування у різних галузях, де вимірювання відстані є важливим елементом, таких як транспортні засоби з можливостями автономного керування, системи контролю та безпеки, розумні міста (за аналогом до розумної системи керування у Чикаго) та багато інших.

Для досягнення поставленої мети дипломна робота складається з трьох основних розділів. У першому розділі проводиться огляд понять, методів та систем, пов'язаних з вимірюванням відстані на відео.

Другий розділ присвячений детальному дослідженню методів, алгоритмів та технологій, що використовуються для розробки системи вимірювання відстані.

В третьому розділі представлено процес розробки системи, включаючи обробку та вибір даних, тренування моделі, оцінку її роботи та розробку веб-інтерфейсу для взаємодії із моделлю.

Ця робота має на меті не лише описати рішення для вимірювання відстані на відео, але й дослідити його ефективність та потенціал застосування. Очікується, що результати цієї роботи не тільки розширять загальні знання про вимірювання відстані до об'єктів на відео, але й сприятимуть подальшому розвитку систем візуального аналізу даних та забезпеченню безпеки та ефективності в різних галузях.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ВИМІРЮВАННЯ ВІДСТАНІ НА ВІДЕО.

1.1 Визначення поняття відстань та її вимірювання на відео

1.1.1 Поняття відстань та її значення в різних областях

Згідно з визначенням джерела [1], відстань - це міра віддаленості об'єктів або точок один від одного. У фізиці або повсякденному вжитку відстань може означати фізичну довжину або оцінку, засновану на інших критеріях. Існують різні способи визначення відстані між фізичними об'єктами в різних контекстах. Пряма або евклідова відстань - це відстань між двома точками у фізичному просторі, яка дорівнює довжині прямої лінії між ними, найкоротшого можливого шляху. Прямолінійна відстань математично формалізується як евклідова відстань у дво- та тривимірному просторі. В евклідовій геометрії відстань між двома точками A і B часто позначають |AB|. У координатній геометрії евклідову відстань обчислюють за допомогою теореми Піфагора. Відстань між точками (x1, y1) і (x2, y2) на площині задається формулою:

$$d = \sqrt{((\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2)} =$$
$$= \sqrt{((x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2)}$$

1.1.2 Методи вимірювання відстані на відео та їх характеристики

Існує багато способів вимірювання прямолінійних відстаней, наприклад, за допомогою лінійки або опосередковано за допомогою радара (для великих відстаней) або інтерферометрії (для дуже коротких відстаней). Джерело [1] згадує про "космічні сходи", які є набором способів вимірювання надзвичайно довгих відстаней.

Деякі із способів вимірювання відстані на відео включають:

- Піксельні методи:
    - Опис: відстань визначається за кількістю пікселів, які займає об'єкт у кадрі відео.
    - Характеристики: простий метод, але схильний до помилок через зміну кута нахилу камери, розміру об'єкта та відстані.
- Масштабно-інваріантне перетворення ознак (SIFT):
    - Опис: порівнює об'єкти на зображенні з базою даних відомих об'єктів для оцінки відстані.
    - Характеристики: точніше, ніж піксельні методи, але вимагає значних обчислювальних ресурсів і великої бази даних відомих об'єктів.
- Структура з руху (SFM):
    - Опис: використовує послідовність зображень для створення 3D-моделі середовища та оцінки відстані.
    - Характеристики: точний метод, але вимагає значної кількості даних і обчислювальних ресурсів.
- Оптичний потік:
    - Опис: вимірює рух пікселів у послідовних відеокадрах для оцінки відстані.
    - Характеристики: швидкий і відносно точний, але схильний до помилок через затінення об'єктів і зміни в освітленні.
- Методи на основі машинного навчання (наприклад, YOLOvN):
    - Опис: використовує моделі глибокого навчання для виявлення об'єктів і оцінки відстані на основі їхнього розміру та положення у відеокадрі.
    - Характеристики: високоточні, але вимагають великого набору даних для навчання і значних обчислювальних ресурсів.

1.2 Огляд існуючих систем вимірювання відстані на відео

1.2.1 Огляд систем, що використовуються в різних галузях

Відповідно до посилань в джерелі [2], існує декілька існуючих систем вимірювання відстані.

Одними із них є алгоритми карт розбіжності стерео зору (Stereo vision disparity map algorithms), які використовуються для оцінки відстані між двома камерами та об'єктом.

Ще один методом є система вимірювання відстані для автономних транспортних засобів за допомогою стереокамери використовує зображення зі стереокамери для оцінки відстані між камерою та об'єктом.

Ще один спосіб це монокулярна оцінка відстані з використанням апроксимації точкової камери для уникнення аварій та аварій з перекиданням транспортного засобу, яка використовує апроксимацію точкової камери для оцінки відстані між камерою та об'єктом, щоб уникнути аварійних ситуацій.

Наступним методом є багатопроменева оптична фазова решітка для дальньої передачі даних LiDAR і даних у відкритому космосі, яка використовує багатопроменеву оптичну фазову решітку для вимірювання відстані для LiDAR і передачі даних у відкритому космосі

За даними джерела [3], відеовимірювання відстаней на основі фокуса - це метод визначення відстаней у полі монобачення. Цей метод передбачає фіксацію об'єктива в положенні, що відповідає максимальній різкості отриманого зображення, відомому як глибина різкості (DFF). DFF - це відстань між найближчим і найвіддаленішим об'єктами в сцені, які виглядають на зображенні прийнятно різкими. Глибина різкості може бути використана для оцінки відстані об'єкта від камери.

Ще одне джерело [4] обговорює метод вимірювання відстані до рухомих об'єктів на відео. У статті використовується метод різниці фонів для виділення цільового об'єкта, а потім обчислюється відстань між цільовим об'єктом і

камерою на основі розміру цільового об'єкта. Цей метод можна використовувати для вимірювання відстані рухомих об'єктів на відео.

Існують різні системи вимірювання відстані, в тому числі ті, що базуються на алгоритмах карти розбіжностей стереозображення, апроксимації камери-обскури, багатопроменевої оптичної фазової решітки та трьох позицій фокусування. Ці методи мають різні сильні і слабкі сторони, і вибір методу залежить від конкретних вимог застосування.

1.2.2 Характеристика існуючих систем та їхні переваги та недоліки

Характеристики існуючих відеосистем вимірювання відстані, їхні переваги та недоліки залежать від конкретного методу, що використовується. Наприклад, система, запропонована у статті [5], використовує дві камери, змонтовані як одна стереокамера, для виявлення транспортних засобів і обчислення відстані між ними за допомогою геометричних похідних і додаткових технічних даних. Перевагами цієї системи є висока точність і ефективне вимірювання в реальному часі, а недоліком - те, що вона вимагає спеціального обладнання і технічних знань.

В статті [6] пропонується система для людей з вадами зору, яка використовує виявлення об'єктів для оцінки відстані. Перевагами цієї системи є її точність і короткий час ідентифікації, а недоліком - те, що вона може не підходити для всіх застосувань.

Система відео-вимірювання відстані, запропонована в джерелі [7], використовує систему відео-вимірювання відстані (VDM), яка є такою ж точною, як і традиційні системи EDM. Перевагами цієї системи є більш точна ідентифікація точки вимірювання і відсутність знаряддя (шипа), яке може змістити пісок навколо мітки, а недоліком - те, що вона може не підходити для всіх застосувань.

Загалом, характеристики, переваги та недоліки існуючих систем відеовимірювання відстані залежать від конкретного методу, що

використовується, та вимог застосування. Кожна система має свої сильні та слабкі сторони, і вибір системи залежить від конкретних потреб проекту.

1.3 Аналіз методів та алгоритмів вимірювання відстані на відео

1.3.1 Приклади методів та алгоритмів вимірювання відстані на відео

В статті [8] для обчислення відстані від об'єкта на відео до камери використовується метод подібності трикутників. Цей метод передбачає знання фокусної відстані камери, висоти об'єкта і радіуса кругового маркера на об'єкті. Відстань можна обчислити за формулою:

$$d = \frac{h * f}{r}$$

де d - відстань, h - висота об'єкта, f - фокусна відстань камери а r - радіус кругового маркера.

Серед інших методів визначення відстані на відео - метод комплексного логарифмічного мапування (CLM), який передбачає перетворення зображень з ортогональної системи координат у полярну і визначення співвідношення між двома фотографіями на основі властивостей концентричних кіл. Інший метод передбачає оцінку глибини за допомогою монокулярної камери та параметрів камери і геометрії зображення. Третій метод передбачає використання однієї камери зі змінним кутом нахилу для отримання вимірювань відстані до об'єкта шляхом оптимізації за методом найменших квадратів.

За даними джерела [8], вимірювання відстані є важливим для різних застосувань, наприклад, для надання інформації про виявлені об'єкти та їхню абсолютну відстань у сцені, знятій системою перед автомобілем, або для автономних транспортних засобів, мульти-дронів і роботів. Інформація про відстань також необхідна комп'ютеру для розпізнавання навколишнього середовища і є важливою технологією для ідентифікації та вимірювання відстані до інших об'єктів, що знаходяться поблизу.

Інші технології, які зараз використовуються для визначення відстані, включають радар і лідар. Радар використовує електромагнітні хвилі для обчислення відстаней, тоді як лідар використовує відбиття променів для вимірювання відстані, форми і речовини навколишнього середовища.

Загалом, існують різні методи та алгоритми визначення відстані на відео, залежно від конкретного застосування та наявного обладнання і технічних знань. Метод подібності трикутників, CLM, оцінка монокулярної камери та методи камер зі змінним кутом нахилу - це лише кілька прикладів з багатьох доступних методів.

1.3.2 Характеристика методів та алгоритмів, їхні переваги та недоліки

Характеристики методів і алгоритмів визначення відстані на відео залежать від конкретного підходу, що використовується. Наприклад, традиційні методи оцінки відстані вимагають складного калібрування внутрішніх і зовнішніх параметрів камери, як пояснюється у статті [4]. Новітні методи, засновані на нейромережевих структурах, можуть підвищити швидкість і точність вимірювання відстані, але можуть вимагати значних обчислювальних ресурсів і технічних знань.

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі було проведено вивчення та аналіз різних аспектів вимірювання відстані до об'єктів на відео, а також огляду існуючих систем та методів, які використовуються для цього.

В ході роботи було визначено поняття відстані та його значення в різних областях. Також були оглянуті різні методи вимірювання відстані на відео та їх характеристики.

Далі був проведений огляд існуючих систем вимірювання відстані на відео, зокрема у різних галузях застосування. Були розглянуті характеристики цих систем, а також їх переваги та недоліки.

Також був проведений аналіз методів та алгоритмів. Були розглянуті приклади вкиористання, а також надана характеристика переваг та недоліків різних підходів.

Загалом, в розділі було досліджено поняття відстані, методи вимірювання відстані на відео та проведено оглядд на існуючі системи та методи, які застосовуються для визначення відстані. Цей розділ надає необхідну базу знань для подальшого розроблення системи вимірювання відстані до об'єктів на відео.

РОЗДІЛ 2 ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ СИСТЕМИ ВИМІРЮВАННЯ ВІДСТАНІ ДО ОБ'ЄКТІВ НА ВІДЕО

2.1 Огляд бібліотеки OpenCV та її функціональних можливостей

2.1.1 Загальна характеристика OpenCV та її використання в комп'ютерному зорі

OpenCV - це широко використовувана бібліотека комп'ютерного зору з відкритим вихідним кодом, розроблена компанією Intel, яка надає повний набір можливостей обробки зображень для комп'ютерного зору в реальному часі. Вона доступна безкоштовно за ліцензією Apache 2 і працює під управлінням найпопулярніших операційних систем, таких як GNU/Linux, OS X, Windows, Android та iOS, що робить її дуже універсальним інструментом для розробників. Інструмент OpenCV особливо корисний для машинного навчання в галузі комп'ютерного зору та містить повну бібліотеку машинного навчання загального призначення, орієнтовану на статистичне розпізнавання образів і кластеризацію. Ця бібліотека оптимізована для обчислювально інтенсивних задач машинного зору і здатна використовувати переваги багатоядерних процесорів. Також вона містить понад 2500 алгоритмів, велику документацію та приклади коду для комп'ютерного зору в реальному часі. [9] OpenCV написано мовами C та C++, а також зараз активно розробляються інтерфейси для Python, Ruby, Matlab та інших мов. Також вона надає функціональність для апаратного прискорення NVIDIA CUDA та графічних процесорів (GPU), а також Open Computing Language (OpenCL), що робить її високоефективним та продуктивним інструментом для операцій у реальному часі. Бібліотека OpenCV є популярним інструментом, який використовується великими підприємствами та державними установами, такими як Google, Toyota, IBM, Microsoft, SONY, Siemens та Facebook. Вона використовується

відомими стартапами в галузі комп'ютерного зору для створення потужних продуктів комп'ютерного зору та рішень ШІ. Багато дослідницьких центрів використовують OpenCV, серед них Стенфорд, Массачусетський технологічний інститут, INRIA, Кембридж та CMU.

Бібліотека OpenCV зосереджена на застосуванні штучного зору в реальному часі і надає прості у використанні інструменти під цю задачу, надаючи понад 500 функцій, які охоплюють багато областей, що допомагає людям швидко створювати складні програми. Також OpenCV аквтивно використовується для контролю продукції на виробництві, медичної візуалізації, аналізу безпеки, людино-машинного інтерфейсу, калібрування камер, стереозоріння (3D-бачення) та роботизованого зору. [9]

OpenCV є найбільш універсальним інструментом для розробки в сфері комп'ютерного зору. Цю бібліотеку використовують в різних сферах, починаючи від розпізнавання зображень, 2D або 3D аналізу та відстеження руху до розпізнавання облич. Також ця бібліотека активно використовується для технологій виявлення об'єктів, що застосовується для розпізнавання зображень і визначення місцезнаходження конкретних об'єктів у відеоданих або зображеннях, таких як автомобілі, люди, тварини, а також конкретні деталі або обладнання в промисловому виробництві. Бібліотека OpenCV використовується і для сегментації зображень, що в процесі сегмантації застосовує алгоритми обробки зображень для поділу зображення на різні частини. Сегментація зазвичай застосовується для спрощення, заміни або покращення зображення, часто в поєднанні з додатковими завданнями комп'ютерного зору. OpenCV також використовується для розпізнавання людських поз і жестів, щоб інтерпретувати і розуміти їх за допомогою аналізу відео. Інші можливості бібліотеки включають автоматичне розпізнавання облич, доповнену реальність і розрахунок пози об'єкта, щоб забезпечити метод розуміння того, як об'єкт розташований у 3D-просторі, наприклад, як він обертається. [9]

OpenCV використовується в численних програмах, продуктах і дослідницьких проектах. Ці програми включають зшивання зображень з камер на супутникових або веб-картах, вирівнювання сканованих зображень, зменшення шуму на медичних зображеннях, аналіз об'єктів, системах безпеки, спостереження та виявлення вторгнень, системах автоматичного моніторингу, системах виробничого контролб ШІ, калібрування камер, оборонних та військових програмах, а також у безпілотних повітряних, наземних та підводних апаратах. Цей інструмент навіть використовувався для розпізнавання звуку та музики, де методи розпізнавання зору застосовувалися до зображень звукових спектрограм. [9]

Останнім часом розробка без коду або з низьким рівнем коду стала новим способом для підприємств та організацій доставляти та підтримувати рішення набагато швидше та ефективніше. Розробка комп'ютерного зору зазвичай дуже складна та вимагає численних циклів ітерацій розробки. Таким чином, впровадження комп'ютерного зору значно виграє від можливості візуальної розробки та можливості автоматизовано розгортати безкодові технології. Платформа комп'ютерного зору Viso Suite надає можливості OpenCV у вигляді модульних будівельних блоків, які можна використовувати для швидкого створення додатків комп'ютерного зору без написання коду з нуля. Це дозволяє командам розробників швидше використовувати бібліотеку і полегшувати інтеграцію з різним обладнанням, таким як камери, периферійні комп'ютери та моделі машинного навчання.

Існує незліченна кількість варіантів використання, які можна побудувати за допомогою бібліотеки OpenCV, включаючи аналіз медичних зображень для підтримки діагностики людини, розпізнавання реклами на телебаченні або розпізнавання логотипів за допомогою штучного зору, відстеження гравців у спорті та фітнесі, розпізнавання сцен та аналіз якості виконання, підрахунок кількості людей у громадських місцях, таких як аеропорти, роботизована автоматизація для інтелектуальних інтерфейсів на основі зору, автоматичний огляд та аналіз відео за допомогою постійного

комп'ютерного зору, пошук зображень на цифрових платформах, виявлення дефектів або несправностей у виробничих процесах, підрахунок кількості транспортних засобів на автомагістралі, а також додатки з камерами відеоспостереження, що дозволяють виявити випадки фізичного насильства, атак та порушень правил дорожнього руху.

Отже, OpenCV - це універсальна бібліотека для комп'ютерного зору з відкритим вихідним кодом, яка надає повний набір можливостей обробки зображень для комп'ютерного зору в реальному часі. Вона широко використовується великими підприємствами, державними установами, стартапами та дослідницькими центрами. Бібліотека написана мовами C та C++ і працює під управлінням найпопулярніших операційних систем, таких як GNU/Linux, OS X, Windows, Android та iOS. Також дана бібліотека має сильну орієнтацію на роботу в режимі реального часу.

2.1.2 Огляд основних функцій та алгоритмів OpenCV для обробки відео

OpenCV містить модуль аналізу відео, який включає алгоритми оцінки руху, віднімання фону та відстеження об'єктів. Алгоритм оцінки руху використовується для визначення руху об'єкта у відеопослідовності. Він реалізований за допомогою методу блочного зіставлення, який порівнює пікселі у двох послідовних кадрах відео. Алгоритм віднімання фону використовується для відокремлення об'єкта на передньому плані від фону. Застосовується для виявлення рухомих об'єктів на відео і реалізується за допомогою моделей гауссового змішування (GMM). Алгоритм відстеження об'єкта використовується для відстеження об'єкта у відеопослідовності. Він реалізований за допомогою фільтра Калмана, який передбачає положення об'єкта в наступному кадрі відео. [10]

OpenCV також надає модуль вводу/виводу відео (videoio), який включає простий у використанні інтерфейс для захоплення відео та відеокодеків. Модуль підтримує читання і запис відео у різних форматах, включаючи AVI,

MPEG і MP4. Він також підтримує захоплення відео в реальному часі з камер і відеопотоків. Модуль надає функції для налаштування властивостей захоплення відео, таких як роздільна здатність, частота кадрів і кодек, а також для отримання інформації про відео, такої як кількість кадрів, частота кадрів і роздільна здатність. [10]

Для обробки відео за допомогою OpenCV у Python можна створити об'єкт cv2.VideoCapture, який є класом для захоплення відео з відеофайлів, послідовностей зображень або камер. Для відтворення відео можна створити цикл while, який зчитує кожен кадр відео за допомогою функції cap.read(). Функція повертає булеве значення, яке вказує на успішність зчитування кадру і сам кадр. Цикл продовжується до тих пір, поки не залишиться жодного кадру для зчитування. У межах циклу можна виконувати різні завдання з обробки відео, такі як застосування фільтрів, виявлення об'єктів або відстеження руху об'єктів.

2.2. Огляд архітектури та особливостей YOLOv8

2.2.1 Загальна характеристика архітектури мережі YOLOv8

YOLOv8 - це найновіша модель виявлення об'єктів, яка спирається на своїх попередників, використовуючи нові методи та засоби оптимізації. Найновіша версія цієї моделі була випущена 10 січня 2023 року. Однією з ключових особливостей даної моделі є її архітектура, що легко налаштовується та дозволяє користувачам легко змінювати структуру та параметри моделі відповідно до своїх потреб. Архітектура YOLOv8 складається з повністю згорткової нейронної мережі, яку можна розділити на дві основні частини: основу та верх. [11]

Основа моделі - це модифікована версія архітектури CSPDarknet53, яка складається з 53 згорткових шарів і використовує техніку під назвою міжступеневі часткові зв'язки для покращення потоку інформації між різними

шарами мережі. Верхівка YOLOv8 складається з декількох згорткових шарів, за якими слідує серія повністю з'єднаних шарів, які відповідають за прогнозування обмежувальних рамок, оцінок об'єктності та ймовірностей класів для об'єктів, виявлених на зображенні .

Одним з ключових покращень в моделях YOLO є використання механізму самоконтролю у верху мережі. Цей механізм дозволяє моделі фокусуватися на різних частинах зображення і регулювати важливість різних ознак, виходячи з їхньої релевантності для завдання. Ще однією важливою особливістю є здатність виконувати різномасштабне виявлення об'єктів, що досягається за допомогою мережі піраміди ознак . Ця мережа складається з декількох шарів, які виявляють об'єкти різного масштабу, що дозволяє моделі виявляти великі та малі об'єкти на зображенні.

YOLOv8 також підтримує різні магістралі, такі як EfficientNet, ResNet і CSPDarknet, надаючи користувачам гнучкість у виборі найкращої моделі для конкретного випадку використання. Крім того, в моделі використовується адаптивне навчання для оптимізації швидкості під час навчання та збалансування функції втрат, що призводить до кращої продуктивності моделі. Дана модель також використовує передові методи доповнення даних, такі як MixUp та CutMix, для покращення надійності та узагальнення.

З точки зору точності, існує декілька моделей виявлення об'єктів, які вважаються передовими, такі як EfficientDet, DETR та Cascade R-CNN. Хоча YOLOv8 і може конкурувати з цими моделями, вибір моделі для використання в кінцевому підсумку залежить від конкретного випадку використання та вимог.

Для використання YOLOv8 вам знадобиться комп'ютер з графічним процесором, підтримка фреймворку глибокого навчання (наприклад, PyTorch або TensorFlow) та доступ до GitHub YOLOv8. Програмна база YOLOv8 має відкритий вихідний код і доступна для досліджень і розробок на GitHub.

Як і його попередники, YOLOv8 створено для ефективної роботи на стандартному обладнанні, що робить його надійним рішенням для задач

виявлення об'єктів у реальному часі, в тому числі в умовах обмежених ресурсів. YOLOv8 продемонстрував покращену точність порівняно з попередніми версіями YOLO та є повністю конкурентоспроможним з найсучаснішими моделями виявлення об'єктів. [12]

Рисунок 2.2 – Опціональні задачі які може виконувати YOLOv8 [12]

Отже, YOLOv8 - це модель для виявлення об'єктів, що легко налаштовується. Ця модель використовує модифіковану версію магістралі CSPDarknet53 і має механізм самоконтролю у вершині мережі для досягнення підвищеної точності і виявлення різномасштабних об'єктів. Модель підтримує різні базові мережі, адаптивне навчання та передові методи розширення даних для покращення продуктивності моделі. Хоча YOLOv8 успішно конкурує з найсучаснішими моделями виявлення об'єктів, кінечний вибір моделі для використання в залежить від конкретного випадку використання та вимог.

2.2.2 Огляд основних особливостей та переваг YOLOv8 для обробки відео

YOLOv8 - це найсучасніша модель виявлення об'єктів, яка пропонує неперевершену продуктивність з точки зору швидкості та точності [13]. Однією з головних переваг YOLOv8 для обробки відео є її підвищена швидкість, оскільки вона досягає вищої швидкості обробки, ніж інші моделі виявлення об'єктів, зберігаючи при цьому високу точність. Це робить цю модель придатною для задач обробки відео в реальному часі, наприклад, для систем автономного керування транспортними засобами та систем спостереження. YOLOv8 також підтримує різні опорні мережі, такі як

EfficientNet, ResNet і CSPDarknet, надаючи користувачам гнучкість у виборі найкращої моделі для їхнього конкретного випадку використання.

Ще однією перевагою YOLOv8 для обробки відео є його здатність виконувати багатомасштабне виявлення об'єктів, що досягається за допомогою мережі піраміди ознак. Ця мережа складається з декількох шарів, які виявляють об'єкти різного масштабу, що дозволяє моделі виявляти великі та малі об'єкти на зображенні. Це особливо корисно для задач обробки відео, де об'єкти можуть мати різні пропорції та розміри.

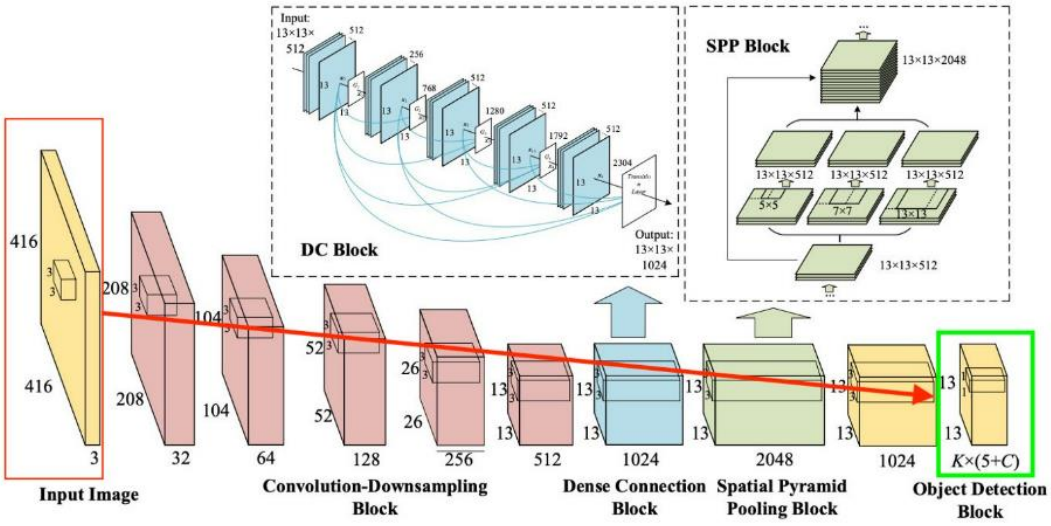


Рисунок 2.2 – Схема обробки зображень у YOLOv8 [13]

Крім того, YOLOv8 використовує передові методи доповнення даних, такі як MixUp та CutMix, щоб покращити надійність та узагальнення моделі. Це важливо для задач обробки відео, де модель повинна вміти виявляти об'єкти в різних умовах освітлення, фону та орієнтації.

YOLOv8 також добре налаштовується, дозволяючи користувачам легко змінювати структуру та параметри моделі відповідно до своїх потреб. Це корисно для задач обробки відео, де вимоги можуть змінюватися залежно від конкретного випадку використання. YOLOv8 також надає попередньо навчені моделі для легкого використання та перенесення навчання на різні набори даних.

Загалом, YOLOv8 - це потужний і універсальний алгоритм виявлення об'єктів, який можна використовувати в різних реальних сценаріях для задач обробки відео, таких як автономні транспортні засоби, відеоспостереження та робототехніка. Підвищена швидкість, виявлення різномасштабних об'єктів, передові методи доповнення даних та архітектура, що налаштовується, роблять його придатним рішенням для широкого спектру застосувань.

2.3. Огляд методів побудови датасетів для розпізнавання об'єктів на відео

2.3.1 Загальна характеристика методів побудови датасетів для тренування мережі YOLOv8

Створення набору даних є важливим кроком у навчанні моделі YOLOv8 на користувацькому наборі даних [14]. Одним з перших кроків у створенні набору даних є збір зображень та їх анотування за допомогою обмежувальних рамок навколо об'єктів, що представляють для користувача інтерес. Це можна зробити за допомогою різних інструментів анотування, таких як LabelImg або Roboflow. Анотації повинні містити мітку класу для кожного об'єкта, а також координати обмежувальної рамки [15].

Після того, як зображення анотовані, їх потрібно розділити на навчальну, перевірочну та тестову множини. Навчальна множина використовується для навчання моделі YOLOv8, тоді як валідаційна множина використовується для оцінки роботи моделі під час навчання та налаштування гіперпараметрів. Тестовий набір використовується для оцінки кінцевої продуктивності моделі після навчання.

Наступним кроком є створення YAML-файлу, який визначає розташування навчальних, валідаційних та тестових наборів, а також мітки класів для об'єктів у наборі даних. Цей файл використовується навчальним скриптом YOLOv8 для завантаження даних і налаштування процесу навчання.

Файл YAML повинен містити шляхи до файлів зображень та анотацій, а також кількість класів та назви класів.

Після підготовки набору даних модель YOLOv8 можна навчати за допомогою команди yolo train [16]. Під час навчання параметри моделі оптимізуються таким чином, щоб вона могла точно передбачати класи та розташування об'єктів на зображенні. Процес навчання може тривати кілька годин або навіть днів, залежно від розміру набору даних та обраних гіперпараметрів.

Для точного налаштування моделі YOLOv8 існує безліч гіперпараметрів, які можна регулювати, наприклад, швидкість навчання, розмір пакету, спадання значень ваги тощо. Оптимальні значення цих гіперпараметрів залежать від конкретного набору даних і сценарію використання. Рекомендується експериментувати з різними гіперпараметрами та оцінювати продуктивність моделі на валідаційному наборі, щоб знайти найкращу комбінацію.

Отже, створення набору даних для навчання моделі YOLOv8 включає збір та анотування зображень, розділення набору даних на навчальний, валідаційний та тестовий набори, створення YAML-файлу для налаштування процесу навчання та точне налаштування моделі за допомогою різних гіперпараметрів. Створення високоякісного набору даних має вирішальне значення для досягнення хорошої продуктивності в задачах виявлення об'єктів і вимагає ретельної уваги до деталей і точності анотацій.

2.3.2 Огляд основних джерел даних та методів побудови датасетів для розпізнавання об'єктів на відео

Згідно з джерелом [17], створення набору даних для розпізнавання об'єктів на відео передбачає збір та анотування зображень з обмежувальними рамками навколо об'єктів, що нас цікавлять, як було вказано в пункті 2.3.1.

Джерело [18] також підкреслює важливість наявності адекватного еталонного набору даних для розпізнавання об'єктів на відео. Хоча багато

наборів даних було створено для конкретних завдань, деякі з них мають обмеження, наприклад, містять лише об'єкти з фронтальним видом або упередженість щодо певного регіону зображення. Для створення належного набору даних необхідна велика кількість об'єктів з різною орієнтацією та без упереджень.

У джерелі [19] наведено список найкращих наборів відеоданих для розпізнавання об'єктів у 2022 році, включаючи набір даних BDD100K, який є найбільшим відкритим набором відеоданих для розпізнавання об'єктів. Цей набір даних використовується для водіння і включає відстеження сегментації декількох об'єктів, тегування зображень, виявлення дорожніх об'єктів, семантичну сегментацію, виявлення смуг руху, сегментацію проїжджої частини, сегментацію екземплярів, відстеження виявлення декількох об'єктів, адаптацію до домену та імітаційне навчання.

У [18] також згадуються інші набори даних, зокрема ті, що використовуються для розпізнавання облич, розпізнавання тексту, розпізнавання пішоходів, дорожніх знаків, світлофорів і виявлення цілей за даними дистанційного зондування. Ці набори даних пов'язані з виявленням об'єктів і можуть бути корисними для конкретних завдань.

Отже, створення набору даних для розпізнавання об'єктів на відео передбачає збір і анотування зображень, поділ їх на різні набори та запис результатів. Наявність адекватного еталонного набору даних має вирішальне значення, і для різних застосувань доступні різні набори даних.

2.4 Огляд методів оптимізації та прискорення роботи системи вимірювання відстані на відео

Оптимізація та прискорення роботи системи вимірювання відстані у відео передбачає використання різних методів. Нижче наведено деякі характеристики цих методів:

- Згорткові нейронні мережі (ЗНМ) є окремим видом штучних нейронних мереж (ШНМ) та знаходять своє застосування в комп'ютерному зорі та паралельних розподілених обчисленнях для обробки великих обсягів даних, що генеруються датчиками, а також для задоволення енергетичних обмежень пристроїв IoT. ШНМ можна використовувати для оптимізації та прискорення роботи системи вимірювання відстані на відео шляхом навчання мережі розпізнавати та аналізувати різні об'єкти на відео. Однак навчання нейронних мереж є громіздким і займає багато часу, який може займати дні або навіть тижні. Це обмежує застосування ШНМ у дослідницьких галузях, де швидкість обчислень є надзвичайно важливою. Таким чином, існує потреба у відповідній та підвищеній обчислювальній швидкості, щоб задовольнити вимоги цих додатків у реальному часі [20].
- Стохастична градієнтна оптимізація (Stochastic Gradient Decent, SGD) може бути використана для прискорення швидкості навчання ШНМ без шкоди для точності. SGD-оптимізація - це стохастична версія алгоритму оптимізації градієнтного спуску. Це ітераційний метод, який оновлює ваги нейронної мережі для мінімізації помилки на виході. SGD-оптимізація є обчислювально ефективною і може бути використана для оптимізації роботи системи вимірювання відстані у відео [20].
- Швидка конволюція - це ще одна техніка, яка може бути використана для прискорення швидкості навчання ШНМ. Швидка згортка - це математичний метод, який зменшує кількість обчислень, необхідних для виконання операції конволюції. Вона базується на алгоритмі швидкого перетворення Фур'є (ШПФ), який є ефективним алгоритмом для обчислення дискретного перетворення Фур'є (ДПФ) послідовності. Швидка конволюція може бути використана для оптимізації роботи системи вимірювання відстані у відео за рахунок зменшення обчислювальної складності операції конволюції [20].

- Використання паралелізму - це ще одна техніка, яка може бути використана для прискорення швидкості навчання ШНМ. Паралелізм передбачає розподіл робочого навантаження між декількома процесорами або ядрами для одночасного виконання обчислень. Ця техніка може бути використана для оптимізації роботи системи вимірювання відстані у відео за рахунок зменшення часу, необхідного для виконання обчислень [20].

Отже, оптимізація та прискорення роботи системи вимірювання відстані у відео включає в себе різні методи, такі як CNNs, SGD-оптимізація, швидка конволюція та використання паралелізму. Кожен з цих методів має свої переваги та недоліки, і вибір методу залежить від конкретних вимог програми.

2.5 Огляд методів покращення точності вимірювання відстані до об'єктів на відео

Підвищення точності вимірювання відстані до об'єктів на відео передбачає використання різних методів. Нижче наведено деякі характеристики цих методів:

- Системи стерео зору використовують дві камери для захоплення зображень об'єкта під різними кутами, які потім використовуються для визначення положення об'єкта в тривимірному просторі. Використовуючи методи оптимізації, такі як нейронні мережі, можна підвищити точність вимірювання відстані. Наприклад, в роботі [21] розроблено та протестовано систему стерео зору для виявлення загальних джерел невизначеності. Методи оптимізації використовуються для навчання нейронної мережі, а отримане рівняння може бути реалізоване в системах стереозображення в реальному часі. Проведено обчислювальні експерименти та порівняльний аналіз для визначення навчальної функції з мінімальною похибкою для такого методу. Запропонований метод є універсальною технікою моделювання, придатною для вирішення різноманітних проблем, що виникають у

системах стерео зору. Нарешті, запропонований метод застосовано до розробленої системи стереозображення і проведено статистичний аналіз для перевірки отриманих покращень.

- Конфігурації ємнісного зондування сенсорного масиву та машинне навчання можуть бути використані для оцінки відстані та позиціонування цільового об'єкта. У [22] запропоновано новий метод оцінки відстані та позиціонування, який поєднує 14 конфігурацій зондування для створення 14 ємнісних датчиків для виявлення цілі. Ємності використовуються як вхідний вектор моделей машинного навчання, і порівнюються чотири алгоритми машинного навчання для оцінки відстані вручну. Результати експерименту показують, що мережа з радіально-базисною функцією (RBF) має найкращі показники: середньоквадратична похибка (RMSE), середня абсолютна похибка (MAE) і середня відносна похибка (MRE) становлять 0,59 см, 0,45 см і 5,32%, відповідно. Результати тестування порівнюються з використанням Y-подібного розташування датчиків і показують, що запропонований метод працює краще, ніж метод з Y-подібним розташуванням датчиків, точність покращується більш ніж на 98,8%, а модель BPNN досягає найвищої точності.
- Для вимірювання відстані до об'єкта можна використовувати одну візуальну камеру. Однак, важко досягти високої точності через відсутність інформації про глибину. В роботі [24] запропоновано новий метод підвищення точності вимірювання відстані на основі однієї візуальної камери. Досліджено причину, чому вимірювання відстані за допомогою однієї камери не може досягти високої точності, і запропоновано новий метод для підвищення точності. Запропонований метод оцінюється і порівнюється з існуючими методами, щоб продемонструвати ефективність і високу точність.

Підсумовуючи, можна сказати, що для підвищення точності визначення відстані до об'єктів на відео використовуються різні методи, такі як системи

стерео зору, конфігурації ємнісного зондування масиву сенсорів і машинного навчання, а також методи, засновані на використанні однієї візуальної камери. Кожен з цих методів має свої переваги і недоліки, і вибір методу залежить від конкретних вимог програми.

2.5.1 Огляд основних методів покращення точності вимірювання відстані з використанням OpenCV та YOLOv8

У відкритих джерелах не згадуються конкретні методи підвищення точності визначення відстані до об'єктів на відео за допомогою OpenCV та YOLOv8. Однак є деякі пов'язані з цим обговорення, які можуть бути корисними для досягнення цієї мети.

Один з можливих підходів полягає у використанні OpenCV для виявлення об'єктів у кадрах відео, а потім застосування методів оцінки відстані на основі виявлених об'єктів. Наприклад, в [25] виявлення об'єктів відбувається за допомогою каскадів Хаара, а потім відстань до об'єкта оцінюється на основі розміру об'єкта на зображенні. Це можна зробити за відомим розміром об'єкта або шляхом калібрування камери для оцінки розміру об'єкта.

Інший підхід полягає у використанні виявлення контурів для визначення місцезнаходження об'єктів на кадрах відео, а потім застосування методів оцінки відстані на основі контурів. У [26] розглядається метод вимірювання відстані між двома контурами на відео за допомогою OpenCV. Метод полягає у виявленні контурів у кадрах відео та обчисленні відстані між ними за формулою евклідової відстані. Однак цей метод працює лише для вимірювання відстані між двома контурами і може бути непридатним для більш складних випадків.

Загалом, підвищення точності визначення відстані до об'єктів на відео вимагає поєднання методів виявлення об'єктів, виявлення контурів та оцінки відстані. YOLOv8 - це сучасний алгоритм виявлення об'єктів, який можна використовувати в поєднанні з OpenCV для виявлення об'єктів у відеокадрах.

Після виявлення об'єктів можуть бути застосовані різні методи оцінки відстані в залежності від конкретних вимог програми. Наприклад, системи стерео зору, конфігурації ємнісного зондування масиву сенсорів і машинного навчання, а також методи на основі однієї візуальної камери можуть бути використані для оцінки відстані до об'єктів на відео з високою точністю, як обговорювалося в попередній відповіді.

Отже, підвищення точності визначення відстані до об'єктів на відео за допомогою OpenCV і YOLOv8 передбачає поєднання методів виявлення об'єктів, виявлення контурів і оцінки відстані. Конкретні методи, що використовуються, залежать від вимог програми, і для досягнення високої точності існують різні способи.

2.6 Огляд методів підтримки множини камер для вимірювання відстані

2.6.1 Загальна характеристика методів підтримки множини камер для вимірювання відстані до об'єктів на відео

Існує кілька методів обчислення відстані до об'єкта за допомогою однієї фіксованої камери, а також різні методики, які є дуже простими і лаконічними в області комп'ютерного зору. Однак останнім часом методи обчислення відстані стали сферою великого дослідницького інтересу в галузі робототехніки та комп'ютерного зору. Вимірювання відстані - це метод, який використовується в різних додатках. Наприклад, необхідно визначити відстань до системи, що знімає сцену перед автомобілем. Це дозволяє надати інформацію про виявлені об'єкти та їх абсолютні відстані. Крім того, інформація про відстань необхідна для автономних транспортних засобів, мульти-дронів та роботів. Наразі радар і лідар дозволяють визначати відстані. Проблема оцінки відстані до об'єкта на відео наразі широко досліджується в галузі обробки зображень та комп'ютерного зору [8]. Одним з підходів до вирішення цієї проблеми є використання декількох камер. Основним

компонентом цього методу є багатошарова нейронна мережа Multi-DisNet, яка вивчає взаємозв'язок між розмірами обмежувальних рамок об'єкта на зображеннях з камер і відстанню між об'єктом і камерами [26]. Використання декількох камер надає дані, необхідні для застосування методів дискретизації за відстанню перехоплення мітки [27]. У багатьох недавніх роботах почали використовувати кілька датчиків або камер для різних типів задач, таких як побудова тривимірних зображень, виявлення оклюзії тощо [28].

Існує багато методів обчислення відстані до об'єкта за допомогою однієї фіксованої камери. Одним з найефективніших методів у комп'ютерному зорі є метод подібності трикутників, який є найбільш розвиненою технікою. З його допомогою ми можемо обчислити відстань від камери до відомого об'єкта. Метод базується на принципах роботи камери, і він досить простий і зрозумілий, якщо відомий реальний розмір об'єкта. Відстань від об'єкта до камери можна виміряти за допомогою однієї камери зі змінним кутом нахилу, покращеним шляхом оптимізації за методом найменших квадратів. Це простий і точний метод, який дозволяє виміряти відстань до об'єкта за допомогою камери зі змінним кутом нахилу.

Стереозір - це метод, який використовується для оцінки глибини точкового об'єкта з камери за допомогою двох камер. Основа стерео зору схожа на сприйняття 3D в людському зорі і базується на тривимірному обчисленні променів з декількох точок зору. Стерео зіставлення - це процес пошуку відповідних точок на двох зображеннях, знятих двома різними камерами. Ми можемо взяти вікно/блок пікселів на лівому зображенні і використовувати його як шаблон для пошуку відповідного вікна такого ж розміру на правому зображенні. Існує низка методів розрахунку подібності [29].

При проектуванні стереосистеми ми хочемо виміряти відстань дуже точно, оскільки саме вона дає нам глибину. Для цього нам потрібно використовувати стереоконфігурацію, в якій базова лінія є достатньо великою,

оскільки чим більша базова лінія, тим точніше ми можемо виміряти диспропорцію.

Отже, існує декілька методів підтримки декількох камер для визначення відстані до об'єктів на відео. Використання декількох камер надає дані, необхідні для застосування методів дискретизації відстані по міткам, тому в багатьох останніх роботах почали використовувати декілька датчиків зору або камер для різних типів задач. Стереобачення - це метод, який використовується для оцінки відстані до точкового об'єкта з камери за допомогою двох камер. Існує безліч методів розрахунку відстані до об'єкта за допомогою однієї фіксованої камери, а також різні методи, які є дуже простими і компактними в області комп'ютерного зору. Вимірювання відстані - це метод, який використовується в різних додатках, а методи обчислення відстані останнім часом стали сферою великого дослідницького інтересу в робототехніці та комп'ютерному зорі.

2.6.2 Огляд основних методів підтримки множини камер для вимірювання відстані до об'єктів на відео з OpenCV та YOLOv8

Для визначення відстані до об'єктів на відео можна використовувати декілька камер за допомогою OpenCV та YOLOv8. OpenCV надає кілька методів розрахунку відстані до об'єкта за допомогою однієї фіксованої камери, наприклад, метод подібності трикутників, який можна використовувати для обчислення відстані від камери до відомого об'єкта. Стереобачення - ще один метод, який може бути використаний для оцінки глибини точкового об'єкта від камери за допомогою двох камер.

Для реалізації виявлення об'єктів та вимірювання відстані за допомогою YOLOv8 у репозиторії Object-Detection-and-Distance-Measurement на GitHub є скрипт під назвою object_detection.py. Скрипт використовує YOLOv3 для виявлення об'єктів у відеокадрі, а потім обчислює відстань об'єкта від камери, використовуючи інформацію про глибину, яку камера використовує для малювання обмежувальних рамок для локалізації об'єктів [30]. Вимірювання

відстані відбувається шляхом обчислення ширини і висоти обмежувальної рамки, які змінюються залежно від відстані об'єкта від камери. Відстань можна обчислити за такою формулою:

$$distance = (2 \text{ x } 3{,}14 \text{ x } 180) \div (w + h \text{ x } 360) \text{ x } 1000 + 3$$

де w і h - ширина і висота обмежувальної рамки відповідно. Формула враховує заломлення зображення при проходженні через об'єктив камери, а також різницю в розмірах об'єкта на зображенні, що виникає в результаті.

Іншим методом підтримки декількох камер для визначення відстані до об'єктів на відео є використання багатошарової нейронної мережі під назвою Multi-DisNet, яка вивчає зв'язок між розмірами обмежувальних рамок об'єктів на зображеннях з камер та відстанню між об'єктом і камерами [32]. Цей метод корисний для отримання даних, необхідних для застосування методів дискретизації за відстанню перехоплення мітки .

Отже, існує декілька методів підтримки декількох камер для визначення відстані до об'єктів на відео за допомогою OpenCV та YOLOv8. OpenCV надає методи для обчислення відстані до об'єкта за допомогою однієї фіксованої камери, тоді як стереобачення можна використовувати для оцінки глибини точкового об'єкта з камери, що використовує дві камери. У репозиторії Object-Detection-and-Distance-Measurement на GitHub є скрипт, який використовує YOLOv3 для виявлення об'єктів у відеокадрі та обчислення відстані об'єкта від камери, використовуючи інформацію про глибину, яку камера використовує для малювання обмежувальних рамок для локалізації об'єктів. Інший метод полягає у використанні багатошарової нейронної мережі Multi-DisNet, яка вивчає зв'язок між розмірами обмежувальних рамок об'єктів на зображеннях камер і відстанню між об'єктом і камерами.

2.7 Огляд методів автоматичного калібрування системи вимірювання відстані до об'єктів

У статті [33] представлено повністю автоматичну і гнучку процедуру калібрування стаціонарних багатокамерних систем для тривимірного спостереження динамічних подій на основі послідовності зображень. Метод базується на стаціонарних камерах і рухомих мішенях, використовуючи природу отримання послідовності зображень більшості твердотільних камер. Метод базується на відстеженні одного маркера, що легко виявляється, через послідовності зображень декількох попередньо відкаліброваних камер, таким чином уникаючи необхідності ідентифікації гомологічних ознак для встановлення відповідності між кількома видами. У більш досконалій версії еталонна смуга відомої довжини переміщується в просторі об'єкта, при цьому проблема ідентифікації ознак і встановлення багаторакурсних відповідностей зводиться до відстеження двох цілей. Постійна довжина еталонної лінійки використовується як додаткова інформація про геометричні обмеження при самокалібруванні пучка, що значно посилює рішення і дозволяє виконати повне калібрування кожної камери багатокамерної системи, включаючи параметри внутрішньої орієнтації.

Методи "переміщення однієї точки" і "переміщення опорної лінії" є простими і зрозумілими в реалізації, але мають певні обмеження. Якщо зовнішню орієнтацію і параметри викривлення об'єктива можна визначити, то внутрішню орієнтацію камер неможливо реконструювати за інформацією єдиної переміщеної точки. Встановлення відповідностей також зводиться до виявлення або відстеження двох об'єктів або цілей, що робить його менш придатним для самокалібрування. Метод "переміщеної референтної лінії" можна використовувати для визначення внутрішньої орієнтації декількох камер, однак існує брак детального аналізу методу, а калібрувальних схем, що включають специфікації щодо ідеальної кількості, розташування та орієнтації

послідовних спостережень за референтною лінією, які слід використовувати для калібрування системи, не існує.

Інші методи автоматичного калібрування системи визначення відстані до об'єктів на відео включають використання калібрувального шаблону, наприклад, шахової дошки або сітки, яка знімається камерами під різними кутами для оцінки внутрішніх і зовнішніх параметрів камер. OpenCV надає функції для калібрування камер, зокрема calibrateCamera() та stereoCalibrate(), які можна використовувати для калібрування одно- та стереокамер відповідно. Ці функції оцінюють матрицю камери, коефіцієнти спотворення, а також вектори повороту і трансляції, які можуть бути використані для розкладання і виправлення зображень, а також для обчислення тривимірного положення точок у сцені [34].

2.8 Загальна характеристика методів автоматичної калібрування системи вимірювання відстані до об'єктів на відео

Одним з найпоширеніших методів автоматичного калібрування системи визначення відстані до об'єктів на відео з використанням OpenCV є виконання шахового калібрування камери. Цей метод усуває радіальні та тангенціальні спотворення, які можуть впливати на вихідне зображення, а отже, і на вихідні вимірювання об'єктів на зображенні [35]. OpenCV надає функції для калібрування камери, зокрема calibrateCamera() та stereoCalibrate(), які можна використовувати для калібрування одно- та стереокамер відповідно. Ці функції оцінюють матрицю камери, коефіцієнти спотворення, а також вектори повороту і трансляції, які можуть бути використані для спотворення і виправлення зображень, а також для обчислення тривимірного положення точок у сцені.

Інший метод автоматичного калібрування системи визначення відстані до об'єктів на відео полягає у використанні еталонного об'єкта відомого розміру, наприклад, шахової дошки або сітки, і оцінці внутрішніх і зовнішніх

параметрів камер за зображеннями об'єкта з різних ракурсів. Цей метод схожий на метод калібрування камери за шахівницею, але використовує відомий об'єкт розміру, а не шахівницю. Після оцінки внутрішніх і зовнішніх параметрів відстань до об'єктів на відео можна обчислити за допомогою таких методів, як метод подібності трикутників або стереобачення.

Отже, основними методами автоматичного калібрування системи визначення відстані до об'єктів на відео з використанням OpenCV та YOLOv8 є виконання шахового калібрування камери та використання еталонного об'єкта відомого розміру. OpenCV надає функції для калібрування камери, за допомогою яких можна оцінити матрицю камери, коефіцієнти спотворення, а також вектори обертання і трансляції. Після того, як ці параметри оцінені, відстань до об'єктів на відео може бути обчислена за допомогою різних методів, таких як метод подібності трикутників та стереобачення.

2.9 Огляд фреймворків та бібліотек для розробки веб-додатків з можливістю обробки відеоданих

2.9.1 Дослідження можливостей Flask для розробки веб-додатків з можливістю обробки відеоданих

Flask - це легкий фреймворк веб-додатків WSGI, який розроблений для швидкого і зручного старту, з можливістю масштабування до складних додатків [36]. Flask - це мікро-фреймворк, який використовується для швидкого створення простих веб-додатків. Він включає підтримку шаблонів Jinja, обробку запитів та передачу сигналів для додатків. Flask пропонує більшу гнучкість у порівнянні з самовпевненим фреймворком Django, що робить його більш придатним для програмістів з більшим досвідом кодування або тих, кому потрібен більший контроль над дизайном додатку [37].

Flask можна використовувати для розробки веб-додатків з можливістю обробки відеоданих. Наприклад, у навчальному посібнику з потокової передачі відео у веб-браузерах за допомогою Flask та OpenCV автор демонструє, як використовувати Flask та OpenCV для потокової передачі відео з веб-камери у веб-браузері [37]. Інший підручник з розгортання програми обробки відео на Python на сервері з використанням Flask і OpenCV містить код для обробки відеокадрів за допомогою OpenCV і відображення обробленого відео у веб-браузері за допомогою Flask [38].

Flask також можна використовувати для створення веб-додатків, які дозволяють користувачам завантажувати та обробляти відеофайли. Наприклад, у навчальному відео на YouTube про потокове відео з веб-камери у веб-фреймворку Flask автор демонструє, як використовувати Flask для створення веб-додатку, який дозволяє користувачам завантажувати відеофайл, а потім обробляє і відображає відеофайл у веб-браузері за допомогою OpenCV і Flask [39].

Отже, Flask - це легкий фреймворк для веб-додатків, який можна використовувати для розробки веб-додатків з можливістю обробки відеоданих. Flask можна використовувати для трансляції відео з веб-камери у веб-браузері, обробки відеокадрів за допомогою OpenCV та створення веб-додатків, які дозволяють користувачам завантажувати та обробляти відеофайли.

2.9.2 Розгляд можливостей використання Flask для створення веб-додатків з використанням нейромереж

Flask можна використовувати для створення веб-додатків, що використовують штучний інтелект для розпізнавання відеозображень. Насправді, Flask можна використовувати в поєднанні з бібліотеками штучного інтелекту, такими як OpenCV, для створення таких додатків. Наприклад, у навчальному посібнику про те, як відображати потокове відео з веб-камери за

допомогою Flask, автор демонструє, як використовувати Flask і OpenCV для захоплення і відображення відео у веб-браузері [40]. Інший посібник зі створення веб-додатку для машинного навчання з використанням Flask описує, як за допомогою Flask створити веб-додаток, який дозволяє користувачам завантажувати зображення, а потім використовує модель машинного навчання для класифікації зображень [41].

Для інтеграції ШІ для розпізнавання відеозображень у веб-додаток Flask можна використовувати бібліотеку ШІ, наприклад, OpenCV. Наприклад, у навчальному посібнику з інтеграції API Clarifai у додаток Flask автор демонструє, як використовувати Flask і API Clarifai для розпізнавання зображення їжі [42]. Аналогічно, за допомогою Flask і моделі машинного навчання було створено веб-додаток для прогнозування зображення цифри ручного знаку, як це описано в навчальному посібнику з побудови додатку Flask для розпізнавання зображень [43].

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі було проведено дослідження і огляд різноманітних методів, технологій та бібліотек, які використовуються для системи вимірювання відстані до об'єктів на відео.

Починаючи з огляду бібліотеки OpenCV та її функціональних можливостей, була розглянута загальна характеристика OpenCV та її використання в комп'ютерному зорі. Далі були розглянуті основні функції та алгоритми OpenCV для обробки відео.

Далі була проведена характеристика архітектури мережі YOLOv8 та огляд її особливостей і переваг для обробки відео. Дослідження включало загальну характеристику методів побудови датасетів для тренування мережі YOLOv8, а також огляд основних джерел даних та методів побудови датасетів для розпізнавання об'єктів на відео.

Потім було оглянуто методи оптимізації та прискорення роботи системи вимірювання відстані на відео, а також методи покращення точності визначення відстані до об'єктів на відео.

Також досліджено методи підтримки множини камер для визначення відстані до об'єктів на відео, включаючи загальну характеристику цих методів і огляд основних методів підтримки множини камер з використанням OpenCV та YOLOv8.

Було проведено огляд методів автоматичної калібрування системи визначення відстані до об'єктів на відео, включаючи загальну характеристику цих методів та їхніх недоліків і переваг.

Нарешті, були розглянуті фреймворки та бібліотеки для розробки веб-додатків з можливістю обробки відеоданих, зокрема досліджені можливості Flask для розробки таких веб-додатків і розглянуті можливості використання Flask для створення веб-додатків з використанням штучного інтелекту для розпізнавання образів на відео. До того ж було визначено переваги цього

фреймворку над іншими, що принципово покращує, чого він був обраний для виконання дипломного проєкту.

Загалом, в даному розділі детально досліджено різні аспекти, методи, алгоритми та технології, які стосуються системи вимірювання відстані до об'єктів на відео, зокрема з використанням бібліотеки OpenCV, архітектури YOLOv8, принципу побудови датасетів, оптимізації та покращення точності визначення відстані, підтримки множини камер, автоматичного калібрування та розробки веб-додатків для обробки відеоданих та зображень.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ

3.1 Вибір алгоритму визначення об’єктів

Вибір відповідного алгоритму для визначення об'єктів є критичним кроком у завданні комп'ютерного зору. Одним із найсучасніших та ефективних алгоритмів для визначення об'єктів є YOLOv8. Цей алгоритм вражає своєю здатністю працювати в реальному часі та досягати високої точності вирізняння об'єктів.

YOLOv8 (You Only Look Once, version 8) є одним із останніх розробок у сімействі алгоритмів YOLO. Він використовує глибокі нейронні мережі, зокрема згорткові нейронні мережі, для виявлення та класифікації об'єктів на зображенні чи відео. Одна з головних переваг YOLOv8 - це його швидкість та здатність працювати в режимі реального часу. Це робить його особливо привабливим для випадків, де важлива швидка обробка відеоданих або потоку зображень. (Рис. 3.1)

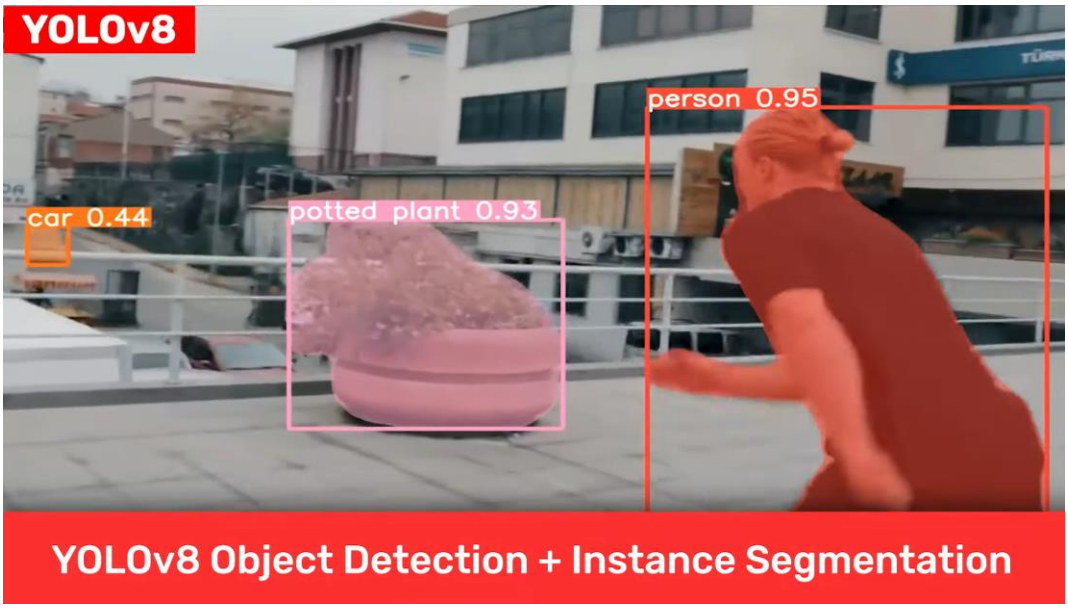


Рисунок 3.1 - Демонстрація роботи алгоритму YOLOv8 [23]

Найбільш вражаючою рисою YOLOv8 є його висока точність вирізнення об'єктів. Алгоритм здатний впевнено та точно визначати об'єкти різних класів, незалежно від їх розміру та форми. Він демонструє високу стійкість до змін у

контексті зображень, таких як зміна освітлення, перекриття об'єктів або зміна кута огляду. Це робить його універсальним та надійним інструментом для розпізнавання об'єктів у різних сценах та умовах.

Також YOLOv8 є алгоритмом, що здатний визначати відразу багато об'єктів на одному зображенні або відеокадрі одночасно. Він дозволяє виявляти та вирізняти об'єкти з різних класів, забезпечуючи повний контекст та повну інформацію про зображення. Це особливо корисно для сценаріїв, де присутні різноманітні об'єкти, а також для завдань, де потрібно визначати та розуміти взаємодію між об'єктами. (Рис. 3.2)

Рисунок 3.2 - Результати визначення людей на зображенні від YOLO [13]

Отже, вибір YOLOv8 як алгоритму для визначення об'єктів обґрунтовується його сучасністю, здатністю працювати в режимі реального часу та високою точністю вирізняння об'єктів. Цей алгоритм є потужним та надійним інструментом у сфері комп'ютерного зору, який може використовуватись для різноманітних завдань, від автономних транспортних засобів до систем відеоспостереження та розширеної реальності.

3.2 Вибір методу вимірювання відстані

Детекція та визначення відстані є одним з ключових завдань в області комп'ютерного зору. Цей процес полягає в ідентифікації та локалізації об'єктів на зображенні або відео. В останні роки було розроблено багато алгоритмів та методів для вирішення цієї задачі, і вибір підходящого алгоритму може бути важким завданням.

Один з можливих підходів до визначення відстані до об'єктів полягає у використанні інформації про пікселі та фокусну відстань камери. Цей метод базується на використанні геометричних властивостей зображення та фізичних характеристик об'єктів. Існують деякі переваги використання цього підходу, зокрема простота реалізації та можливість використання з однією камерою в режимі прямого ефіру. (Рис. 3.3)

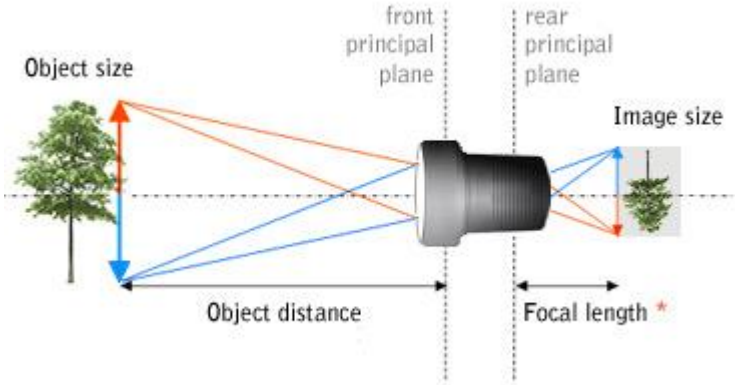


Рисунок 3.3 - Демонстрація роботи алгоритму на основі Focal Length [35]

Значення пікселів на зображенні можуть бути використані для визначення розташування та форми об'єктів. Це може бути досягнуто за допомогою аналізу інтенсивності пікселів та їх взаємного відношення. Наприклад, різниця у яскравості між об'єктом та фоном може свідчити про наявність об'єкта на зображенні. Такі методи можуть бути реалізовані шляхом використання фільтрів, порігового значення або відстеження контурів. Також для визначення розмірів об’єкта можуть використовуватись моделі на основі нейромереж, такі як YOLO, що визначають розташування об’єкта на зображенні в прямому ефірі без затримок. (Рис. 3.4)

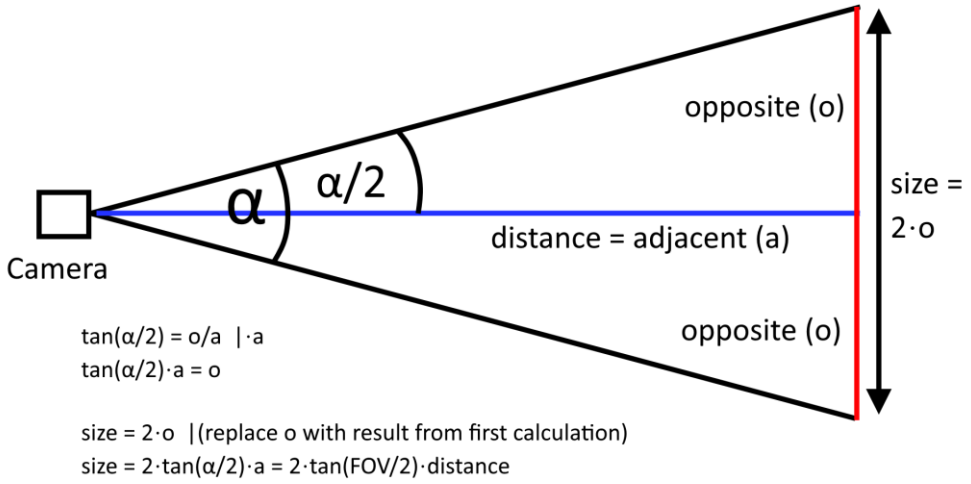


Рисунок 3.4 - Демонстрація роботи алгоритму вимірювання відстані на основі пікселів

Також для визначення відстані до об'єктів на зображенні може бути використана фокусна відстань камери. Це може бути здійснено шляхом аналізу розмірів об'єктів на зображенні та їх відстані до камери. Знання фокусної відстані дозволяє виконувати глибинне визначення об'єктів, що є важливим для багатьох сфер, таких як робототехніка, автономні транспортні засоби та віртуальна реальність.

Об'єднуючи інформацію про пікселі та фокусну відстань камери, можна отримати більш точну інформацію про розташування та властивості об'єктів. Знання про відстань до об'єктів дозволяє враховувати їх просторові взаємозв'язки, що може бути корисним при визначенні орієнтації та масштабу об'єктів.

Вибір алгоритму визначення об'єктів на основі пікселів та фокусної відстані є бути обґрунтованим із декількох причин. По-перше, цей метод має просту реалізацію, оскільки він базується на стандартних методах аналізу зображень та обчислення фокусної відстані камери. Це означає, що його можна легко використовувати без потреби в складних обчислювальних ресурсах або спеціалізованому обладнанні.

По-друге, метод на основі пікселів та фокусної відстані може бути успішно використаний з однією камерою в режимі прямого ефіру. Це означає, що він може бути застосований в реальному часі без необхідності обробки або аналізу записаних відеоданих. Це робить його ефективним і зручним для використання в широкому спектрі застосувань, де важлива миттєва реакція на об'єкти.

Враховуючи простоту реалізації та можливість використання з однією камерою в прямому ефірі, метод на основі пікселів та фокусної відстані є привабливим вибором для визначення об'єктів.

3.3 Вибір даних

Вибір правильних даних є ключовим етапом в розробці системи визначення об'єктів. Для розробки системи визначення об’єктів на зображенні було обрано відкритий набір даних із Kaggle. З огляду на опис датасету, що вказаний на сайті, наданий набір даних має кілька переваг і може бути відповідним вибором для розробки об'єктного детектора.

Цей датасет був створений AIoT Lab в Національному технологічному університеті Тайваню з використанням фотографій високої роздільної здатності, знятих з дронів, що оснащені 4K відеокамерами. Висока роздільність зображень може значно покращити здатність сучасного детектора об'єктів розпізнавати навіть невеликі об'єкти. Відмінною особливістю цього датасету є те, що він містить реальні фотографії, зняті з дронів, а не просто масштабовані зображення. Це дозволяє отримати більш реалістичні та відповідні умовам реального світу дані для навчання моделі детектування об'єктів. (Рис. 3.5)

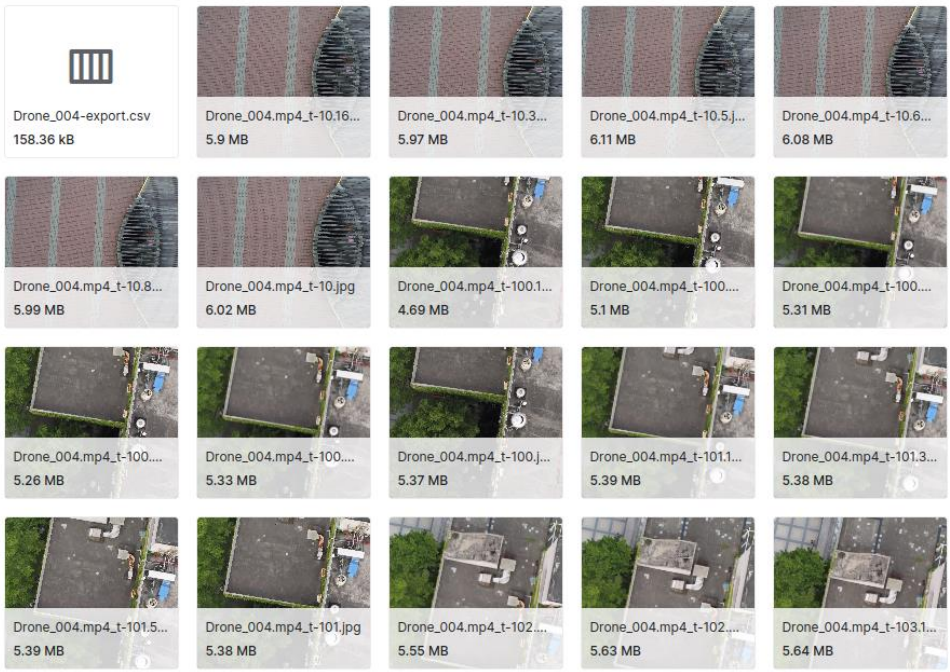


Рисунок 3.5 - Вигляд зображень в наборі даних

Як видно із зображень, наведених вище, картинки для тренування являють собою набір кадрів із відео, що розділені по певних проміжках. Кожна картинка займає велику кількість простору в пам’яті, в середньому 5.4Мб. Розміри картинок становлять 3840x2160 пікселів, що відповідає розмірам 4К зображень. (Рис. 3.6)

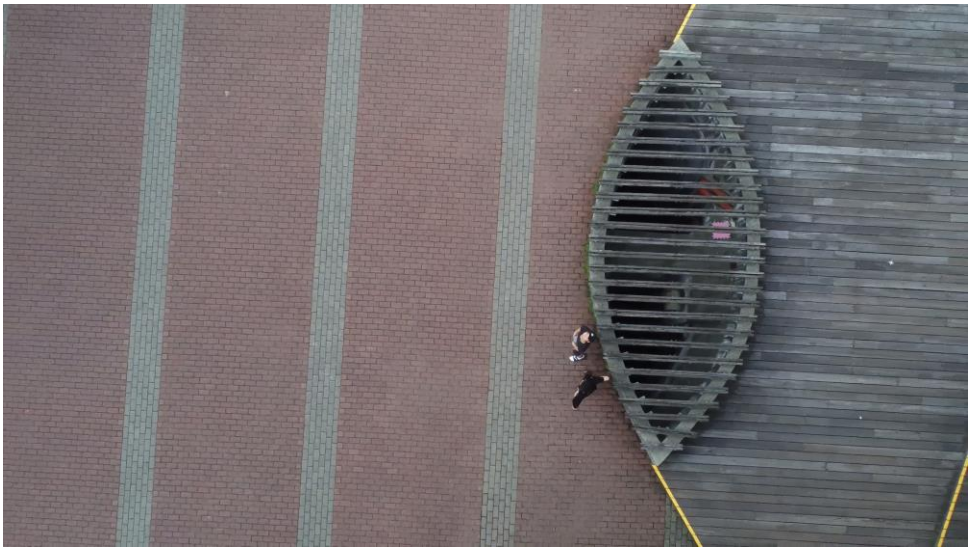


Рисунок 3.6 - Одне із зображень, наданих в датасеті.

Датасет містить чотири набори даних, які були зняті в різних обставинах. Об'єкти в цих наборах були промарковані. Маркуванні включають людей, що йдуть, стоять, їдуть, дивляться в телефон і т.д. Таке маркування надає дані про важливу різноманітність об'єктів, що допоможе моделі більш точно розпізнавати людей, оскільки додаваення нових класів схильне покращувати розпізнавання моделей YOLO. (Рис. 3.7)

Рисунок 3.7 - Вигляд розмічених зображеннь

Дані про зображення зберігаються у форматі CSV для кожного набору даних. Ці CSV файли містять інформацію про розташування об'єктів та посилання на зображення, до якого вони належать. Це дозволяє легко організувати та використовувати дані для навчання моделі. Координати (xmin, ymin, xmax, ymax) та мітки (label) об'єктів вказані в CSV файлі, що дозволяє точно визначити розміри та класи об'єктів. (Рис. 3.8)

image	xmin	ymin	xmax	ymax	label
Drone_005.mp4_t-12.833333	2195.703128246235	159.44943494639233	2343.9043687699805	299.991714358157	walk
Drone_005.mp4_t-12.833333	2195.703128246235	159.44943494639233	2343.9043687699805	299.991714358157	push
Drone_005.mp4_t-12.833333	2485.488565488675	220.78817733990147	2626.528066528067	367.0935960591133	walk
Drone_005.mp4_t-12.666667	2219.653232196073	154.12923790205735	2367.854472720084	294.67151731382205	walk
Drone_005.mp4_t-12.666667	2219.653232196073	154.12923790205735	2367.854472720084	294.67151731382205	push
Drone_005.mp4_t-12.666667	2509.438669438671	207.48768472906403	2650.478170478171	353.7931034482758	walk
Drone_005.mp4_t-12.5	2243.603336146411	135.50854824688494	2391.804576670188	276.0508276586496	walk
Drone_005.mp4_t-12.5	2243.603336146411	135.50854824688494	2391.804576670188	276.0508276586496	push
Drone_005.mp4_t-12.5	2544.033264033265	186.20689655172413	2685.072765072765	332.5123152709359	walk
Drone_005.mp4_t-12.333333	2259.570072113147	114.22776006954504	2407.771312636923 5	254.7700394813097 4	walk
Drone_005.mp4_t-12.333333	2259.570072113147	114.22776006954504	2407.771312636923 5	254.7700394813097 4	push

Рисунок 3.8 - Вигляд файлу із даними про зображення

Також, датасет розбитий на тестовий та тренувальний набори, що є важливим для оцінки та налаштування моделі детектування об'єктів. Розділення на train та test папки дозволяє коректно використовувати дані для тренування та оцінки моделі, забезпечуючи правильну перевірку її продуктивності на незалежних наборах даних. (Рис. 3.9)

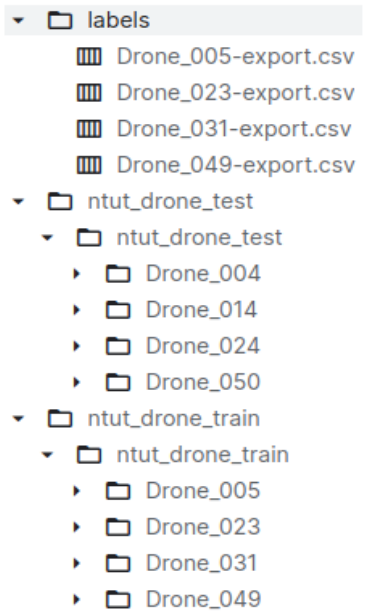


Рисунок 3.9 - Структура директорії набору даних

Отже, датасет, описаний вище, може бути доволі гарним вибором для розробки системи визначення об'єктів, оскільки він містить високороздільні зображення, різноманітність розмаркованих об'єктів та відповідні дані для навчання та тестування моделі.

3.4 Обробка даних

Для тренування обраної моделі дані потрібно перевести у спеціальний формат. Формат даних для YOLO detection вимагає двох окремих видів файлів для проведення тренування: файлу зображень та файлу міток.

Файл зображень має такі ознаки:

- Кожне зображення повинно мати свій унікальний ідентифікатор, наприклад, ім'я файлу або шлях до зображення.
- Зображення можуть бути у різних форматах, наприклад, JPEG або PNG.

Файл міток має такі ознаки:

- Кожен рядок у файлі міток відповідає одному об'єкту на зображенні і має такий формат: <клас> <x-центр> <y-центр> <ширина> <висота>
    - <клас> - мітка класу об'єкту, це може бути числове значення або текстова назва класу.
    - <x-центр>, <y-центр> - відносні координати центру об'єкту, де (0, 0) є верхнім лівим кутом, а (1, 1) - нижнім правим кутом зображення.
    - <ширина>, <висота> - відносні розміри об'єкту відповідно до ширини та висоти зображення.
- У файлі міток можуть бути вказані мітки для кількох об'єктів на одному зображенні. Кожний рядок представляє окремий об'єкт.

Таблиця 3.1 - Приклад файлу міток

0	0.45	0.60	0.30	0.40
1	0.75	0.30	0.60	0.55
2	0.20	0.70	0.25	0.35

У наведеному вище прикладі перший рядок означає, що на зображенні є об'єкт класу 0 з центром у координатах 0.45 та 0.60, шириною 0.30 та висотою 0.40. Другий рядок означає, що є об'єкт класу 1 з центром у координатах 0.75 та 0.30, шириною 0.60 та висотою 0.55. Третій рядок означає, що є об'єкт класу 2 з центром у координатах 0.20 та 0.70, шириною 0.25 та висотою 0.35.

Дані у файлі мають бути нормалізованими, щоб поміщатись у проміжок від 0 та 1. Це зроблено із метою надання змоги працювати алгоритму із будь-яким розміром зображень.

Датасет для моделі YOLO має мати специфічну структуру директорії для навчання:

```
dataset/
├── images/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
└── labels/
    ├── image1.txt
    ├── image2.txt
    └── ...
```

, де:

- dataset - головна папка, що містить усі дані для тренування або тестування моделі YOLO.
- images - папка, де зберігаються зображення. Кожне зображення повинно мати свій унікальний ідентифікатор, наприклад, ім'я файлу або шлях до зображення.
- labels - папка, де зберігаються файли міток для кожного зображення. Кожен файл міток відповідає конкретному зображенню та містить інформацію про об'єкти, що виявлені на зображенні у форматі, придатному для YOLO моделі. Зазвичай файли міток мають той самий ідентифікатор, що й відповідні зображення, але з розширенням .txt або іншим відповідним розширенням.

Дана структура директорії може бути повторена для наборів для тестування та валідації моделі.

Крім того, для тренування моделі потрібно створити YAML файл із даними про класи та папки. Даний файл має мати таку структуру:

train: шлях/до/набору/даних/train.txt
val: шлях/до/набору/даних/val.txt

nc: кількість_класів
names: [клас_1, клас_2, ..., клас_nc]

Файл data.yaml для даної роботи, відповідно до структури, наведеної вище виглядає так:

```
path: /kaggle/working/data
train: /kaggle/working/data/train
val: /kaggle/working/data/test
nc: 9
names:
  0: walk
  1: push
  2: block50
  3: block75
  4: stand
  5: block25
  6: watchphone
  7: riding
  8: sit
```

Як видно зі схеми, наведеної вище, для тренування моделі було використано два набори із даними, що включали дані для тренування та валідації. Модель була натренована на 9 класах, які включали в себе людей, що йдуть, штовхаються, стоять, дивляться в телефони, їдуть, сидять та інше.

3.5 Тренування моделі

Тренування моделі відбувалось протягом трьох епох. Оскільки зображення для набору мають розміри 4К та їх кількість є доволі значною, додаткових тестувань на кількість оптимальної кількості епох не проводилось. За допомогою вбудованих GPU на Kaggle тренування вдалось виконати у хмарі, швидше ніж на власному девайсі та не перевантажуючи власний комп'ютер. Таблиця із результатами навчання виглядає так:

Таблиця 3.2 - Результати тренування моделі

epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
0	1.9371	3.9021	1.4144	0.81632	0.14433	0.61004	0.26059	1.7639	4.2003	1.3285	0.070323	0.003295	0.003295
1	1.8068	2.0226	1.3909	0.78823	0.7226	0.76159	0.33305	1.864	2.2885	1.5328	0.038134	0.004447	0.004447
2	1.7422	1.7542	1.3509	0.78004	0.74087	0.74862	0.3259	1.8216	1.8807	1.4815	0.037462	0.003387	0.003387

Крім того, була створена візуалізація таблиці із використанням графіків:

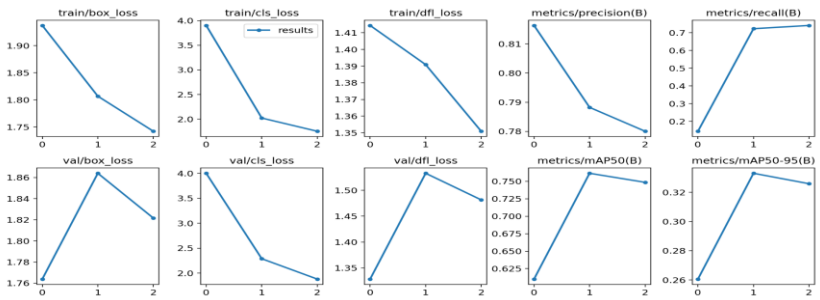


Рисунок 3.10 - Візуалізація таблиці

Як видно із таблиці та графіків наведених вище, кількість тренування в три епохи є цілком оптимальною, оскільки графіки точності почали зменшуватись після 2 епохи.

3.6 Оцінка результатів роботи моделі

В результаті тренування моделі було створено декілька зображень, що демострували здатність моделі виявляти людей на зображеннях:

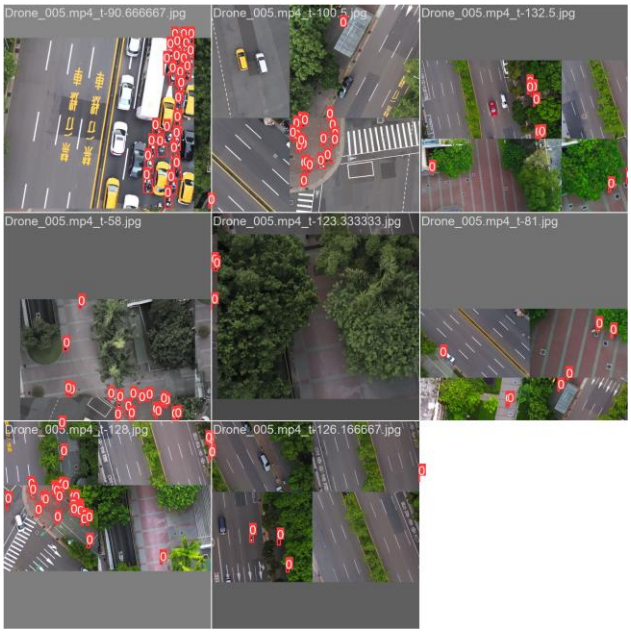


Рисунок 3.11 - Результати тренування Batch0

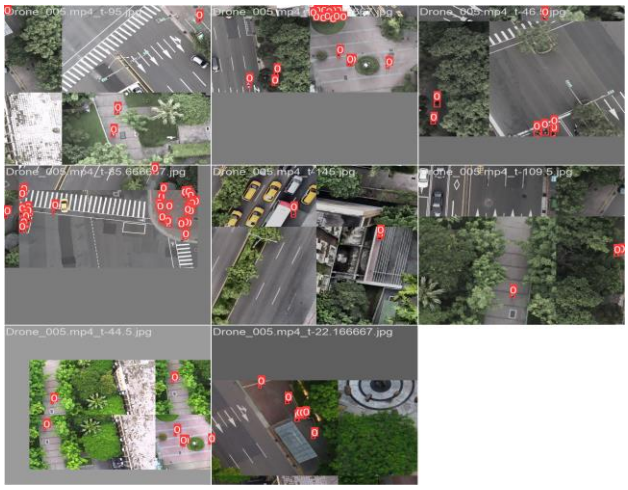


Рисунок 3.12 - Результати тренування Batch1

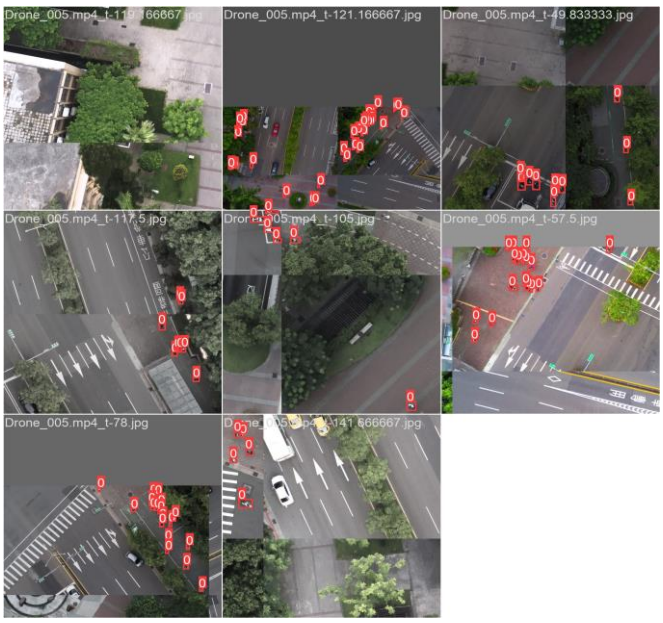


Рисунок 3.11 - Результати тренування Batch2

Як видно із зображень вище, модель доволі гарно справляється із задачею виявлення людей на зображеннях. Враховуючи маленькі розміри людей на вказаних кадрах, такі результати є дійсно високої якості. Проте, слід зазначити, що наведені вище картинки є результатом роботи моделі на тренувальних даних. Для візуальної перевірки моделі на валідаційних даних було створено ще три набори картинок:

Рисунок 3.12 - Правдиві мітки даних на val_batch2

Рисунок 3.13 - Передбачені мітки даних на val_batch2

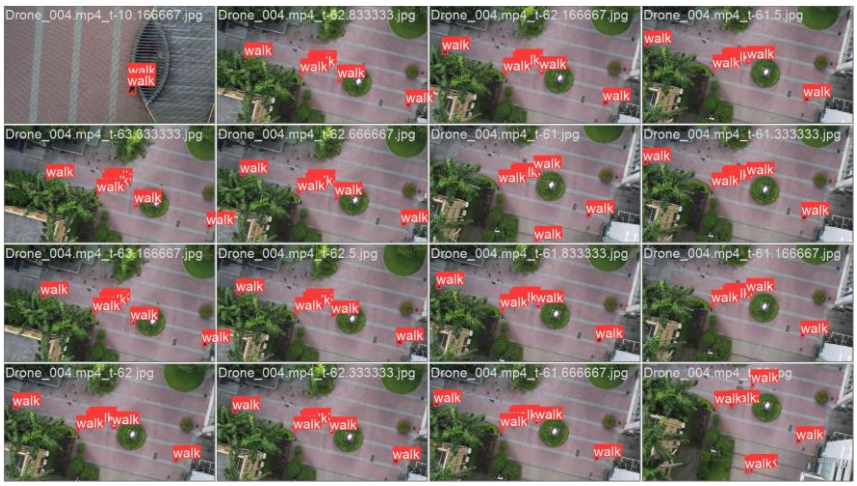


Рисунок 3.13 - Правдиві мітки даних на val_batch1

Рисунок 3.14 - Передбачені мітки даних на val_batch1

Як видно із утворених картинок, модель доволі успішно справляється із виявленням людей, навіть із таких значних відстаней. Проте, присутні певні

проблеми із виявленнями зайвих об’єктів і маркуванням їх за людей. Це може бути вирішене додатковим тренуванням моделі на нових типах даних.

3.7 Поєднання алгоритму із моделлю

Отже, в результаті розробки було утворено модель, що здатна визначати розташування людей із камери дрона. Наступним кроком до реалізації системи є розробка алгоритму, що буде визначати відстань до визначеної людини.

В алгоритмі будуть використовуватись дані про ширину рамки навколо людини, реальних розмірів промаркованої області та фокусної відстані дрона. Фокусну відстань в камері не можна змінити, тому вона є завжди сталою. Ширина рамки навколо людини вимірюється в пікселях, що буде нашою змінною. Реальний розмір виділеної області становить від 40 до 35 см. Ці гіперпараметри в подальшому можуть бути налаштовані під конкретний пристрій зчитування зображення, також реальні розміри об’єкта можуть бути ще одним із параметрів нейромережі, що дозволило б повністю автоматизувати процес визначення відстані до об’єктів за наявності відповідних даних.

Відстань до об’єкта обраховувалась із використанням формули (3.1):

$$distance = (focal\_length * object\_width) / (w * img\_w) \quad 3.1$$

де $focal\_length$ - фокусна відстань пристрою, $object\_width$ - реальні розміри виділеної області, w - кількість пікселів виділеної області(xmax-xmin), img_w - кількість пікселів всієї картинки.

Дана формула дозволяє використовуючи вхідні дані із нейромережі та дані про об’єкти для визначення та дані про камеру визначати відстань між камерою та об’єктом.

Розроблений функціонал включав в себе функцію, що приймала на вхід зображення та дані про об’єкт та камеру, проводила виявлення об’єктів на відео, проводила обрахунки та виводила результат.

3.8 Оцінка результатів роботи системи

В результаті розробки було отримано систему, що має змогу проводити визначення відстані до людей, використовуючи кадри відео зняті із дронів в польоті.

Модель була протестована на тренувальних даних, деякі параметри накшталт фокусної відстані, що не була відома з початку, оскільки в датасеті не була вказана модель дрону, були підібрані згодом методом спроб та помилок.

Результати роботи моделі виглядають таким чином:

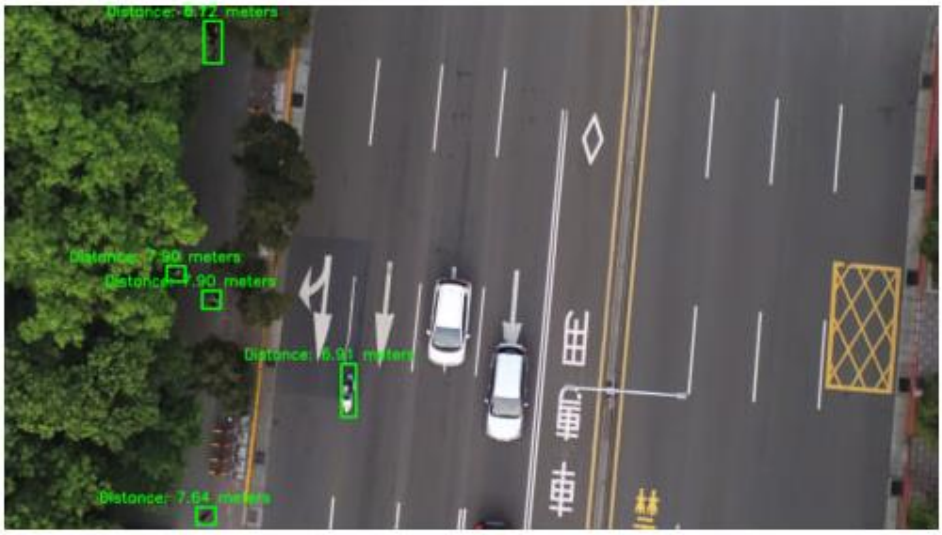


Рисунок 3.15 - Результати роботи розробленої моделі

Як видно із рисунку, наведеного вище, модель непогано справляється із завданням визначення об’єктів на відео, проте точність може значним чином залежати від точності визначення рамок навколо об’єкту на зображенні. Для покращення роботи слід додати нові дані для моделі та потенційно використовувати методи сегментації для більш точного визначення піксельного розміру об’єкта. Також можна використовувати нейромережі, що будуть додатково передбачати розміри об’єктів, що може значним чином покращити результати роботи.

3.9 Розробка інтерфейсу

В даній роботі для реалізації зручного інтерфейсу для взаємодії із моделлю було використано бібліотеку Flask для Python. Можливості даної бібліотеки були описані в попередніх розділах, було наведено декілька джерел, що демонстрували можливості даного інструмента для роботи із нейромережами та здатність та зручність інтегрування із нейромережами, що забезпечує Flask.

Для користувача було створено дві сторінки:

- Сторіка що забезпечує можливість обрати файл та завантажити його, тим самим викликавши роботу нейромережі
- Сторінка виведення результатів, що дозволяла завантажити відео користувачу для перегляду результатів

Розроблені сторінки мають такий вигляд:

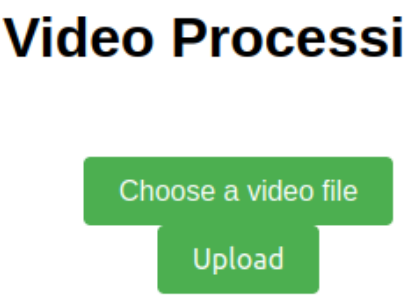


Рисунок 3.16 - Панель взаємодії із системою на першій сторінці

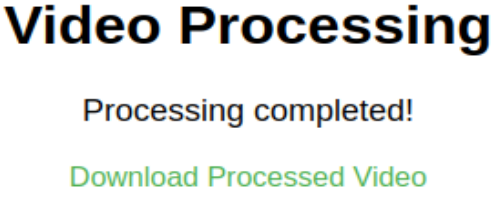


Рисунок 3.17 - Сторінка виведення результатів моделі

Як видно із рисунків вище, інтерфейс має доволі мінімалістичний дизайн, проте безперебойно забезпечує зручну роботу із розробленою системою, що може бути використано для тестування роботи та попереднього перегляду результатів.

Код, що забезпечує взаємодію із інтерфейсом завантажує дані в папку uploads, потім запускає в модель кадри поштучно та записує їх у відео в папці outputs. Директорія weights має ваги для моделі, щоб забезпечити зручну роботу із налаштування системи. (Рис. 3.18)

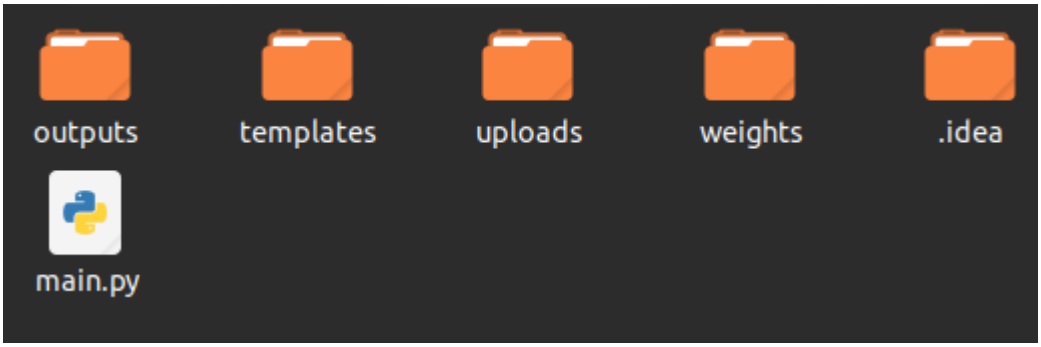


Рисунок 3.18 - Директорія із проєктом

В результаті завантаження відео та запуску моделі, відбувається покадрова обробка відео. На звичайному CPU із ноутбука на один кадр займає в середньому одну секунду, що є доволі довгим проміжком, враховуючи швидкість роботи алгоритму YOLO. Проте, кадри, що обробляє модель мають розміри 4К, що значно ускладнює необхідні обчислювальні ресурси для моделі. За використання GPU швидкість обробки відео може бути пришвидшена потенційно до 30 кадрів на секунду.

Модель була протестованана через інтерфейс, використовуючи відео з дрону із відкритих джерел. (Рис. 3.19-20)

Рисунок 3.19 - Результати роботи моделі на нових даних

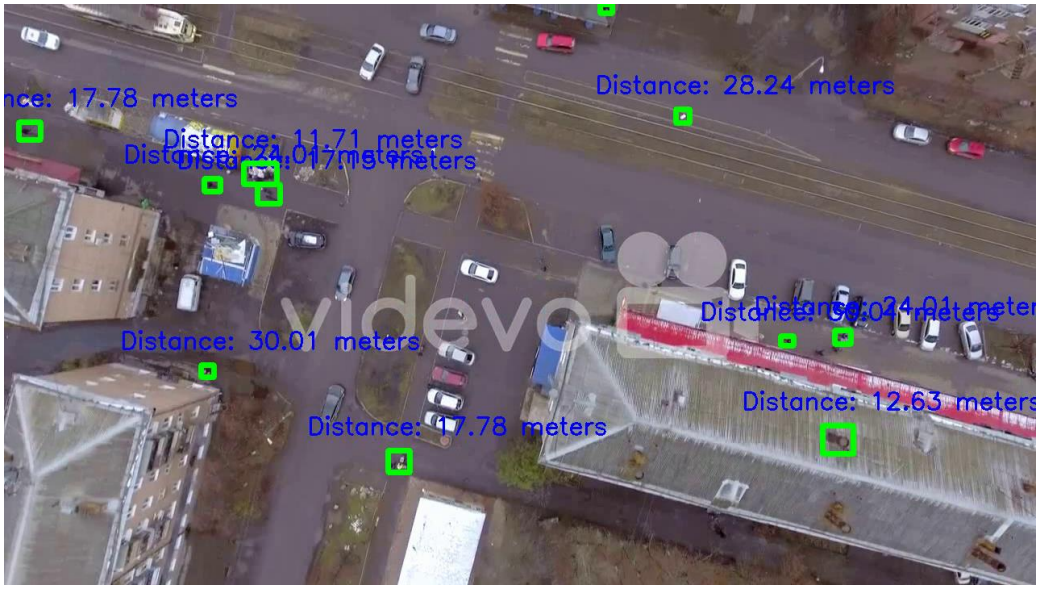


Рисунок 3.20 - Результати роботи моделі на нових даних

Як видно із зображень вище, модель доволі якісно розпізнає людей, проте вона допускає помилки, класифікуючи зайві об’єкти типу димарів як людей, що є наслідком малої різноманітності даних для тренування. Ці похибки можуть бути виправлені через донавчання моделі на нових типах даних.

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі було проведено ретельний огляд системи YOLO, що дозволяє ефективно виконувати об'єктний аналіз зображень в режимі реального часу. З використанням YOLO було розроблено систему для визначення відстані до об'єктів, зокрема людей, на зображеннях отриманих з дронів.

У процесі обробки та вибору даних було використано різні техніки для покращення точності розпізнавання об'єктів. Було враховано фактори, такі як різні розміри об'єктів, освітлення, та інші аномалії, що можуть вплинути на точність моделі.

Розроблена модель була тренована з використанням розмічених даних, що містять зображення людей в різних позах та на різних відстанях від камери. Тренування моделі включало ітеративний процес зміни параметрів, вибірки даних та оптимізації алгоритму, з метою досягнення оптимальної точності.

Оцінка роботи моделі показала її багатообіцяючий потенціал. Вона продемонструвала задовільні результати на 4к зображеннях, отриманих з дронів, здатна була розпізнавати людей з високою точністю. Проте, іноді модель сплутувала людей з об'єктами, які раніше не спостерігала, наприклад, димарів. Це може бути пов'язано з обмеженими даними для тренування, що не містили достатньо варіацій таких об'єктів.

Для подальшого вдосконалення системи рекомендується розширити набір тренувальних даних, включивши більше прикладів із димарями та іншими дрібними об'єктами, що можуть бути сплутані з людьми.

У результатах дослідження видно потенціал розробленої моделі та веб-інтерфейсу для взаємодії із нею. З вдосконаленням алгоритмів та додатковими навчальними даними, система може знайти широке застосування у різних сферах.

ВИСНОВКИ

Дипломна робота присвячена проблемі вимірювання відстані до об'єктів на відео. У ході дослідження було проведено огляд різноманітних методів, алгоритмів та систем, що використовуються для вимірювання відстані. Було вивчено та проаналізовано поняття відстані, методи вимірювання відстані на відео, а також існуючі системи та методи, які застосовуються для цієї задачі.

У розділі 1 проведено вивчення основних понять і термінології, пов'язаної з визначенням відстані, а також оглянуто існуючі методи вимірювання відстані на відео та їх характеристики.

У розділі 2 було детально досліджено різні аспекти, методи, алгоритми та технології, пов'язані з системою вимірювання відстані до об'єктів на відео. Було оглянуто функціональні можливості бібліотеки OpenCV, вивчена архітектура мережі YOLOv8 для обробки відео, проведено дослідження методів побудови датасетів, оптимізації та покращення точності визначення відстані, а також досліджено методи підтримки множини камер та автоматичної калібрування. Крім того, вивчено можливості розробки веб-додатків з використанням штучного інтелекту для розпізнавання об'єктів на відео.

У розділі 3 проведено розробку системи вимірювання відстані до об'єктів на відео. Було проведено обробку та вибір даних, тренування моделі з використанням алгоритму YOLOv8, оцінка її роботи, розробка алгоритму визначення відстані та розробка веб-інтерфейсу для взаємодії із моделлю. В результаті роботи була отримана багатообіцяюча модель, яка працювала на 4K зображеннях з дронів та була здатна розпізнавати людей. Однак, було виявлено проблему - іноді модель плутала людей з об'єктами, наприклад, димарями, що не спостерігались при навчанні раніше. Це може бути пов'язано з обмеженими даними для тренування.

Загалом, дипломна робота дозволила вивчити та проаналізувати різноманітні аспекти визначення відстані до об'єктів на відео. Розроблені

система та модель показали свій потенціал, але потребують подальшого вдосконалення. Результати дослідження можуть бути використані в різних сферах, таких як системи спостереження, безпека, військова сфера та інші, де точне визначення відстані до об'єктів, або їх координат має важливе значення.

Завдання на бакалаврську дипломну роботу було виконано у повному обсязі відповідно до усіх вимог згідно стандартів та правил оформлення. Проєкт виконує поставлені задачі та працює без помилок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Distance. From Wikipedia, the free encyclopedia. Режим доступу: https://en.wikipedia.org/wiki/Distance (Дата звернення 06.05.2023)
2. Elena Serea, Mihai Penciuc, Marinel Costel Temneanu and Codrin Donciu. Video Distance Measurement Technique Using Least Squares Based Sharpness Cost Function. ежим доступу: https://www.mdpi.com/2227-7390/10/18/3273 (Дата звернення 06.05.2023)
3. Harasim Eugen, Codrin Donciu. Video distance measurement based on focus. 2014. Режим доступу: https://ieeexplore.ieee.org/document/9305593 (Дата звернення 06.05.2023)
4. Wei Zhou, Youfeng Zhang, Unhong Zheng, Pingfan Li. Research on distance measurement method of moving objects in video. Режим доступу: https://www.researchgate.net/publication/340312912_Research_on_distance_measurement_method_of_moving_objects_in_video (Дата звернення 06.05.2023)
5. Abdelmoghit Zaarane, Ibtissam Slimani, Wahban Al Okaishi, Issam Atouf, Abdellatif Hamdoun. Distance measurement system for autonomous vehicles using stereo camera. March 2020. Режим доступу: https://www.researchgate.net/publication/340312912_Research_on_distance_measurement_method_of_moving_objects_in_video (Дата звернення 06.05.2023)
6. Ciprian Dragne, Isabela Todirite, Marius Pandelea, Mihaiela Iliescu. Distance Assessment by Object Detection-For Visually Impaired Assistive Mechatronic System. June 2022. Режим доступу: https://www.researchgate.net/publication/361463112_Distance_Assessment_by_Object_Detection-For_Visually_Impaired_Assistive_Mechatronic_System (Дата звернення 06.05.2023)

7. Doha 2010 - Introducing Video Distance Measurement. *World Athletics*. Режим доступу: https://worldathletics.org/news/news/doha-2010-introducing-video-distance-measur (Дата звернення 06.05.2023)
8. Computer Vision: Determining the Distance From an Object in a Video. *Baeldung*. March 29, 2023. Режим доступу: https://www.baeldung.com/cs/cv-compute-distance-from-object-video (Дата звернення 06.05.2023)
9. Gaudenz Boesch. What is OpenCV? The Complete Guide (2023). Режим доступу: https://viso.ai/computer-vision/opencv/ (Дата звернення 06.05.2023)
10. OpenCV: Introduction. *OpenCV Documentation*. Режим доступу: https://docs.opencv.org/4.x/d1/dfb/intro.html (Дата звернення 06.05.2023)
11. Akshit Mehra. Understanding YOLOv8 Architecture, Applications & Features. Apr 16, 2023. Режим доступу: https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/ (Дата звернення 06.05.2023)
12. YOLOv8 for Object Detection Explained [Practical Example]. *Encord*. Mar 22, 2023. Режим доступу: https://medium.com/cord-tech/yolov8-for-object-detection-explained-practical-example-23920f77f66a (Дата звернення 06.05.2023)
13. Introducing Ultralytics YOLOv8. *Ultralytics*. Режим доступу: https://docs.ultralytics.com/ (Дата звернення 06.05.2023)
14. Piotr Skalski. How to Train YOLOv8 Object Detection on a Custom Dataset. JAN 10, 2023. Режим доступу: https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/ (Дата звернення 06.05.2023)
15. Sovit Rath. Train YOLOv8 on Custom Dataset – A Complete Tutorial. Jan 31, 2023. Режим доступу: https://learnopencv.com/train-yolov8-on-custom-dataset/ (Дата звернення 06.05.2023)
16. Train - Ultralytics YOLOv8. *Ultralitycs*. Режим доступу: https://docs.ultralytics.com/modes/train/ (Дата звернення 06.05.2023)

17. Haidi Zhu, Haoran Wei, Baoqing Li, Xiaobing Yuan and Nasser Kehtarnavaz. A Review of Video Object Detection: Datasets, Metrics and Methods. 4 November 2020. Режим доступу: https://www.mdpi.com/2076-3417/10/21/7834 (Дата звернення 06.05.2023)
18. Jaskirat Kaur & Williamjeet Singh. Tools, techniques, datasets and application areas for object detection in an image: a review. 23 April 2022. Режим доступу: https://link.springer.com/article/10.1007/s11042-022-13153-y (Дата звернення 06.05.2023)
19. Top Object Recognition Video Datasets of 2022. *Twine AI*. Режим доступу: https://www.twine.net/blog/top-object-recognition-video-datasets/ (Дата звернення 06.05.2023)
20. Gousia Habib, Shaima Quresh. Optimization and acceleration of convolutional neural networks: A survey. July 2022. Режим доступу: https://www.sciencedirect.com/science/article/pii/S1319157820304845 (Дата звернення 06.05.2023)
21. J.C. Rodríguez-Quiñonez, O. Sergiyenko, W. Flores-Fuentes, M. Rivas-lopez, D. Hernandez-Balbuena, R. Rascón, P. Mercorelli. Improve a 3D distance measurement accuracy in stereo vision systems using optimization methods' approach. May 2017. Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S1230340217300100 (Дата звернення 06.05.2023)
22. Yong Ye, Yuting Liu, Weihan Yin, Jiahao Deng & Xiaofeng Zhu. Improving the measurement accuracy of distance and positioning for capacitive proximity detection in human-robot interaction. 19 April 2021. Режим доступу: Режим доступу: https://link.springer.com/article/10.1007/s00542-021-05223-2%D1%84 (Дата звернення 06.05.2023)
23. Zewei Liu, Dongming Lu, Weixian Qian, Kan Ren, Guohua Gu, Jun Zhang & Chen Mao. A new method for increasing accuracy of distance measurement based on single visual camera. 21 February 2019. Режим доступу:

https://link.springer.com/article/10.1007/s11082-019-1786-z (Дата звернення 06.05.2023)

24. Realtime Distance Estimation Using OpenCV – Python. *GeeksForGeeks*. Режим доступу: https://www.geeksforgeeks.org/realtime-distance-estimation-using-opencv-python/ (Дата звернення 06.05.2023)
25. Measuring distance between two contours in video? OpenCV Python. *StackOverflow*. Режим доступу: https://stackoverflow.com/questions/41832237/measuring-distance-between-two-contours-in-video-opencv-python (Дата звернення 06.05.2023)
26. Haseeb Muhammad Abdul, Ristić-Durrant Danijela, Gräser Axel, Banić Milan & Stamenković Dušan. Multi-DisNet: Machine Learning-Based Object Distance Estimation from Multiple Cameras. 23 November 2019. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-030-34995-0_41 (Дата звернення 06.05.2023)
27. Eric J. Howe, Stephen T. Buckland, Marie-Lyne Després-Einspenner, Hjalmar S. Kühl. Distance sampling with camera traps. 10 May 2017. Режим доступу: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.12790 (Дата звернення 06.05.2023)
28. Amelia Azman, M. Hanafi Ani. Object Distance and Size Measurement Using Stereo Vision System. December 2012. Режим доступу: https://www.researchgate.net/publication/274773924_Object_Distance_and_Size_Measurement_Using_Stereo_Vision_System (Дата звернення 06.05.2023)
29. Apar Garg. Stereo Vision: Depth Estimation between object and camera. Nov 17, 2021. Режим доступу: https://www.researchgate.net/publication/274773924_Object_Distance_and_Size_Measurement_Using_Stereo_Vision_System (Дата звернення 06.05.2023)

30. Paul Pias. Object-Detection-and-Distance-Measurement. *Github*. Режим доступу: https://github.com/paul-pias/Object-Detection-and-Distance-Measurement (Дата звернення 06.05.2023)
31. NEDA CVIJETIC. To Go the Distance, We Built Systems That Could Better Perceive It
32. See how deep neural networks trained on radar and lidar data predict 3D distances from 2D images. *Nvidia DRIVE Labs*. Режим доступу: https://blogs.nvidia.com/blog/2019/06/19/drive-labs-distance-to-object-detection/ (Дата звернення 06.05.2023)
33. Hans-Gerd Maas. Image sequence based automatic multi-camera system calibration techniques. December 1999. Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0924271699000295 (Дата звернення 07.05.2023)
34. Zhen Liu, Fengjiao Li, Guangjun Zhang. An external parameter calibration method for multiple cameras based on laser rangefinder. January 2014. Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0263224113005289 (Дата звернення 07.05.2023)
35. Adrian Rosebrock. Find distance from camera to object/marker using Python and OpenCV. January 19, 2015. Режим доступу: https://pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/ (Дата звернення 07.05.2023)
36. Flask. *PyPi*. Режим доступу: https://pypi.org/project/Flask/ (Дата звернення 07.05.2023)
37. Nakul Lakhotia. Video Streaming in Web Browsers with OpenCV & Flask. Oct 20, 2020. Режим доступу: https://towardsdatascience.com/video-streaming-in-web-browsers-with-opencv-flask-93a38846fe00 (Дата звернення 07.05.2023)

38. Mark Winterbottom, Brooke Rutherford. Flask vs. Django: Which Web Framework is Best? Режим доступу: https://blog.udemy.com/flask-vs-django/ (Дата звернення 07.05.2023)
39. Daniel Diaz. 25 Python Frameworks to Master in 2023. May 3, 2023. Режим доступу: https://kinsta.com/blog/python-frameworks/ (Дата звернення 07.05.2023)
40. Irfan Alghani Khalid. How to Display Video Streaming From A Webcam Using Flask. Aug 30, 2021. Режим доступу: https://towardsdatascience.com/how-to-display-video-streaming-from-a-webcam-using-flask-7a15e26fbab8 (Дата звернення 07.05.2023)
41. Gerry Christian Ongko. Building a Machine Learning Web Application Using Flask. Feb 3, 2022. Режим доступу: https://towardsdatascience.com/building-a-machine-learning-web-application-using-flask-29fa9ea11dac (Дата звернення 07.05.2023)
42. Diane Phan. What's Cookin'? Build an Image Recognition App on WhatsApp using Twilio MMS, Clarifai API, Python, and Flask 07 oct, 2020. Режим доступу: https://www.twilio.com/blog/image-recognition-whatsapp-mms-python-flask-clarifai (Дата звернення 07.05.2023)
43. Sachin Pal. Build Flask App For Image Recognition Using Deep Learning Model. Sachin Pal. Nov 17, 2022. Режим доступу: https://geekpython.in/flask-app-for-image-recognition (Дата звернення 07.05.2023)

ДОДАТОК 1

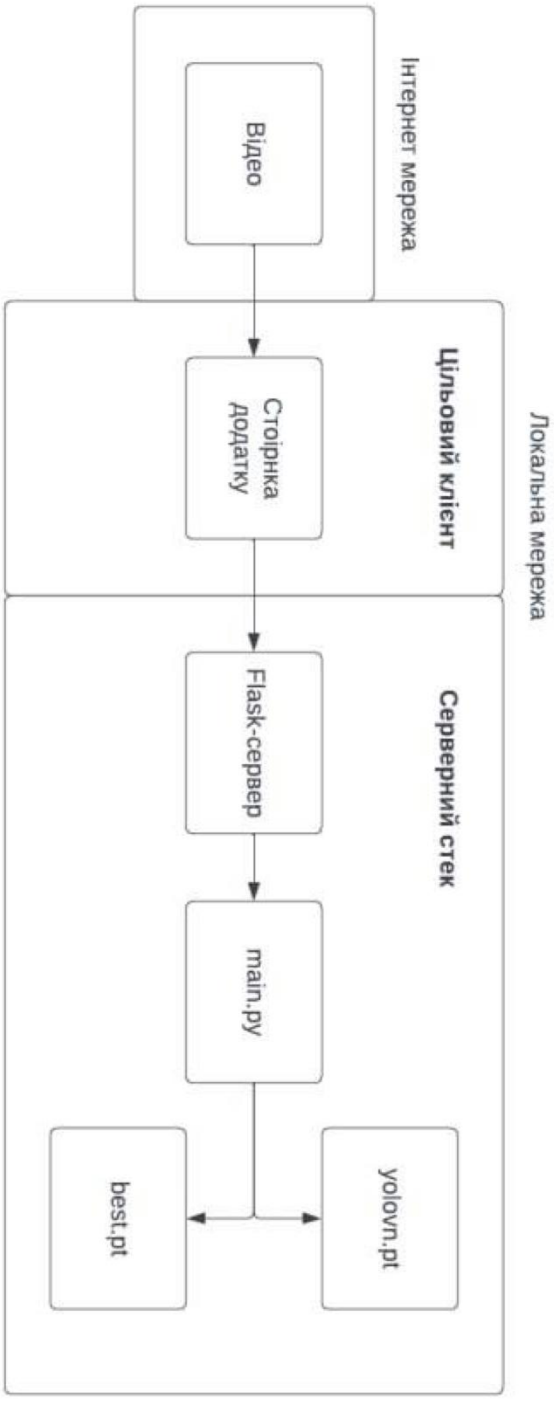
Система вимірювання відстані до об’єктів на відео

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ – 2023 р.



					ІАЛЦ.467200.004 Д1	
Зм.	Арк.	№ докум.	Підп.	Дата		
Розроб.		Цоколов М.В.			*Система вимірювання відстані до об'єктів на відео* Пристрій вимірювання відстані (Структурна схема)	Літ. / Арк. 1 / Аркушів 1
Перевір.		Алєнін О.І.				
Н. контр.						***КПІ ім. Ігоря Сікорського ФІОТ ІВ-93***
Затверд.						

ДОДАТОК 2

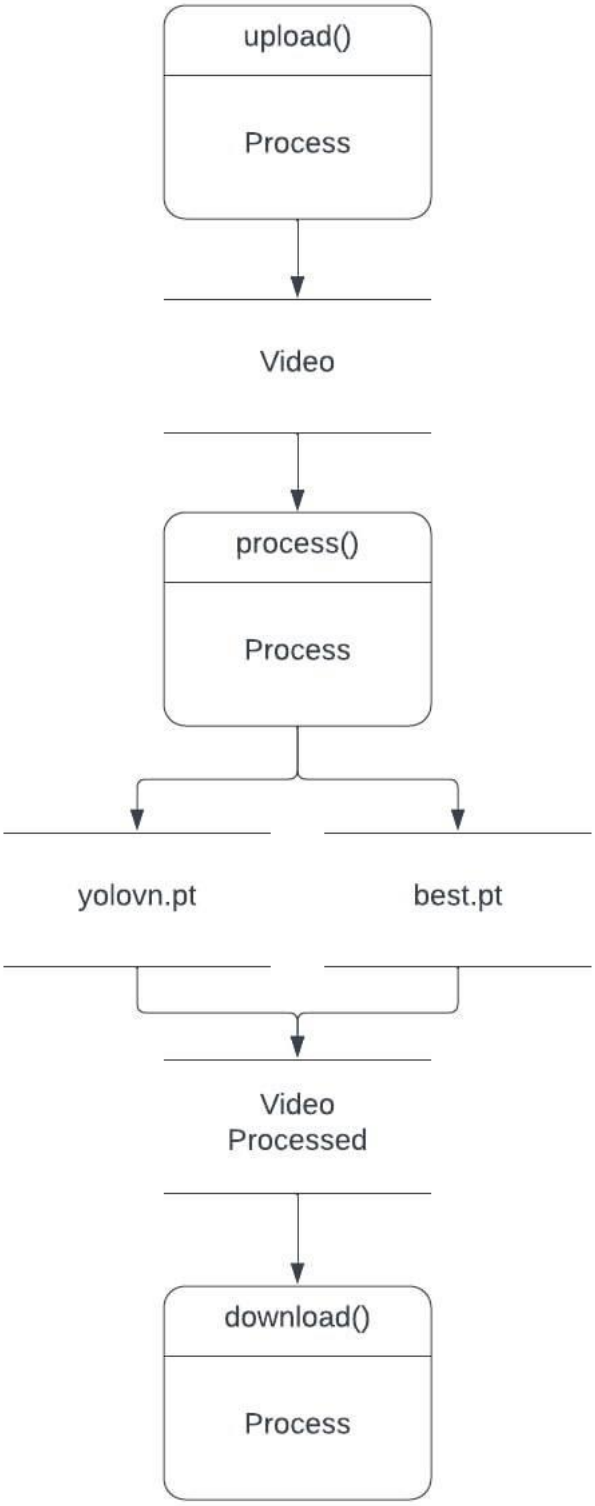
Система вимірювання відстані до об’єктів на відео

Функціональна схема

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ – 2023 р.



					ІАЛЦ.467200.005 Д2	
Зм.	Арк.	№ докум.	Підп.	Дата		
Розроб.		Цоколов М.В.			*Система вимірювання відстані до об'єктів на відео* Застосунок вимірювання відстані (Функціональна схема)	Літ. / Арк. 1 / Аркушів 1
Перевір.		Алєнін О.І.				
Н. контр.						***КПІ ім. Ігоря Сікорського ФІОТ ІВ-93***
Затверд.						

ДОДАТОК З

Система вимірювання відстані до об’єктів на відео

Принципова схема

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ – 2023 р.

					ІАЛЦ.467200.006 Д3		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Цоколов М.В.			*Система вимірювання відстані до об'єктів на відео* Алгоритм дії системи вимірювання відстані (Принципова схема)	Літ.	Арк. 1 / Аркушів 1
Перевір.		Алєнін О.І.					
Н. контр.						***КПІ ім. Ігоря Сікорського ФІОТ ІВ-93***	
Затверд.							

ДОДАТОК 4

Система вимірювання відстані до об’єктів на відео

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 2

Київ – 2023 р.

```
from flask import Flask, render_template, request, send_file
from werkzeug.utils import secure_filename
import os
from ultralytics import YOLO
import cv2
import shutil


app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
app.config['OUTPUT_FOLDER'] = 'outputs'
app.config['ALLOWED_EXTENSIONS'] = {'mp4', 'pt'}  # Specify allowed file
extensions

yolo_weights = ''  # Global variable to store the selected YOLO weights file
path


def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
app.config['ALLOWED_EXTENSIONS']


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/upload', methods=['POST'])
def upload():
    global yolo_weights

    file = request.files['video']
    yolo_weights_file = request.form['weights']

    if file and allowed_file(file.filename) and yolo_weights_file:
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

        yolo_weights = yolo_weights_file
        print(filename, yolo_weights)
        filename = process_video(filename)
        return render_template('download.html',
filename=os.path.basename(filename))
    else:
        return "Invalid file format. Please upload an MP4 video and choose a
YOLO weights file."


def process_video(video_path):
    model = YOLO(f'weights/{yolo_weights}')

    print(video_path)
    video = cv2.VideoCapture('uploads/' + video_path)
```

```
    frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = video.get(cv2.CAP_PROP_FPS)

    output_path = f'outputs/processed_{video_path}'
    output_video = cv2.VideoWriter(output_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_width, frame_height))

    while True:
        ret, frame = video.read()

        if not ret:
            break

        print('read')

        results = model(frame)[0]
        print('framed')
        for box in results.boxes.xyxy:
            x1, y1, x2, y2 = map(int, box)

            # Calculate the distance to the object
            focal_length = 1200.29  # Focal length of the camera (in mm)
            object_width = 0.4  # Width of the object (in meters)
            distance = (focal_length * object_width) / (x2 - x1)

            # Draw the rectangle
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 5)
            cv2.putText(frame, f"Distance: {distance:.2f} meters", (x1-100,
y1-20), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

        output_video.write(frame)

    if os.path.exists(output_path):
        return output_path
    else:
        print(video_path, output_path)
        shutil.copy2('uploads/' + video_path, output_path)
        return output_path

@app.route('/download/<filename>')
def download(filename):
    file_path = os.path.join(app.config['OUTPUT_FOLDER'], filename)
    if os.path.exists(file_path):
        return send_file(file_path, as_attachment=True)
    else:
        return "File not found."


if __name__ == '__main__':
    app.run(debug=True)
```