Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра обчислювальної техніки
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

МЕТОД І ЗАСОБИ ВІДСЛІДКОВУВАННЯ ОБ'ЄКТІВ НА ВІДЕО В РЕАЛЬНОМУ ЧАСІ

Виконав: студент 2–го курсу, групи 2КІ–23м
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Кушнір В.А.
(прізвище та ініціали)

Керівник: к. пед. н., доц. каф..
Добровольська Н.В.
(прізвище та ініціали)
«15» 12 2024 р.

Опонент: д.т.н., професор каф. ПЗ
Коваленко О.О.
(прізвище та ініціали)
«16» 12 2024 р.

Допущено до захисту
Завідувач кафедри
д. т. н., проф. Азаров О. Д.
(прізвище та ініціали)
«17» 12 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра комп'ютерної інженерії
Рівень вищої освіти ІІ–й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 123 «Комп'ютерна інженерія»
Освітньо–професійна програма – «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф. Азаров О. Д.
«18» вересня 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ

Кушніру Володимиру Анатолійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи «Метод і засоби відслідковування об'єктів на відео в реальному часі» керівник роботи к. пед. н., доц. каф., Добровольська Н.В. затверджено наказом ВНТУ від «17 09 2024 року № 310
2. Строк подання студентом роботи 12 12 2024 року
3. Вихідні дані до роботи: мова програмування – об'єктно орієнтована, інтерфейс – інтуїтивно зрозумілий, кількість одночасно відслідковуваних об'єктів – не менше 3, кількісь кадрів на секунду у відеопотоці – від 20
4. Зміст текстової частини:

Вступ, аналіз предметної області відслідковування об'єктів в реальному часі, аналіз предметної області, проєктування інформаційних технологій відслідковування об'єктів, програмна реалізація засобів відслідковування об'єктів на відео в реальному часі, висновки, перелік використаних джерел, додатки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): Схема алгоритму загального функціонування методів відслідковування об'єктів, UML–діаграма класів основних компонентів програми.

6. Календарний план виконання МКР приведений в таблиці 2.

Таблиця 1 – Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Виконання прийняв
1-4	Добровольська Н.В. к. пед. н., доц. каф		
5	Небава М. І. д.т.н., проф.		

7. Дата видачі завдання 18 09 2024 року

8. Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 – Календарний план

тапу	Назва етапу	Строк виконання етапів роботи	При– мітка
	Аналіз предметної області відслідковування об’єктів в реальному часі	20.09.2024	Виконано
	Проєктування інформаційних технологій відслідковування об'єктів	03.10.2024	Виконано
	Програмна реалізація засобів відслідковування об’єктів на відео в реальному часі	27.10.2024	Виконан
	Тестування та аналіз результатів роботи розробленої програми	20.11.2024	Виконано
	Розробка інструкції користувача	01.12.2024	Виконано
	Оформлення матеріалів до захисту МКР	15.12.2024	Виконано

Студент _____ Кушнір В.А.
(підпис)

Керівник роботи _____ Добровольська Н.В.
(підпис)

АНОТАЦІЯ

УДК 004.051

Кушнір В.А. Система відслідковування об'єктів на відео в реальному часі. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2024 — 107 с. На укр. мові. Бібліогр.: 22 назв; рис.: 15; табл. 15.

У роботі розглянуто принципи побудови системи відслідковування об'єктів на відео в реальному часі з використанням сучасних алгоритмів комп'ютерного зору. Проведено аналіз існуючих підходів до виявлення та відстежування об'єктів, зокрема застосування алгоритмів YOLO для виявлення об'єктів та Kalman Filter для їхнього відстеження. Окремо розглянуто проблему динамічної адаптації до змінюваних умов середовища, таких як рівень освітленості та наявність перешкод. Розроблено удосконалену систему відслідковування, яка забезпечує високу точність та стабільність роботи в реальних умовах. У роботі також представлено алгоритм для інтеграції цих технологій, а також структурну та функціональну схему системи, що дозволяє відслідковувати об'єкти в реальному часі в складних умовах.

Ключові слова: відстеження об'єктів, штучний інтелект, YOLO, Kalman Filter.

ABSTRACT

УДК 004.051

Last Name First Name. Object Tracking System on Video in Real–Time. Master's thesis in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2024. — 107 p. In Ukrainian language. Bibliographer: 22 titles; fig.: 15; tabl. 15.

This thesis examines the principles of building an object tracking system for real–time video using modern computer vision algorithms. An analysis of existing approaches to object detection and tracking is provided, including the application of YOLO algorithms for object detection and Kalman Filter for tracking. The issue of dynamic adaptation to changing environmental conditions, such as lighting levels and the presence of obstacles, is discussed separately. An improved tracking system has been developed that ensures high accuracy and stability in real–world conditions. The thesis also presents an algorithm for integrating these technologies, as well as the structural and functional diagrams of the system, which allows for real–time object tracking in complex conditions.

Key words: object tracking, artificial intelligence, YOLO, Kalman Filter.

ЗМІСТ

ВСТУП

Актуальність теми полягає в тому, що відслідковування об’єктів на відео відіграє значну роль в сучасному світі, де технології постійно змінюються та вдосконалюються, відслідковування об'єктів у реальному часі стає важливим завданням для багатьох галузей, включаючи безпеку, логістику, транспорт та автоматизацію. Застосування сучасних алгоритмів та методів для аналізу відеоданих дозволяє покращити точність і швидкість відслідковування, що є критично важливим у реальних умовах. Унікальною проблемою, з якою стикаються подібні системи, є варіативність середовища, що включає зміну освітлення, перешкоди та непередбачуваність руху об'єктів.

Розробка програмного забезпечення, яке використовує методи комп'ютерного зору та машинного навчання, дозволяє створювати надійні системи, що працюють у режимі реального часу. Використання таких алгоритмів, як YOLO для виявлення об'єктів, та Kalman Filter для їхнього відстеження, забезпечує не лише високу точність, але й ефективність у складних умовах. Крім того, динамічна адаптація до зміни умов середовища, зокрема освітлення, є важливою складовою для забезпечення стабільності роботи системи.

Зважаючи на це, тема розробки програмного забезпечення для відслідковування об'єктів на відео в реальному часі є надзвичайно актуальною, оскільки вона сприяє автоматизації та вдосконаленню процесів у багатьох сферах діяльності.

Метою магістерської кваліфікаційної роботи є удосконалення, та покращення ефективності роботи системи відстеження об'єктів у реальному часі в умовах та з обмеженими обчислювальними можливостями.

Для досягнення поставленої мети необхідно сформулювати та виконати наступні **завдання:**

— провести аналіз предметної області та об'єкту дослідження;

— дослідити існуючі системи для відстеження об'єктів на основі відео, а також їх особливості та недоліки;

— розробити алгоритми для відстеження об'єктів, зокрема реалізацію алгоритмів на базі YOLO, Kalman Filter та інших;

— виконати програмну реалізацію системи відстеження об'єктів на основі вибраних алгоритмів;

— провести тестування розробленого програмного забезпечення та проаналізувати результати, зокрема точність і надійність відстеження;

— оцінити економічну ефективність розробленої системи, вивчаючи її потенційне застосування в різних сферах.

Об’єкт дослідження — процес відстеження об'єктів на відео в реальному часі за допомогою БПЛА.

Предмет дослідження — інформаційні технології та алгоритми для відстеження об'єктів у реальному часі.

Методи дослідження — У роботі використовуються такі методи наукових досліджень: графічний метод для візуалізації результатів відстеження; методи комп'ютерного зору, включаючи алгоритми для виявлення об'єктів; методи фільтрації, зокрема Kalman Filter, для прогнозування траєкторій руху об'єктів; а також методи об'єктно – орієнтованого програмування для реалізації системи.

Апробація результатів роботи була проведена на конференції «Молодь в науці: дослідження, проблеми, перспективи»

За результатами досліджень **опубліковано** тези доповіді на науково–технічній конференції [1].

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ВІДСЛІДКОВУВАННЯ ОБ’ЄКТІВ В РЕАЛЬНОМУ ЧАСІ

1.1 Обґрунтування доцільності використання методів і засобів відслідковування об'єктів на відео в реальному часі

Відслідковування об'єктів на відео в реальному часі є однією з ключових технологій комп'ютерного зору, що забезпечує автоматизацію багатьох процесів в різних галузях — від охорони і контролю доступу до військових операцій та екологічного моніторингу. Ця технологія дозволяє не тільки виявляти об'єкти на статичних зображеннях, але й відслідковувати їхній рух у динамічних відеопотоках, що є особливо важливим у випадках, коли об'єкти постійно змінюють свої позиції або умови середовища змінюються.

Сучасні методи відслідковування об'єктів на відео можна поділити на кілька основних категорій [2].

Методи на основі ключових точок та ознак — ці методи базуються на аналізі характерних ознак об'єктів, таких як текстура, контури та кути. Алгоритми, як–от SIFT (Scale–Invariant Feature Transform) та ORB (Oriented FAST and Rotated BRIEF), дозволяють виявляти унікальні точки на зображенні та відслідковувати їх між кадрами. Основною перевагою цих методів є висока точність у випадках, коли об'єкти мають добре визначені геометричні форми або текстури. Проте вони менш ефективні у випадках, коли об'єкти змінюються або перебувають у складних умовах освітлення.

Методи на основі шаблонів — ці методи використовують попередньо визначені шаблони об'єктів для їх відслідковування. Один із класичних підходів — використання кореляційних фільтрів, таких як MOSSE (Minimum Output Sum of Squared Error), які відстежують об'єкт шляхом порівняння шаблону із зображенням на кожному кадрі. Хоча цей метод є ефективним у реальному часі через низькі обчислювальні витрати, він може бути не досить точним у випадках, коли об'єкти змінюють свою форму або мають багато схожих сусідніх об'єктів.

Методи на основі глибинного навчання — глибинні нейронні мережі (DNN), особливо конволюційні нейронні мережі (CNN), стали основним

інструментом у відслідковуванні об'єктів на відео в реальному часі. Алгоритми, такі як YOLO (You Only Look Once), SSD (Single Shot Detector), та DeepSORT, дозволяють не тільки розпізнавати об'єкти, але й відслідковувати їхній рух на кожному кадрі відеопотоку. Особливістю цих методів є висока точність і можливість одночасного відстеження кількох об'єктів на різних відеопотоках. Важливо, що ці алгоритми здатні працювати навіть при складних умовах, таких як мінливе освітлення або часткова закритість об'єктів. Використання DNN вимагає значних обчислювальних ресурсів, але сучасні графічні процесори дозволяють досягти обробки відео у реальному часі.

Методи, засновані на фільтрації руху — одним із класичних підходів до відслідковування є Kalman Filter, який використовується для передбачення майбутнього положення об'єкта на основі його попереднього руху. Для більш складних траєкторій або у випадках, коли об'єкт змінює швидкість або напрямок руху, застосовуються фільтри на основі Байєсових мереж або Particle Filter (Фільтр частинок), які можуть передбачати поведінку об'єкта в умовах невизначеності.

Для реалізації відслідковування об'єктів використовуються різні програмні інструменти та платформи:

OpenCV (Open Source Computer Vision Library) — це одна з найпоширеніших бібліотек, що використовується для комп'ютерного зору та аналізу відео. Вона надає широкий спектр інструментів для виявлення і відстежування об'єктів, включаючи алгоритми для обробки зображень, виявлення об'єктів, а також алгоритми фільтрації руху [3].

TensorFlow та PyTorch — це бібліотеки для машинного навчання, що дозволяють використовувати нейронні мережі для відстежування об'єктів у реальному часі. Вони підтримують розробку та навчання моделей глибокого навчання, що дозволяє ефективно відстежувати рухи об'єктів на відеопотоках.

Dlib — це інша популярна бібліотека, яка містить алгоритми для машинного навчання та комп'ютерного зору, включаючи інструменти для відстежування об'єктів на основі шаблонів та ознак.

Використання цих методів і засобів для відслідковування об'єктів у реальному часі особливо актуальне в задачах, пов'язаних з безпілотними літальними апаратами (БПЛА). БПЛА використовуються в багатьох сферах, де відстежування об'єктів у реальному часі є критично важливим.

Один із найяскравіших прикладів доцільності використання БПЛА — це військові операції. У сучасних конфліктах БПЛА активно застосовуються для розвідки, коригування артилерійських ударів, виявлення ворожих позицій та моніторингу пересування сил противника. Особливо важливим є їхнє використання під час війни в Україні, де БПЛА масово застосовуються для захисту територій та збору інформації про ворога. Завдяки можливості відстежувати об'єкти з великої висоти та в умовах, де неможливо використовувати пілотовану авіацію, БПЛА стали невід'ємною частиною сучасних бойових операцій. Наприклад, українські сили оборони активно використовують БПЛА для моніторингу бойових зон, що дозволяє зменшити втрати серед військовослужбовців та підвищити ефективність планування і виконання операцій.

Окрім військових завдань, БПЛА ефективно використовуються для моніторингу природних катастроф, таких як лісові пожежі, повені, землетруси та виверження вулканів. У таких ситуаціях звичайні методи спостереження можуть бути занадто небезпечними або фізично неможливими. БПЛА можуть літати в складних умовах і передавати відео в реальному часі, що дозволяє рятувальним службам швидко оцінити масштаби катастрофи та координувати дії для мінімізації збитків. Наприклад, у 2020 році під час лісових пожеж у Каліфорнії, США, БПЛА активно використовувалися для моніторингу ситуації з повітря, передаючи дані в реальному часі про напрямок поширення вогню та зони найбільшої небезпеки.

Ще однією важливою сферою є використання БПЛА в промисловому та екологічному моніторингу. Завдяки своїй мобільності та здатності працювати в складних умовах, БПЛА можуть контролювати великі території, інфраструктурні об'єкти (наприклад, трубопроводи, електростанції, заводи) або

екосистеми (ліси, водоймища). Це дозволяє виявляти можливі порушення або аварії на ранніх стадіях, що сприяє їхньому швидкому вирішенню та мінімізації екологічної шкоди.

Однією з ключових переваг БПЛА є можливість працювати в режимі реального часу. Використання відеокамер високої роздільної здатності та систем комп'ютерного зору дозволяє БПЛА безперервно спостерігати за об'єктами, аналізувати їхній стан та пересування. Завдяки сучасним технологіям обробки даних, інформація може бути передана оператору миттєво або використовуватися для автономних рішень самого БПЛА. Це є особливо важливим у випадках, коли апарат втрачає зв'язок з пультом керування. У таких ситуаціях система повинна мати здатність самостійно продовжувати виконання завдання, базуючись на раніше отриманих даних або застосовуючи алгоритми штучного інтелекту для прийняття рішень.

Підсумовуючи, використання БПЛА для відслідковування об'єктів у реальному часі має численні переваги, які роблять цю технологію незамінною в багатьох сферах. Їхня здатність діяти автономно, навіть у складних умовах або при втраті зв'язку, забезпечує високу ефективність виконання місій та знижує ризики для людського життя. Тому дослідження і розробка нових систем для відслідковування об'єктів за допомогою БПЛА є надзвичайно важливими і актуальними на сучасному етапі розвитку технологій.

1.2 Аналіз сучасних технологій відслідковування об'єктів на відео в реальному часі

У сучасному світі технології відслідковування об'єктів на відео в реальному часі використовуються в різних сферах, таких як безпека, автоматизація виробництва, військові операції, транспорт, а також у сфері розваг. Вони базуються на алгоритмах комп'ютерного зору та методах машинного навчання, що дозволяють ідентифікувати об'єкти та стежити за їх переміщенням. Основні технології, що застосовуються сьогодні, можна розділити на кілька категорій.

Алгоритми на основі ключових точок та ознак були одними з перших підходів до відслідковування об'єктів. Методи, такі як SIFT (Scale – Invariant Feature Transform) та ORB (Oriented FAST and Rotated BRIEF), дозволяють знаходити унікальні характеристики об'єктів (ключові точки) і відслідковувати їх у різних кадрах відео. Ці методи є досить ефективними, коли йдеться про об'єкти з добре визначеними формами або текстурами, які можна ідентифікувати та порівнювати між кадрами [4].

Основними перевагами цих методів є їхня стійкість до змін масштабу та обертання об'єктів. Однак вони втрачають ефективність при роботі в умовах поганого освітлення або значних змін вигляду об'єктів, наприклад, при частковому закритті або зміні форми об'єкта. Крім того, ці алгоритми мають відносно високу обчислювальну складність, що обмежує їх використання в реальних додатках з великою кількістю об'єктів або високими вимогами до швидкості.

Методи відслідковування об'єктів на основі шаблонів ґрунтуються на використанні попередньо визначених зразків або шаблонів об'єкта. Один із найпоширеніших методів — MOSSE (Minimum Output Sum of Squared Error) — є простим і швидким кореляційним фільтром, який дозволяє відстежувати об'єкти на основі їхньої текстури або контурів [5].

Ці методи є надзвичайно ефективними у реальному часі через свою низьку обчислювальну складність, але вони мають обмеження. Вони менш стійкі до змін об'єктів (зміни форми, розміру, кольору) і можуть втратити об'єкт, якщо він сильно змінюється між кадрами або має схожі об'єкти поблизу.

Сьогодні методи глибинного навчання стали домінуючими в області відслідковування об'єктів. Глибинні нейронні мережі (DNN), такі як YOLO (You Only Look Once), SSD (Single Shot Detector) та Faster R – CNN, дозволяють здійснювати точне та швидке відслідковування об'єктів у реальному часі. Основна перевага глибинного навчання — це здатність працювати з дуже різними об'єктами, незалежно від їхнього розміру, форми або освітлення [6].

YOLO є одним із найшвидших методів для відслідковування об'єктів. Він об'єднує процес розпізнавання та відстежування об'єктів в одній нейронній мережі, що дозволяє аналізувати зображення за один прохід. Завдяки цьому YOLO забезпечує високу швидкість обробки відео, що робить його ідеальним для додатків реального часу. Однак точність цієї моделі може бути нижчою для дуже малих або дуже складних об'єктів.

SSD також дозволяє одночасно виявляти та відстежувати кілька об'єктів на одному зображенні. Ця модель використовує багатошарову архітектуру для одночасного прогнозування меж і класів об'єктів, що робить її ефективною для відслідковування різних типів об'єктів.

Faster R — CNN є більш точною, але повільнішою моделлю порівняно з YOLO та SSD. Вона використовує два етапи: спочатку виявляє області інтересу, а потім класифікує та відслідковує об'єкти в цих областях. Ця модель краще підходить для задач, де важлива точність, а не швидкість [7].

Основною перевагою методів глибинного навчання є їхня виняткова адаптивність до складних умов та здатність ефективно обробляти сцени зі значною кількістю змінних. Ці методи можуть виявляти й ідентифікувати об’єкти, які змінюють свою форму, розташування або частково закриті іншими об’єктами чи елементами фону. Завдяки цьому вони знаходять застосування у завданнях, де традиційні алгоритми комп'ютерного зору виявляються малоефективними. Такі ситуації часто виникають у реальних умовах, де середовище є динамічним, а об’єкти мають різноманітні розміри, швидкість руху та орієнтацію.

Водночас, значною вимогою до використання методів глибинного навчання є наявність потужних обчислювальних ресурсів. Це обумовлено потребою у виконанні великої кількості операцій на кожному кадрі відео, особливо коли йдеться про сцени з високою роздільною здатністю або значною кількістю об'єктів. Для роботи таких методів необхідні сучасні графічні процесори, здатні паралельно обробляти великі обсяги даних. Крім того, для

досягнення високої точності моделі потребують навчання на значних обсягах даних, що також потребує часу та ресурсів.

Щоб зрозуміти ключові характеристики та порівняти ефективність методів глибинного навчання з іншими підходами, створимо таблицю 1.1.

Таблиця 1.1 — Порівняльна характеристика методів визначення об'єктів

Метод	Висока швидкість	Висока точність	Підходить для малих об'єктів	Вимагає багато ресурсів	Реальний час обробки
YOLO	+	+	–	–	+
Faster R–CNN	–	+	+	+	–
SSD	+	+	–	–	+
RetinaNet	–	+	+	+	–

Для відслідковування об'єктів, які переміщуються на відео, часто використовуються методи фільтрації руху, такі як Kalman Filter та Particle Filter.

Kalman Filter — це лінійний рекурсивний фільтр, який використовує інформацію про попередні положення об'єкта для передбачення його майбутнього руху. Він є досить ефективним для задач, де об'єкт рухається по передбачуваній траєкторії. Однак він може мати обмеження при роботі з нелінійними траєкторіями або при раптових змінах руху [8].

Particle Filter — це більш складний метод, який використовує набір частинок для моделювання положення об'єкта. Цей підхід є ефективним для задач, де рух об'єкта є більш хаотичним або нелінійним, але потребує більше обчислювальних ресурсів і є повільнішим.

Кожен з розглянутих методів має свої переваги та недоліки залежно від конкретних умов задачі. Методи на основі ознак та шаблонів є простими у реалізації, але не завжди ефективними при зміні об'єктів або складних умовах освітлення. Методи глибинного навчання, зокрема YOLO та SSD, є найбільш адаптивними та точними, особливо для задач, де об'єкти постійно змінюються, але вони потребують більше ресурсів. Методи фільтрації руху, такі як Kalman

Filter та Particle Filter, є ефективними для задач, де потрібно передбачати траєкторію об'єкта.

Після проведеного аналізу сучасних технологій відслідковування об'єктів на відео в реальному часі, у таблиці 1.2 наведено порівняльну характеристику основних методів, що використовуються в цій сфері, з метою визначення їхніх переваг і недоліків.

Таблиця 1.2 — Порівняльна характеристика методів відслідковування об'єктів

Метод	Підходить для нелінійного руху	Швидкість	Висока точність	Вимагає багато ресурсів	Обробка оклюзій
Kalman Filter	–	+	+	–	–
Particle Filter	+	–	+	+	+
SORT	–	+	–	–	–
Deep SORT	+	–	+	+	+

На основі проведеного порівняльного аналізу технологій відслідковування об'єктів на відео в реальному часі можна зробити висновок про необхідність комбінованого підходу, що дозволяє максимально ефективно використовувати переваги кожного з методів, зменшуючи їх недоліки. Це підкреслює важливість подальших досліджень у цій галузі, спрямованих на вдосконалення алгоритмів і технологій для досягнення високої точності та швидкості обробки в умовах реального часу.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ВІДСЛІДКОВУВАННЯ ОБ'ЄКТІВ

2.1 Архітектура системи відслідковування об'єктів

Архітектура системи відслідковування об'єктів на відео в реальному часі складається з кількох ключових модулів, які забезпечують ефективну обробку інформації, ідентифікацію об'єктів та їх подальше відстежування. Загальна архітектура системи може бути представлена у вигляді багатошарової модульної структури, що дозволяє гнучко адаптувати систему до різних умов і вимог.

На наступному етапі зосередимо увагу на компонентах, що формують основу розробленої системи. Кожен модуль виконує специфічну функцію в загальній структурі та забезпечує реалізацію певних етапів обробки даних.

Модуль збору даних є першим компонентом системи відслідковування об'єктів, і його основне завдання полягає в отриманні відеопотоку з камер або інших джерел у режимі реального часу. Цей модуль може включати в себе різні типи камер, такі як статичні, мобільні, або безпілотні літальні апарати, оснащені камерами, а також підтримувати різні формати відеозйомки, наприклад, HD або 4K.

Основною функцією модуля є підключення до джерел відео. Це забезпечує гнучкість системи, дозволяючи використовувати різноманітні пристрої для збору інформації. Наступним важливим етапом є налаштування параметрів зйомки. Модуль надає можливість регулювати параметри, такі як роздільна здатність, частота кадрів, експозиція та баланс білого. Це налаштування дозволяє досягти оптимальних умов для отримання якісного відео, що безпосередньо впливає на точність наступних етапів обробки.

Крім того, модуль здійснює обробку відеопотоку, включаючи буферизацію даних, щоб уникнути втрати інформації в разі переривання передачі. Модуль також відповідає за моніторинг стану джерела, що включає перевірку працездатності підключених камер і забезпечення їх стабільної роботи. У випадку виникнення проблем модуль може генерувати сповіщення для оператора, що дозволяє оперативно реагувати на збої.

Узагальнюючи, модуль збору даних є критично важливим етапом у системі відслідковування об'єктів, оскільки забезпечує якість і доступність вихідних даних для подальшої обробки. Правильне налаштування та функціонування цього модуля значно впливають на точність і швидкість роботи всієї системи.

Модуль обробки зображень виконує критично важливу роль у системі відслідковування об'єктів, оскільки його основним завданням є підготовка отриманих відеопотоків до наступного етапу — виявлення об'єктів. Цей модуль виконує низку функцій, які покращують якість зображення та готують дані для подальшого аналізу.

Першою важливою функцією є попередня обробка зображень. У цьому процесі застосовуються фільтри для зменшення шуму, що може значно покращити якість зображень. Наприклад, фільтр Гауса використовується для розмивання зображення, що знижує вплив випадкових шумів [9]. Цей фільтр працює за допомогою матриці, відомої як "ядро Гауса", яка застосовується до пікселів зображення для розмиття країв, що дозволяє зменшити високочастотні компоненти шуму (рисунок 2.1). Далі модуль виконує покращення контрастності. Це може бути досягнуто за допомогою алгоритмів вирівнювання гістограми, які допомагають виділити деталі в зображеннях з низьким контрастом.

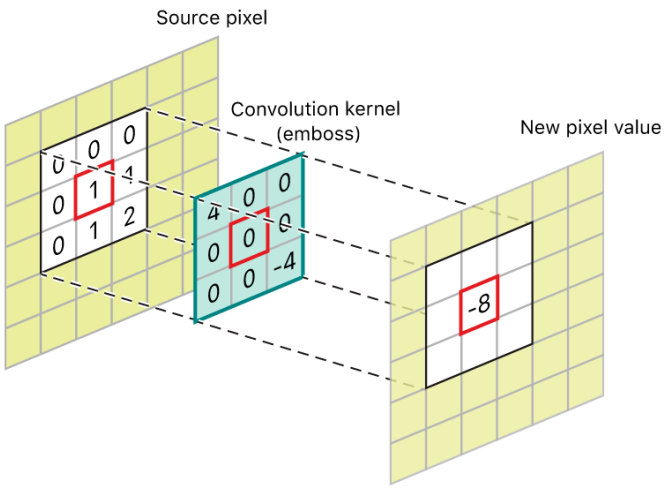

Рисунок 2.1 — Матриця розмиття для зменшення високочастотні компоненти шуму.

Наступним кроком є зміна розміру та нормалізація зображень. Модуль перетворює зображення до стандартних розмірів, що є необхідним для подальшого аналізу. Нормалізація яскравості та кольорів також може бути виконана, щоб створити однорідний набір даних, що сприяє покращенню точності виявлення.

Крім того, модуль виявляє краї та контури об'єктів у зображеннях. Використання алгоритмів для виявлення країв, таких як алгоритм Кенделла [10], дозволяє виділити важливі елементи на зображеннях, які будуть корисні для наступних етапів виявлення об'єктів.

Після завершення обробки модуль передає оброблені зображення до модуля виявлення об'єктів для подальшого аналізу. Це забезпечує безперервність роботи всієї системи та її здатність реагувати в реальному часі.

Таким чином, модуль обробки зображень забезпечує якісну підготовку вхідних даних для подальшої роботи, що є критично важливим для досягнення високої точності виявлення об'єктів у реальному часі.

Модуль виявлення об'єктів є одним із ключових компонентів системи відслідковування, оскільки він відповідає за ідентифікацію та локалізацію об'єктів у оброблених зображеннях. Основною метою цього модуля є точне виявлення об'єктів, що з'являються на вхідних зображеннях, для подальшого їх відстежування.

Процес виявлення об'єктів починається з вибору алгоритму виявлення. Залежно від вимог системи, таких як точність, швидкість обробки та специфіка об'єктів, можуть використовуватися різні алгоритми, такі як YOLO (You Only Look Once), SSD (Single Shot Detector) та Faster R — CNN. Кожен з цих алгоритмів має свої особливості та переваги, що впливають на загальну продуктивність системи.

Після вибору алгоритму модуль обробляє вхідні зображення, отримані з модуля обробки зображень. Алгоритм виявлення проводить аналіз зображення, використовуючи свої внутрішні механізми для виявлення об'єктів. Наприклад, YOLO аналізує зображення за один прохід, розбиваючи його на сітку та

прогнозуючи ймовірності присутності об'єктів у кожній клітинці (рисунок 2.2). У свою чергу, Faster R — CNN реалізує два етапи: спочатку визначає області інтересу, а потім класифікує та відстежує об'єкти в цих областях.

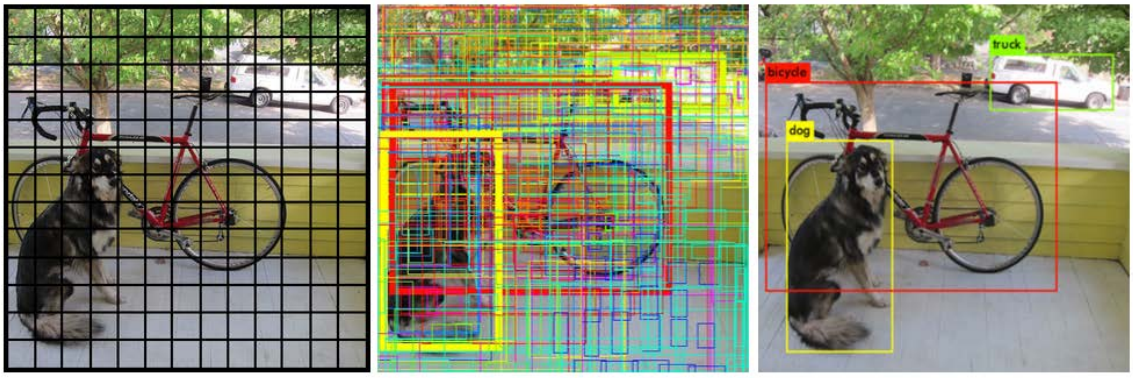

Рисунок 2.2 — Процес розпізнавання об’єктів за допомогою алгоритму YOLO

Після виявлення об'єктів модуль отримує координати (рамки) і класи для кожного виявленого об'єкта. Ця інформація є критично важливою для подальшого відстежування. Для зменшення помилкових спрацьовувань модуль виконує постобробку результатів, яка може включати алгоритми, такі як Non – Maximum Suppression [11]. Цей процес допомагає зберегти лише найбільш вірогідні виявлення, усуваючи дублюючі рамки.

Завершивши виявлення, модуль передає отримані дані (оновлені координати та ідентифікатори) до модуля відстежування, забезпечуючи безперервний моніторинг об'єктів.

Модуль виявлення об'єктів є критично важливим для забезпечення точності та швидкості виявлення, що впливає на загальну ефективність системи. Якість роботи цього модуля визначає успішність наступних етапів відстежування об'єктів у реальному часі.

Модуль відстежування об'єктів є важливим елементом у системі відслідковування об'єктів, оскільки його основне завдання полягає в моніторингу переміщень виявлених об'єктів протягом усього відеопотоку. Після того, як об'єкти були виявлені модулем виявлення, модуль відстежування забезпечує їх

подальше розпізнавання та моніторинг у кожному наступному кадрі, що є критично важливим для роботи систем у реальному часі.

Першим етапом є ідентифікація об'єктів. Модуль отримує дані про координати та класи виявлених об'єктів і присвоює кожному з них унікальний ідентифікатор (ID). Це дозволяє системі однозначно розпізнавати кожен об'єкт навіть після його переміщення. Відстежування переміщення об'єктів є ключовою функцією цього модуля. Для цього застосовуються різні алгоритми, зокрема Kalman Filter та Particle Filter. Kalman Filter є лінійним рекурсивним фільтром, який передбачає майбутнє положення об'єкта на основі його попереднього руху, використовуючи оцінки стану і невизначеність вимірювань, що робить його дуже ефективним для передбачуваних траєкторій руху. Він базується на математичних моделях, які описують динаміку руху об'єкта і його спостереження, що дозволяє системі не тільки оцінювати положення об'єкта, а й коригувати свої прогнози в реальному часі (рисунок 2.3). У свою чергу, Particle Filter є більш гнучким методом і підходить для складніших і хаотичних траєкторій руху, використовуючи набір частинок для моделювання можливого положення об'єкта.

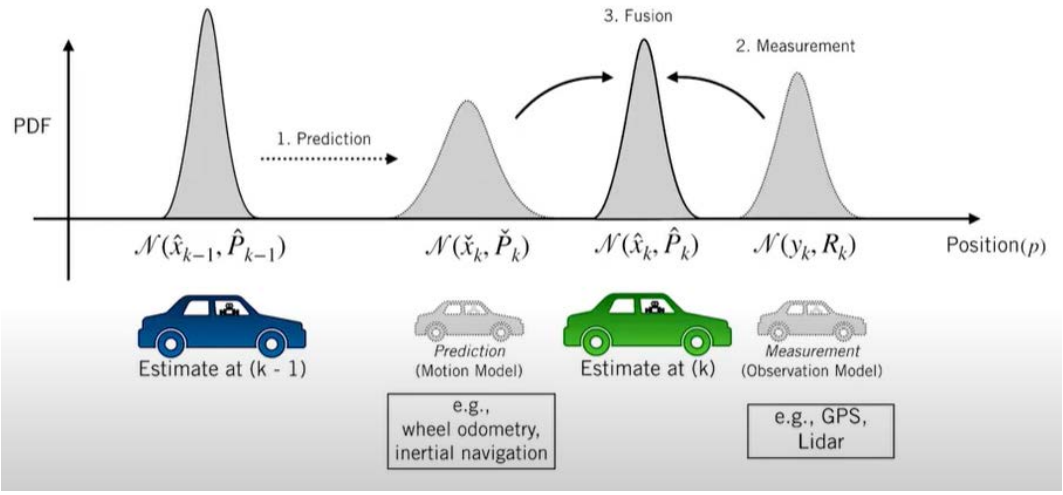

Рисунок 2.3 — Діаграма роботи Kalman Filter на прикладі автомобіля

Модуль відстежування також має здатність адаптуватися до змін у поведінці об'єктів. Наприклад, у випадку, коли об'єкт змінює форму, швидкість

або напрямок руху, система використовує дані про попередні положення об'єкта для корекції та прогнозування його подальших переміщень. Ця здатність до адаптації є критично важливою для збереження точності відстежування навіть у мінливих умовах, наприклад, у випадках, коли кілька об'єктів перетинаються або взаємодіють один з одним.

Моніторинг стану об'єктів є ще однією важливою функцією модуля. Він оцінює стан об'єктів, включаючи їх активність та взаємодію з іншими об'єктами. Це може включати в себе виявлення зіткнень або наближення між об'єктами. Такий моніторинг дозволяє системі залишатися гнучкою та відповідати на зміни, що відбуваються у відеопотоці.

Після завершення процесу відстежування модуль передає результати — оновлені координати об'єктів та їх ідентифікатори — до модуля візуалізації. Це дозволяє користувачеві або операторам бачити актуальну інформацію про поточний стан і розташування об'єктів у реальному часі.

Модуль відстежування об'єктів виконує надзвичайно важливу функцію в системі, забезпечуючи стабільний контроль за об'єктами під час їхнього переміщення. Здатність системи прогнозувати рух об'єктів і адаптуватися до змін дозволяє забезпечити високу точність та ефективність у реальному часі. Ефективна робота цього модуля визначає загальну успішність системи відслідковування об'єктів.

Модуль візуалізації є кінцевою ланкою у системі відслідковування об'єктів, яка відповідає за відображення результатів роботи всіх попередніх модулів у зручному для користувача форматі. Основною функцією цього модуля є представлення даних про виявлені та відстежувані об'єкти на екрані в режимі реального часу. Візуалізація дозволяє не лише відстежувати поточні координати та характеристики об'єктів, але й приймати швидкі рішення на основі отриманих даних.

Основним завданням модуля є накладення графічних елементів, таких як рамки, на виявлені об'єкти у відеопотоці. Ці рамки відображають межі кожного об'єкта, що було виявлено та відстежено системою. Крім того, над рамками може

бути показано додаткову інформацію, таку як клас об'єкта, його унікальний ідентифікатор та, за необхідності, ймовірність правильності класифікації.

Крім рамок, модуль візуалізації дозволяє відображати додаткові метрики, такі як швидкість руху об'єктів, напрямок їхнього переміщення, взаємодія з іншими об'єктами тощо. Це дозволяє користувачеві отримувати більш повну картину про стан об'єктів у полі зору та їхній взаємний вплив. У реальних застосуваннях, таких як охоронні системи або моніторинг у промисловості, така візуалізація може суттєво підвищити ефективність прийняття рішень.

Ще однією важливою функцією модуля візуалізації є можливість масштабування та налаштування зображення. Користувач може збільшувати або зменшувати масштаб зображення для більш детального перегляду окремих ділянок, а також змінювати вигляд або кут огляду. Це особливо корисно для моніторингу великих територій або складних об'єктів.

Модуль візуалізації також відповідає за інтерактивність. Оператор може використовувати інтерфейс для взаємодії з системою, наприклад, для виділення важливих об'єктів або вибору специфічних параметрів для моніторингу. Така інтерактивність робить систему більш гнучкою та адаптованою до конкретних потреб користувача.

Після виведення всіх візуальних елементів на екран модуль також забезпечує збереження відеопотоку з накладеними графічними елементами у вигляді звіту або запису. Це дає можливість перегляду історії відстежування об'єктів у майбутньому, що є корисним у випадках, коли потрібно провести аналіз минулих подій.

Модуль візуалізації є ключовим елементом, що дозволяє користувачам безпосередньо взаємодіяти з системою відслідковування об'єктів. Він забезпечує зручне та інтуїтивне відображення результатів у реальному часі, надаючи оператору можливість приймати швидкі та обґрунтовані рішення на основі отриманих даних. Правильна реалізація цього модуля дозволяє значно покращити ефективність системи та її зручність для кінцевого користувача.

Модуль аналізу даних є важливою частиною системи відслідковування

об'єктів, який відповідає за обробку та інтерпретацію даних, зібраних під час виявлення та відстежування об'єктів. Основна мета цього модуля полягає у глибокому аналізі даних, що дозволяє не лише моніторити об'єкти в режимі реального часу, але й здійснювати довготривале зберігання, статистичну обробку, виявлення аномалій та підготовку інформації для подальшого використання.

Першочергово модуль аналізу даних отримує інформацію від попередніх модулів системи, таких як модулі виявлення та відстежування об'єктів. Ця інформація включає координати об'єктів, їх класи, ідентифікатори, швидкість переміщення та інші метрики, що можуть бути важливими для аналізу. Зібрані дані зберігаються у базі даних, що дозволяє їх використовувати для подальшої обробки, в тому числі для навчання штучного інтелекту або оптимізації алгоритмів.

Однією з основних функцій модуля аналізу даних є статистична обробка. Модуль проводить збір і обробку статистики по кожному з об'єктів, відстежуючи їхній рух, зміну характеристик, взаємодію з іншими об'єктами. На основі цієї інформації можуть бути створені звіти про активність об'єктів за певний період, що є корисним для різноманітних сфер, таких як безпека, охорона, моніторинг промислових процесів або навіть у наукових дослідженнях.

Крім того, модуль аналізу даних займається виявленням аномалій. Це означає, що система здатна розпізнавати незвичні або нетипові зміни в поведінці об'єктів, такі як раптові зміни швидкості, напрямку руху або взаємодії з іншими об'єктами. Виявлення аномалій є критично важливим для застосувань, де необхідно швидко реагувати на непередбачені події, наприклад, у системах безпеки або контролю на виробництвах.

Модуль аналізу даних також може забезпечити прогнозування подій на основі історичних даних. Використовуючи методи машинного навчання, система здатна аналізувати попередні патерни поведінки об'єктів і передбачати можливі майбутні сценарії. Це особливо корисно для завдань, пов'язаних із плануванням операцій або прийняттям рішень у реальному часі.

Завдяки модулю аналізу даних система відслідковування об'єктів стає більш інтелектуальною та здатною до самонавчання. Зібрані та оброблені дані можуть бути використані для оптимізації алгоритмів виявлення та відстежування об'єктів, що дозволяє системі адаптуватися до нових умов і покращувати свою продуктивність.

Модуль аналізу даних відіграє ключову роль у підвищенні інтелектуальної складової системи відслідковування об'єктів. Його здатність до збору, обробки та аналізу великого обсягу даних дозволяє не лише моніторити об'єкти у режимі реального часу, але й здійснювати довгостроковий аналіз та приймати обґрунтовані рішення на основі отриманих результатів. Таким чином, цей модуль значно підвищує ефективність і точність роботи всієї системи.

Підсумовуючи, розробка системи відслідковування об'єктів з використанням такої багатошарової архітектури дозволяє досягти високої точності, гнучкості та адаптивності. Кожен з модулів виконує важливі функції, які в сукупності забезпечують стабільну та ефективну роботу системи в реальному часі.

2.2 Вибір алгоритмів для відслідковування об'єктів

Вибір алгоритмів для відслідковування об'єктів у реальному часі є критично важливим етапом у побудові ефективної системи. Для досягнення високої точності та швидкості роботи, необхідно враховувати специфіку завдання, доступні ресурси та умови, в яких працюватиме система. У нашій системі, яка базується на обробці відеопотоків і має працювати в реальному часі, обрано кілька ключових алгоритмів для виявлення та відстежування об'єктів.

Для виявлення об'єктів на зображеннях обрано алгоритм YOLO (You Only Look Once). Цей алгоритм є одним з найбільш популярних у задачах комп'ютерного зору завдяки його високій швидкості та точності при роботі у реальному часі. YOLO розбиває зображення на сітку та для кожної клітинки прогнозує ймовірність присутності об'єкта, його координати та клас. Головна перевага YOLO полягає у тому, що виявлення об'єктів виконується за один

прохід нейронної мережі, що дозволяє значно знизити час обробки зображення порівняно з іншими методами.

Далі наведені причини, чому було обрано саме YOLO:

— висока швидкість;

— висока точність;

— універсальність.

YOLO здатний обробляти до 45 кадрів на секунду (FPS), що робить його одним із найшвидших алгоритмів виявлення об'єктів. Для системи, яка має працювати в реальному часі, швидкість обробки є вирішальним фактором.

Незважаючи на високу швидкість, алгоритм демонструє точні результати, особливо для великих і добре помітних об'єктів.

Універсальність. YOLO підтримує виявлення широкого спектру об'єктів, що є важливим для універсальних систем спостереження та моніторингу.

YOLO (You Only Look Once) використовує нейронну мережу, яка прогнозує рамки для об'єктів та їх класифікацію одночасно. Ключовою особливістю YOLO є те, що він розбиває вхідне зображення на сітку розміром S×S, де кожна клітинка прогнозує декілька рамок (bounding boxes) та ймовірності приналежності до певного класу.

Формула, що описує функцію втрат для YOLO (2.1).

$$L = \lambda_{\boldsymbol{coord}} \sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left((x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right) + \lambda_{\boldsymbol{coord}} \sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left(\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right)+$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+$$
$$+\lambda_{\boldsymbol{noord}} \sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2+\sum_{i=0}^{S^2} 1_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2 \quad (2.1)$$

де $\lambda_{\boldsymbol{coord}}$ і $\lambda_{\boldsymbol{noord}}$ — вагові коефіцієнти для різних частин функції втрат;

1_{ij}^{obj} — індикатор того, чи є об'єкт в конкретній рамці;

(x_i, y_i) і $(\hat{x}_i, \hat{y}_i)$ — прогнозовані і реальні координати центру рамки об'єкта;

(w_i, h_i) і $(\hat{w}_i, \hat{h}_i)$ — ширина та висота рамки об'єкта;

C_i — довіра до рамки (confidence score), тобто ймовірність наявності об'єкта;

$p_i(c)$ — ймовірність, що об'єкт належить до класу c.

Цей алгоритм добре підходить для задач, де важлива швидкість і достатня точність для виявлення основних об'єктів, які не мають складних форм чи взаємодій. YOLO прагне мінімізувати цю функцію втрат, балансуючи між точністю прогнозу рамок та ймовірністю класифікації.

Для відстежування об'єктів після їхнього виявлення обрано поєднання Kalman Filter та DeepSORT [12].

Kalman Filter є основним алгоритмом для прогнозування та оцінки положення об'єкта в динамічних системах, особливо коли об'єкт рухається за передбачуваною траєкторією. Він використовує поточний стан об'єкта та прогнозує майбутнє положення з урахуванням шуму та невизначеності.

Kalman Filter складається з двох основних етапів: прогнозування (2.2) та оновлення (2.3).

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \quad (2.2)$$

де $\hat{x}_{k|k-1}$ — прогнозований стан на момент часу k;

A — матриця стану, яка описує, як попередній стан переходить до наступного;

u_k — керуючий сигнал;

$P_{k|k-1}$ — прогнозована коваріаційна матриця, яка описує невизначеність прогнозу;

Q — матриця шуму процесу.

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(z_k - H\hat{x}_{k|k-1}\right)$$
$$P_{k|k} = (I - K_kH)P_{k|k-1} \quad (2.3)$$

де K_k — Kalman gain, що визначає, наскільки сильно слід коригувати прогноз;

H — матриця вимірювань;

z_k — фактичне вимірювання на момент часу k;

R — матриця шуму вимірювань.

DeepSORT є розширенням алгоритму SORT, де використовується глибинна нейронна мережа для покращення асоціації між треками та новими виявленнями об'єктів. Основою алгоритму є Kalman Filter, проте його основна сила полягає в асоціації об'єктів за допомогою appearance–based features.

У DeepSORT на кожен об'єкт будується дескриптор, який представляє вигляд об'єкта у векторному просторі ознак. Ці дескриптори використовуються для вирішення проблеми асоціації об'єктів, особливо коли об'єкти перетинаються або взаємодіють.

Формула асоціації об'єктів (2.4).

$$d(a,b) = ||f(a) - f(b)||_2 \quad (2.4)$$

де $f(a)$ і $f(b)$ — вектори ознак для об'єктів a і b;

$d(a,b)$ — евклідова відстань між цими векторами, яка використовується для оцінки схожості.

Якщо відстань $d(a,b)$ є достатньо малою, нове виявлення b асоціюється з треком a.

Kalman Filter забезпечує стабільне відстежування об'єктів з передбачуваною траєкторією руху, а DeepSORT підвищує точність у випадках складних рухів чи перетинань об'єктів.

DeepSORT дозволяє системі гнучко адаптуватися до нових умов та

точніше відстежувати об'єкти в динамічних середовищах.

Завдяки поєднанню двох методів відстежування система залишається ефективною навіть при великій кількості об'єктів на сцені, що робить її придатною для масштабних застосувань.

Ця комбінація алгоритмів забезпечує баланс між швидкістю та точністю, що дозволяє ефективно вирішувати завдання у реальному часі.

Обрані алгоритми дозволяють системі бути ефективною як у швидкості, так і в точності. YOLO забезпечує швидке і точне виявлення об'єктів, що особливо важливо для систем, які працюють з відеопотоками в реальному часі. Kalman Filter та DeepSORT, у свою чергу, дозволяють не лише точно відстежувати об'єкти, але й адаптуватися до змін у їхній поведінці, що підвищує загальну надійність системи. Поєднання цих алгоритмів забезпечує стабільну роботу навіть у складних динамічних середовищах.

2.3 Розробка алгоритму відслідковування об’єктів на відео в реальному часі

Алгоритм відеовідстеження є сукупністю інструкцій, що визначають, як система ідентифікує і тримає об'єкт у фокусі під час руху або змін середовища. Основна мета реалізації полягає у забезпеченні точності і ефективності виявлення і прогнозування руху об'єктів. Відповідно, до кожної стадії алгоритму потрібно підходити із забезпеченням точних математичних моделей та ефективних обчислювальних методів.

Першим етапом є отримання відеопотоку в реальному часі з камери БПЛА. Система отримує цей потік у вигляді кадрів (послідовність зображень) і передає їх для подальшої обробки. Важливою частиною на цьому етапі є попередня обробка даних, кадри нормалізуються для забезпечення однакових умов роботи алгоритму. Можуть застосовуватися методи фільтрації шумів, покращення контрасту або масштабування, щоб підвищити якість зображення.

Для виявлення об'єктів на сцені застосовуються конволюційні нейронні мережі (YOLO). На кожному кадрі система шукає об'єкти, які відповідають

певним класам (наприклад, транспортні засоби, люди). Основним математичним інструментом тут є згортки з ядрами фільтрації для автоматичного виокремлення особливостей об'єктів. Набір даних використовується для тренування моделі, яка здатна виявляти та класифікувати об'єкти з високою точністю.

Для кожного кадру з відеопотоку створюється матриця ймовірностей на основі функції втрат, яка описує розбіжність між передбаченими та фактичними координатами об'єкта. Розв'язком системи є мінімізація цієї втрати через зворотне поширення помилки, що забезпечує точність прогнозування місцезнаходження об'єкта на нових кадрах.

Коли об'єкт визначено, застосовується алгоритм відстеження на основі Kalman Filter або Sort [13]. В основі Kalman Filter лежить стохастичний процес прогнозування, де система намагається передбачити положення об'єкта в наступному кадрі на основі попередніх даних. Це дозволяє не лише слідкувати за об'єктом під час його руху, але і коригувати зміни у випадку пропущених кадрів або шуму.

Математично це описується системою рівнянь, яка враховує поточне положення та швидкість об'єкта. Наприклад, алгоритм Sort використовує формулу (2.5).

$$x_{t+1} = Fx_t + w_t \tag{2.5}$$

де x_t — це стан об'єкта в момент часу t;

F — матриця переходів;

w_t — випадкова шумова складова.

Ця модель коригує положення об'єкта з урахуванням можливих варіацій і неточностей вимірювань.

Якщо об'єкт тимчасово зникає з кадру, використовується екстраполяція його руху. На основі попередніх позицій система прогнозує ймовірне майбутнє положення об'єкта, навіть якщо його тимчасово не видно. Прогнозування

дозволяє БПЛА не втрачати об'єкт з поля зору, що критично важливо у випадку втрати зв'язку або швидких маневрів.

Після обробки даних система надсилає команду БПЛА щодо необхідних маневрів для збереження об'єкта в полі зору камери. На цьому етапі алгоритм також обробляє можливі помилки, пов'язані з втратою об'єкта, і виводить повідомлення про помилки або підтвердження успішного відстеження. Завдяки інтеграції системи з штучним інтелектом, БПЛА здатен адаптувати свою стратегію в реальному часі.

Блок-схема алгоритму відслідковування об'єктів починається з етапу "Початок", де здійснюється ініціалізація системи, налаштування обладнання для збору відеопотоку та підготовка алгоритмів для подальшої роботи. На цьому етапі також перевіряється підключення до джерела відео, такого як веб-камера або відеофайл.

Далі система перевіряє наявність нового кадру у відеопотоці. У разі його відсутності система переходить у режим очікування, щоб знизити обчислювальне навантаження. Якщо новий кадр виявлено, він надсилається на попередню обробку.

Після попередньої обробки кадр передається на виявлення об'єктів за допомогою алгоритму YOLO. Якщо об'єкт не знайдено, система повертається до обробки наступного кадру. У разі успішного виявлення координати об’єкта передаються в модуль відстеження, де використовується Kalman Filter або Sort для оцінки його траєкторії.

Алгоритм відстеження забезпечує точність навіть у складних умовах, таких як короткочасне зникнення об'єкта з кадру через перешкоди. Якщо відстеження вдається, система прогнозує наступне положення об’єкта, що дозволяє працювати з мінімальними затримками у режимі реального часу. У разі помилок відстеження система формує повідомлення про несправність, що дозволяє виявити причини збою. Таким чином, алгоритм забезпечує стабільну роботу системи, підтримуючи як високу точність виявлення, так і надійність відстеження об'єктів навіть у змінних умовах середовища (рисунок 2.4).

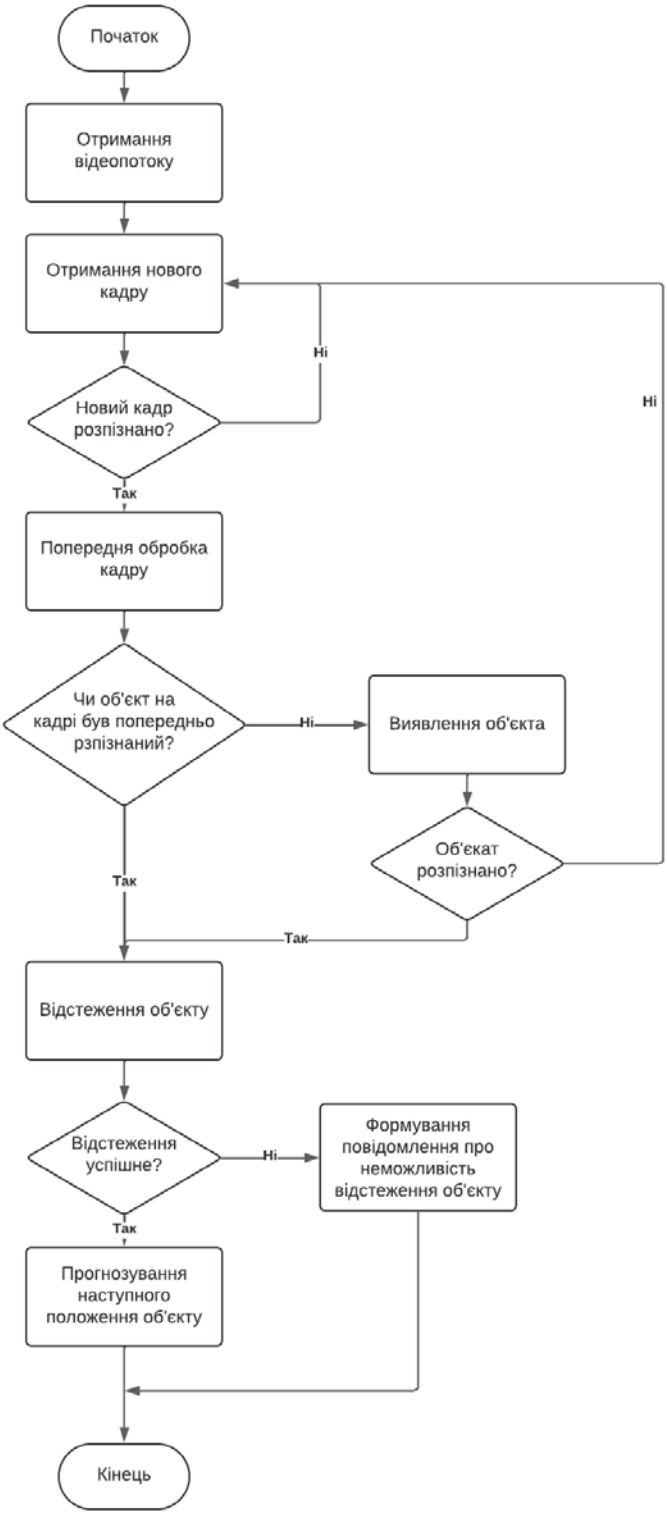

Рисунок 2.4 — Схема алгоритму загального функціонування методів відслідковування об'єктів

Таким чином, весь процес реалізується послідовно, з використанням модульної структури і алгоритмічної оптимізації для ефективного виконання задачі відслідковування об'єктів на відео в реальному часі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ ВІДСЛІДКОВУВАННЯ ОБ’ЄКТІВ НА ВІДЕО В РЕАЛЬНОМУ ЧАСІ

3.1 Обґрунтування вибору мови програмування

У сучасному програмуванні для задач комп'ютерного зору доступно багато мов, і вибір правильної мови є критично важливим кроком у створенні ефективної системи. Кожна мова має свої сильні сторони та обмеження, які можуть по–різному вплинути на продуктивність, точність і зручність розробки. Для реалізації програми відслідковування об’єктів на відео в реальному часі важливо вибрати таку мову, яка забезпечить підтримку необхідних бібліотек для комп’ютерного зору та глибокого навчання, дозволить обробляти відеопотік у режимі реального часу та полегшить інтеграцію різних компонентів системи. У цьому розділі ми дослідимо кілька популярних мов, проаналізуємо їх можливості та визначимо, яка з них є найоптимальнішою для реалізації нашої задачі.

Python є інтерпретованою мовою програмування високого рівня, яка набула широкого поширення у сфері комп’ютерного зору, машинного навчання та штучного інтелекту. Ця популярність зумовлена простим синтаксисом, великою кількістю бібліотек, активною спільнотою та доступністю інструментів для обробки зображень, аналізу даних, нейронних мереж і глибокого навчання, що робить Python зручним і ефективним інструментом для реалізації систем відслідковування об'єктів на відео в реальному часі [14].

Python надає великий вибір бібліотек, які значно спрощують реалізацію завдань комп'ютерного зору та обробки зображень. Серед найбільш відомих і важливих бібліотек можна виділити OpenCV, TensorFlow і PyTorch. OpenCV є однією з найпопулярніших бібліотек для обробки зображень і відео, яка дозволяє виконувати різноманітні операції обробки, такі як розпізнавання об’єктів, сегментація, фільтрація та аугментація зображень. TensorFlow і PyTorch — це потужні фреймворки для машинного навчання і глибокого навчання, що дозволяють легко інтегрувати моделі, такі як YOLO, для ефективного виявлення об'єктів у відеопотоці [15]. Крім того, бібліотеки NumPy та SciPy забезпечують

ефективне виконання операцій з великими масивами та матрицями, що є необхідним для обробки зображень і виконання лінійної алгебри.

Простота синтаксису Python та його інтуїтивно зрозумілий стиль дозволяють скоротити час розробки і тестування. Це дуже зручно для швидкого прототипування та перевірки різних ідей і алгоритмів. Крім того, легкість коду Python значно спрощує підтримку та оновлення системи, що є важливим для великих та складних проєктів.

Python підтримує високорівневі структури даних, такі як списки, кортежі та словники, які дозволяють ефективно маніпулювати інформацією. Це дозволяє будувати добре організовану систему з чіткою структурою та розширюваністю, що особливо важливо для комплексних проєктів.

Хоча Python не є мовою, орієнтованою на високу продуктивність для низькорівневих обчислень, він підтримує паралельні обчислення та використання графічних процесорів (GPU). Завдяки бібліотекам CUDA та CuPy Python може ефективно використовувати GPU для прискорення обчислень у задачах глибокого навчання і обробки відеопотоків у реальному часі. Підтримка GPU є критично важливою для задач реального часу, таких як виявлення об'єктів та їхнє відстежування.

Спільнота Python є однією з найбільших у світі серед розробників, дослідників та інженерів. Це означає, що Python має безліч навчальних матеріалів, форумів, репозиторіїв із прикладами та документацією, які значно полегшують роботу з мовою та вирішення складних задач. Активна спільнота також сприяє швидкому впровадженню інновацій, таких як нові алгоритми та бібліотеки для глибокого навчання, що робить Python ідеальним для задач комп’ютерного зору.

Python підтримує інтеграцію з іншими мовами, такими як C, C++ та Fortran, через інструменти Cython та SWIG. Це дозволяє використовувати Python як інтерфейс для високопродуктивних компонентів, забезпечуючи при цьому зручність і простоту розробки. Така інтеграція є особливо корисною у системах

реального часу, де важлива швидкість обробки даних, але потрібні й інструменти високого рівня для організації складного коду.

Python надає доступ до інструментів для автоматичної оптимізації гіперпараметрів та навчання моделей, таких як Optuna та Hyperopt, що особливо корисно для завдань відслідковування. Це дозволяє налаштовувати параметри фільтрів, наприклад Kalman Filter, для досягнення найвищої точності при мінімальній затримці, що важливо для завдань у реальному часі.

Python також є платформно–незалежною мовою, що дозволяє розробляти кросплатформні рішення для різних операційних систем, таких як Windows, macOS і Linux. Це робить Python популярним вибором у наукових колах та університетах, оскільки він широко використовується для проведення досліджень і розробки експериментальних рішень, включаючи завдання комп'ютерного зору.

Завдяки підтримці популярних архітектур для глибокого навчання Python дозволяє працювати з моделями для виявлення та класифікації об'єктів, такими як YOLO, Faster R–CNN та SSD. Для задач відслідковування Python також підтримує бібліотеки, які реалізують алгоритми Kalman Filter, SORT та DeepSORT, що дозволяє створити повнофункціональну систему для відслідковування об’єктів у реальному часі.

Python є потужним інструментом для розробки систем відслідковування об’єктів завдяки наявності бібліотек для роботи з зображеннями, відео та глибоким навчанням. Він забезпечує швидке прототипування, зручність написання коду та підтримку GPU для обробки великих обсягів даних у реальному часі. Інтеграція Python з іншими мовами дозволяє балансувати між продуктивністю і швидкістю розробки, що робить його оптимальним вибором для створення комплексної системи відслідковування об'єктів на відео в реальному часі.

C++ є однією з найпопулярніших мов програмування для розробки продуктивних систем завдяки своїм можливостям прямого управління ресурсами та високій швидкості виконання. Це мова низького рівня з широкими

можливостями управління пам'яттю, яка дозволяє створювати високопродуктивні програми, критичні для таких задач, як обробка зображень у реальному часі. Мова C++ широко використовується в задачах комп'ютерного зору, зокрема для реалізації інтенсивних обчислень, таких як виявлення та відстеження об'єктів на відео, завдяки високій швидкості та ефективності [16].

Однією з ключових переваг C++ є її здатність до низькорівневого управління пам’яттю, що дозволяє значно оптимізувати програми для досягнення високої продуктивності. Це особливо важливо для комп'ютерного зору, де швидкість обробки даних може бути критичною, особливо для обробки відеопотоку у реальному часі. За допомогою C++ розробники мають можливість точно контролювати використання ресурсів, що дозволяє зменшити затримки та підвищити швидкість виконання алгоритмів.

C++ підтримує потужну об'єктно–орієнтовану парадигму програмування, яка дозволяє створювати модульний код із чіткою структурою. Це значно спрощує побудову великих систем з багатьма модулями, таких як система відслідковування об'єктів. Крім того, C++ підтримує парадигми багатопотокового програмування, що дозволяє ефективно розподіляти задачі на декілька потоків для паралельної обробки. Це особливо важливо при роботі з відео в реальному часі, оскільки багатопотокова обробка дозволяє обробляти декілька об'єктів або кадрів одночасно.

C++ також підтримує інтеграцію з багатьма потужними бібліотеками та фреймворками для комп’ютерного зору та глибокого навчання. Серед найбільш відомих є OpenCV — бібліотека з великою кількістю функцій для обробки зображень і відео, яка містить оптимізовані алгоритми для задач виявлення, сегментації та відстежування об'єктів. Бібліотека OpenCV, розроблена на C++, забезпечує оптимальну продуктивність і є однією з найпопулярніших у задачах комп’ютерного зору. Крім того, C++ підтримує такі фреймворки глибокого навчання, як TensorFlow і Caffe, які мають API для розробки продуктивних моделей, сумісних із високошвидкісними обчисленнями на CPU та GPU.

Іншою важливою перевагою C++ є підтримка низькорівневого доступу до апаратних компонентів, таких як графічні процесори (GPU). Завдяки бібліотекам CUDA та OpenCL можна розробляти програми, що використовують можливості GPU для значного прискорення обчислень. Це робить C++ особливо корисною мовою для задач комп'ютерного зору та машинного навчання, оскільки GPU можуть виконувати масові паралельні обчислення набагато швидше, ніж CPU.

Завдяки активній підтримці багатьох фреймворків для комп'ютерного зору та машинного навчання, а також високій продуктивності, C++ є відмінним вибором для задач, де критично важливі швидкість і контроль над ресурсами. Мова добре підходить для розробки систем відслідковування об'єктів у реальному часі, забезпечуючи баланс між продуктивністю та гнучкістю для оптимізації системи під конкретні вимоги.

Java є мовою програмування високого рівня, яка добре підходить для розробки розподілених і кросплатформних систем завдяки своїй портативності та стабільності. Вона широко використовується у корпоративних рішеннях і системах обробки даних завдяки ефективному управлінню пам’яттю, багатопотоковості та хорошій продуктивності. Незважаючи на те, що Java менш популярна у сферах комп’ютерного зору та глибокого навчання порівняно з Python або C++, її переваги, такі як безпека, масштабованість та кросплатформна сумісність, роблять її хорошим вибором для створення великих, розподілених систем, включаючи обробку відеопотоків у реальному часі.

Однією з найбільш важливих особливостей Java є її віртуальна машина (JVM), яка забезпечує кросплатформність. Це дозволяє розробляти програми, які можна легко запускати на різних операційних системах без значних змін у коді [17]. Кросплатформність є корисною для створення систем, які потребують гнучкості у розгортанні на різних пристроях, зокрема в індустріальних умовах або на серверах для обробки відео та виконання складних обчислень.

Java підтримує багатопотокове програмування, що дозволяє розподіляти завдання обробки на декілька потоків для паралельного виконання. Це особливо корисно для задач відслідковування об’єктів на відео в реальному часі, оскільки

паралельна обробка кадрів або об’єктів допомагає підвищити продуктивність і знизити затримки. У Java є ефективні інструменти для управління багатопотоковістю, які спрощують створення складних систем, здатних обробляти великі обсяги даних.

Java також має ряд бібліотек і фреймворків, які підтримують обробку зображень і комп'ютерне бачення, хоча вони менш розвинені порівняно з бібліотеками на Python чи C++. Наприклад, JavaCV є інтерфейсом до бібліотеки OpenCV, яка забезпечує інструменти для обробки зображень, виявлення об’єктів, фільтрації та аналізу даних у режимі реального часу. Крім того, для глибокого навчання можна використовувати фреймворк Deeplearning4j, який підтримує роботу з нейронними мережами та надає API для створення і навчання моделей на основі глибокого навчання.

Іншою важливою перевагою Java є її стабільність і надійність у великих проєктах. Система автоматичного управління пам'яттю та збірки сміття (Garbage Collection) допомагає уникнути проблем із витоками пам’яті, що є частою проблемою в мовах із ручним управлінням пам'яттю, таких як C++. Це особливо важливо для систем, які мають працювати тривалий час без перезавантаження, наприклад, у відеомоніторингових системах, де потрібна постійна робота з мінімальними збоями.

Java також підтримує інтеграцію з іншими мовами програмування, такими як C та C++, через Java Native Interface (JNI). Це дозволяє використовувати Java як основу для побудови розподіленої системи з можливістю підключення модулів на інших мовах для виконання спеціалізованих задач, де потрібна висока продуктивність, наприклад, для виявлення та відстежування об'єктів.

Незважаючи на меншу популярність у комп’ютерному зорі порівняно з Python, Java надає широкий спектр можливостей для створення масштабованих і надійних систем. Її продуктивність, стабільність і підтримка багатопоточності роблять Java підходящим вибором для реалізації великих проєктів, де важливі стабільність і кросплатформна підтримка, а також для завдань з обробки зображень і відео, що не потребують глибокого навчання на GPU.

MATLAB є потужною мовою та інтегрованим середовищем для чисельних обчислень, яке широко використовується в наукових дослідженнях, інженерії, комп'ютерному зорі та машинному навчанні. MATLAB забезпечує високий рівень продуктивності для наукових та інженерних розрахунків завдяки великій кількості вбудованих функцій і бібліотек, що значно спрощують роботу з математичними та візуалізаційними задачами. Він особливо ефективний для швидкої обробки сигналів, аналізу зображень і побудови алгоритмів для обробки даних [18].

Однією з головних переваг MATLAB є його простий синтаксис і широкий набір інструментів для обробки зображень, включаючи спеціалізовані бібліотеки, як от Image Processing Toolbox і Computer Vision Toolbox. Ці інструменти дозволяють легко створювати і тестувати алгоритми комп'ютерного зору, від початкової обробки зображень до задач виявлення та відстежування об'єктів. MATLAB має вбудовані функції для виконання основних операцій обробки зображень, таких як фільтрація, виявлення країв, сегментація, а також спеціалізовані функції для роботи з фільтрами Гауса, Kalman Filter та іншими алгоритмами відстежування.

MATLAB також відомий своєю потужною системою візуалізації, яка дозволяє легко будувати графіки, зображення, анімації і навіть створювати тривимірні моделі. Це є важливим аспектом для візуалізації результатів обробки зображень і відео, що дозволяє розробникам швидко оцінити роботу алгоритмів. MATLAB дозволяє відображати результати в реальному часі, що може бути корисним для системи відстежування об’єктів.

Ще однією важливою перевагою MATLAB є його інтеграція з різними інструментами для машинного навчання, зокрема з Deep Learning Toolbox, який підтримує побудову та тренування нейронних мереж. MATLAB також має вбудовані інструменти для імпорту моделей глибокого навчання з інших фреймворків, таких як TensorFlow і PyTorch, що може бути корисним для використання вже навчених моделей для задач виявлення та відстежування об'єктів.

MATLAB підтримує багатопотокову обробку та паралельні обчислення, що робить його зручним для задач, які потребують обробки великих обсягів даних. Завдяки цьому MATLAB підходить для роботи з відео в реальному часі, дозволяючи розподіляти обчислення на декілька ядер процесора для підвищення продуктивності. Проте MATLAB має обмежену підтримку роботи з GPU, порівняно з такими мовами, як Python чи C++, що може знижувати його ефективність для дуже великих нейронних мереж та обробки відеопотоків високої роздільної здатності.

Хоча MATLAB не завжди є найкращим вибором для виробничих систем через ліцензійні обмеження та високу вартість, він є дуже корисним у фазі прототипування та досліджень. MATLAB дозволяє розробникам швидко створювати прототипи алгоритмів і тестувати різні підходи без необхідності писати великий обсяг коду, що є особливо цінним для швидкої розробки наукових рішень.

Загалом, MATLAB є хорошим інструментом для швидкого прототипування і тестування алгоритмів комп’ютерного зору, особливо для дослідницьких проектів. Його потужні інструменти обробки зображень і підтримка машинного навчання дозволяють легко реалізовувати та оцінювати алгоритми, хоча обмеження з ліцензіями і менша гнучкість у продуктивних системах можуть обмежувати його застосування для великих і комерційних

Rust — сучасна мова програмування, яка зосереджена на швидкості, безпеці та надійності, особливо при роботі з пам'яттю. Її популярність зростає завдяки вбудованому управлінню пам'яттю без збірки сміття (garbage collection) та високому рівню паралелізму, що робить її перспективним вибором для задач, де важливі продуктивність і низькорівневий контроль над ресурсами. Rust здатна забезпечити високу продуктивність, подібно до C++ та C, але має додаткові механізми для запобігання помилкам із доступом до пам'яті, які часто виникають у мовах низького рівня [19].

Rust особливо підходить для задач комп’ютерного зору, які потребують високої продуктивності та ефективного використання апаратних ресурсів,

оскільки мова дозволяє розробникам писати швидкий і безпечний код. Відсутність збірки сміття означає, що програми на Rust працюють більш передбачувано і без затримок, що є суттєвою перевагою для систем реального часу, таких як обробка відеопотоків і відстежування об'єктів. Завдяки системі управління пам’яттю Rust дозволяє уникнути витоків пам'яті та інших поширених проблем, які можуть знижувати продуктивність і стабільність програм.

Однією з ключових особливостей Rust є паралелізм і безпечне багатопотокове програмування. Rust має унікальну систему управління запозиченням (borrowing) і власністю (ownership), що дозволяє безпечно розподіляти обчислювальні задачі між кількома потоками, без ризику виникнення гонок даних. Це робить Rust потужним інструментом для розробки високопродуктивних програм, які потребують розподілених обчислень і одночасної обробки відео або великих обсягів даних.

Rust також підтримує інтеграцію з іншими мовами, зокрема з C та C++. Це дозволяє використовувати Rust разом з існуючими бібліотеками та фреймворками для комп'ютерного зору, такими як OpenCV. В екосистемі Rust є бібліотеки для обробки зображень, як от image – rs, та інші інструменти для роботи з даними, які можна використовувати для основних операцій обробки зображень. Хоча підтримка бібліотек для комп'ютерного зору в Rust поки що менша, ніж у Python чи C++, можливість інтеграції з іншими мовами дозволяє розробникам ефективно використовувати Rust для обробки даних і паралельних обчислень у рамках систем комп'ютерного зору.

Ще одна перевага Rust — підтримка роботи з GPU, завдяки бібліотекам, як от Rust CUDA та wgpu, які надають можливість використовувати апаратне прискорення для інтенсивних обчислень. Це може бути корисним для програм відстежування об'єктів у реальному часі, де велика кількість обчислень може передаватися на GPU, що значно знижує навантаження на центральний процесор і підвищує швидкість роботи системи.

Однак Rust також має деякі обмеження. Оскільки Rust — відносно нова мова, її екосистема для комп’ютерного зору та машинного навчання менш розвинута порівняно з більш популярними мовами, такими як Python. Це означає, що розробники, які використовують Rust для комп’ютерного зору, можуть стикнутися з меншою кількістю готових інструментів та бібліотек, що може ускладнити реалізацію складних задач.

Загалом, Rust є перспективним вибором для задач, що вимагають високої продуктивності та безпеки при роботі з пам'яттю. Він добре підходить для створення систем реального часу, таких як системи відстежування об'єктів на відео, особливо коли важливі продуктивність, багатопоточність і безпечне управління ресурсами. Незважаючи на молодий вік і меншу екосистему для комп’ютерного зору, Rust пропонує потужні інструменти для створення ефективних і стабільних програм, що робить його перспективним варіантом для обробки зображень і відео у режимі реального часу.

Зважаючи на особливості задачі та вимоги до розробки системи відстежування об'єктів у реальному часі, мова програмування Python є оптимальним вибором для нашого проекту. Python має добре розвинену екосистему бібліотек для комп'ютерного зору та машинного навчання, таких як OpenCV, TensorFlow, Keras, та PyTorch, які дозволяють швидко створювати прототипи та реалізовувати алгоритми для обробки зображень та відстежування об'єктів. Python також підтримує інтеграцію з мовами низького рівня, як–от C і C++, що дозволяє використовувати високопродуктивні модулі в разі потреби.

Хоча мови C++ і Rust мають переваги у продуктивності та оптимальному управлінні пам’яттю, Python надає значну гнучкість і простоту розробки, що є важливим у дослідницькому середовищі. MATLAB, зі своєю спеціалізацією на чисельних обчисленнях і високим рівнем інтеграції бібліотек для обробки зображень, також є потужним інструментом, однак його обмежена продуктивність і високі ліцензійні вимоги обмежують його застосування в реальних системах.

Таким чином, для досягнення балансу між продуктивністю, доступністю інструментів, швидкістю розробки та інтеграцією з потужними бібліотеками Python виглядає як найбільш відповідний вибір. Нижче наведена порівняльна таблиця 3.1, яка показує, як Python відповідає ключовим вимогам порівняно з іншими мовами.

Таблиця 3.1 — Порівняльна характеристика мов програмування

	Python	C++	MATLAB	Rust	Java
Бібліотеки комп'ютерного зору (OpenCV, YOLO, TensorFlow)	+	+	–	–	–
Підтримка нейронних мереж і ML (TensorFlow, PyTorch)	+	+	–	–	+
Легкість синтаксису і швидкість розробки	+	–	+	–	+
Висока продуктивність для обробки відеопотоків	–	+	–	+	+
Підтримка багатопоточності та паралелізму	–	+	–	+	+
Підтримка роботи з GPU	+	+	–	+	–
Інтеграція з низькорівневими бібліотеками	+	+	–	+	–
Ефективне управління пам'яттю для великих даних	–	+	–	+	+

Продовження таблиці 3.1

	Python	C++	MATLAB	Rust	Java
Широка доступність і підтримка наукової спільноти	+	+	+	–	+
Зручність прототипування та тестування	+	–	+	–	+
Кросплатформна підтримка	+	+	+	+	+
Ліцензійна доступність та вартість	+	+	–	+	+

Python виділяється серед інших мов завдяки своїй простоті, широкій підтримці наукових бібліотек та гнучкості. Це дозволяє не тільки швидко розробляти прототипи, а й успішно інтегрувати створені рішення в різноманітні системи. Ураховуючи вимоги проекту, саме Python забезпечить оптимальне співвідношення зручності розробки, доступності інструментів і продуктивності.

3.2 Програмна реалізація модулів виявлення та відстежування об'єктів

Для реалізації програми для виявлення та відстежування об'єктів за допомогою Python необхідно встановити кілька бібліотек, які забезпечать основний функціонал для роботи з комп'ютерним зором, обробки зображень і відстежування об'єктів у режимі реального часу.

OpenCV — одна з найпопулярніших бібліотек для комп'ютерного зору в Python, яка надає широкий набір інструментів для обробки зображень та відеопотоків. OpenCV дозволяє працювати з різними форматами зображень і відео, виконувати попередню обробку, фільтрацію та аналіз даних, що є основою для обробки даних перед виявленням об'єктів.

NumPy — базова бібліотека для роботи з числовими даними в Python, яка надає можливості для маніпуляції з багатовимірними масивами та виконання математичних операцій. NumPy є важливим інструментом для створення та

обробки матриць, які використовуються в обчисленнях під час виявлення та відстежування об'єктів, а також для обробки результатів роботи алгоритмів.

PyTorch — бібліотека для машинного навчання та глибинного навчання, яка підтримує створення нейронних мереж і використання попередньо навчених моделей. PyTorch дозволяє легко використовувати моделі, такі як YOLO, для виявлення об'єктів на зображеннях та відео. Завдяки підтримці роботи з GPU, PyTorch забезпечує високу продуктивність, що критично важливо для програм реального часу.

KalmanFilter з бібліотеки pykalman — класичний алгоритм для відстежування об'єктів, який використовує метод рекурсивного фільтрування для прогнозування наступних позицій об'єкта на основі його попереднього руху. Kalman Filter є надійним методом для передбачення траєкторії об'єкта в умовах шуму та варіацій у його русі.

Scipy — бібліотека для наукових і технічних обчислень, яка часто використовується разом з NumPy. Вона надає розширені математичні та статистичні функції, які можуть знадобитися для обробки даних, виконання фільтрації та підвищення точності обробки результатів виявлення.

Matplotlib — бібліотека для візуалізації даних, яка дозволяє створювати графіки, діаграми та схеми для аналізу результатів відстежування об'єктів. Matplotlib корисна для відображення траєкторії об'єктів або для візуалізації прогнозів, які надає Kalman Filter.

Json — стандартна бібліотека Python для роботи з форматом JSON, яка дозволяє зберігати та обробляти дані в цьому форматі. JSON зручно використовувати для зберігання налаштувань або параметрів моделей, а також для передачі даних між програмами.

Завантаження моделі YOLO для виявлення об'єктів є важливою частиною процесу аналізу зображень або відео за допомогою глибоких нейронних мереж. Для роботи з моделями YOLO в Python зручно використовувати бібліотеку OpenCV, зокрема її модуль для роботи з глибокими нейронними мережами (DNN). OpenCV забезпечує інтеграцію з передтренованими моделями, такими як

YOLO, що дозволяє зручно завантажувати модель, налаштовувати її та застосовувати для детекції об'єктів на вхідних зображеннях.

Основною функцією для завантаження мережі є cv2.dnn.readNet(), яка приймає два основних параметри: шлях до файлу ваг моделі та конфігураційного файлу. Файл ваг містить параметри, які модель використовує для здійснення прогнозів, а конфігураційний файл описує архітектуру мережі — кількість шарів, типи нейронів та їхні зв'язки. У нашій реалізації використовуються стандартні файли для YOLOv3, yolov3.cfg для опису архітектури мережі та yolov3.weights для ваг моделі. Крім того, використовуємо файл coco.names, який містить список класів, що модель може виявляти, таких як "person", "car", "dog" тощо.

Функція cv2.dnn.readNet() створює об'єкт нейронної мережі, який можна використовувати для обробки зображень або відео. Після завантаження мережі важливо перевірити, чи модель була завантажена коректно, що можна зробити через перевірку кількості класів, які модель підтримує. Для цього використовуємо просту перевірку розміру списку класів, зчитаних із файлу coco.names, що гарантує наявність необхідних класів для виявлення об'єктів.

Крім того, важливою частиною цього процесу є завантаження самого списку класів із текстового файлу. Для цього в нашій реалізації використовується функція open() для відкриття файлу класів, та метод readlines() для зчитування кожного рядка з файлу. Кожен рядок файлу містить назву класу, і цей список класів буде використовуватися для подальшої обробки результатів детекції. За допомогою цього підходу ми отримуємо в результаті масив, де кожен елемент відповідає певному класу об'єкта, який може бути виявлений моделлю YOLO.

Простий приклад завантаження моделі YOLO виглядає так.

```
import cv2

# Шляхи до файлів
weights_path = 'yolov3.weights'
cfg_path = 'yolov3.cfg'
names_path = 'coco.names'
```

```
# Завантаження мережі
net = cv2.dnn.readNet(weights_path, cfg_path)

# Завантаження класів
with open(names_path, 'r') as f:
    classes = [line.strip() for line in f.readlines()]

# Перевірка на кількість класів
print(f'Кількість класів: {len(classes)}')
```

У цьому прикладі, функція cv2.dnn.readNet() здійснює завантаження мережі з двох основних файлів — конфігураційного і вагового. Після цього ми відкриваємо файл класів і зчитуємо з нього всі можливі класи, що підтримуються моделлю YOLO. Це забезпечує налаштування моделі для подальшої детекції об'єктів.

Одним з важливих аспектів є перевірка правильності завантаження файлів, що можна зробити, перевіривши, чи відповідає кількість класів очікуваному значенню. У нашому випадку, після завантаження списку класів з файлу coco.names, ми можемо вивести кількість класів і переконатися, що вона збігається з очікуваною для моделі YOLOv3.

Цей підхід використовує стандартні функції бібліотеки OpenCV і дозволяє зручно завантажити модель YOLO для подальшого використання в задачах виявлення об'єктів. Завдяки такій реалізації, користувач отримує доступ до потужних можливостей моделі YOLO і може застосовувати її для обробки зображень або відео в реальному часі.

Ініціалізація фільтра Калмана для відстежування об'єктів є важливим кроком для забезпечення точного прогнозування положення об'єкта на основі наявних вимірювань. У Python для цієї мети зазвичай використовується клас cv2.KalmanFilter, наданий бібліотекою OpenCV. Фільтр Калмана дозволяє коригувати передбачення на основі нових спостережень, що є особливо важливим при відстежуванні об'єктів у реальному часі, де вимірювання можуть

бути шумними або неповними, алгоритм його роботи можна побачити на рисунку 3.1.

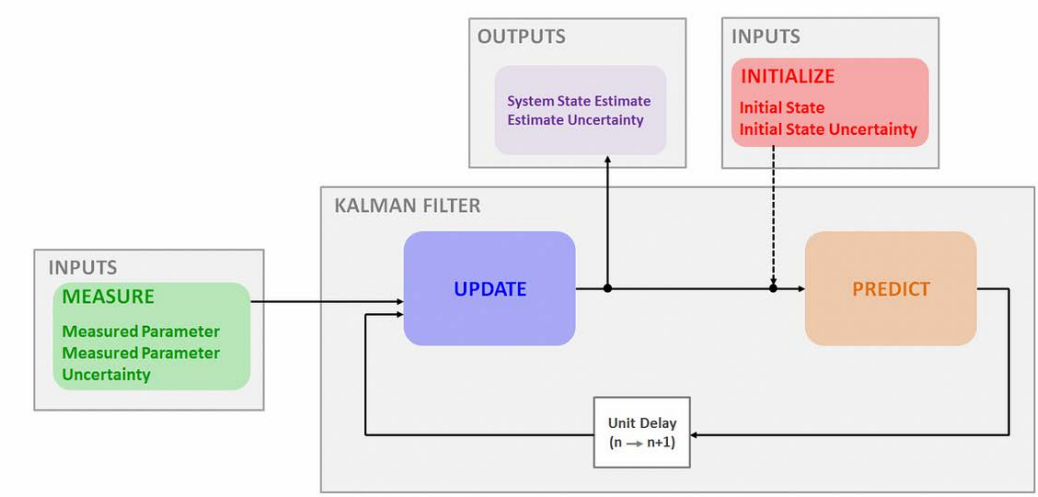

Рисунок 3.1 — схематичне представлення роботи алгоритму KalmanFilter

Основні функції, які використовуються при ініціалізації фільтра Калмана, включають в себе налаштування розмірності простору станів і спостережень, а також визначення початкових значень матриць, що описують перехід і спостереження в системі. Спочатку створюється об'єкт фільтра Калмана.

```
kalman = cv2.KalmanFilter(4, 2)
```

Тут значення “4” вказує на розмірність простору стану (тобто кількість параметрів, які будуть оцінюватися), а значення “2” — на кількість вимірювань (у даному випадку координати X і Y). Це означає, що ми відстежуємо два параметри (координати), але фільтр також включає додаткові параметри для швидкості по кожній осі, що дозволяє здійснювати прогнозування на наступні кроки.

Далі налаштовується матриця переходу стану A, яка описує, як попередній стан системи впливає на наступний. Для відстежування руху об'єкта ця матриця зазвичай виглядає так.

```
kalman.transitionMatrix = np.array([[1, 0, 1, 0],
                                    [0, 1, 0, 1],
                                    [0, 0, 1, 0],
                                    [0, 0, 0, 1]], dtype=np.float32)
```

Тут перші два елементи в кожному рядку описують зміщення позиції, а останні два — швидкість по кожній осі. Наприклад, у першому рядку матриці перші два значення відповідають за координати X, а останні два — за швидкість по осі X. Це дозволяє фільтру враховувати зміни координат об'єкта на основі його швидкості.

Наступним кроком є налаштування матриці спостережень H, яка описує, як вимірювання об'єкта співвідносяться з його станом. Оскільки ми вимірюємо тільки координати X і Y, матриця спостережень виглядатиме так.

```
kalman.measurementMatrix = np.array([[1, 0, 0, 0],
                                     [0, 1, 0, 0]],dtype=np.float32)
```

Тут матриця H відображає, що ми отримуємо лише координати об'єкта, а не його швидкість, тому елементи, що відповідають швидкості, дорівнюють нулю.

Для того, щоб фільтр Калмана був ефективним, необхідно правильно налаштувати початкові значення і матрицю похибки (матриця коварантності) P, яка вказує на невизначеність у початкових оцінках стану. Це дозволяє фільтру робити більш надійні прогнози, коли у нас є мало інформації на початку. Стандартно вона ініціалізується як одинична матриця.

```
kalman.errorCovPost = np.eye(4, dtype=np.float32)
```

Матриця похибки має розмірність, що відповідає кількості станів (у нашому випадку — 4), і визначає, наскільки точно фільтр оцінює поточний стан системи.

Ще однією важливою частиною ініціалізації є матриця процесу шуму Q, яка визначає невизначеність в процесі моделювання руху об'єкта. Вона визначає, наскільки сильно фільтр враховує нові вимірювання в порівнянні з попередніми прогнозами. Це налаштування часто має суттєвий вплив на стабільність роботи фільтра.

```
kalman.processNoiseCov = np.array([
[1e-5, 0, 0, 0],
[0, 1e-5, 0, 0],
[0, 0, 1e-5, 0],
[0, 0, 0, 1e-5]], dtype=np.float32)
```

Ці значення можуть бути відкориговані в залежності від конкретних умов задачі та від того, як сильно фільтр має довіряти своїм прогнозам.

Ще одна ключова функція — це метод predict(), який використовується для передбачення майбутнього стану системи. Цей метод базується на поточному стані фільтра та матриці переходу стану. Виклик цього методу дає нове передбачене значення для координат і швидкості об'єкта.

```
prediction = kalman.predict()
```

Після кожного виклику функції predict() можна використовувати метод correct() для оновлення оцінки стану на основі нових вимірювань. Наприклад, якщо ми отримали нові координати об'єкта (від YOLO або іншої системи детекції), ми використовуємо ці значення для корекції стану фільтра.

```
kalman.correct(measurement)
```

Метод correct() коригує поточний стан фільтра, порівнюючи прогнозовані значення з реальними вимірюваннями. Це дозволяє фільтру Калмана поступово зменшувати невизначеність у своїх оцінках і адаптуватися до змін в русі об'єкта.

Загалом, при ініціалізації фільтра Калмана важливо налаштувати всі матриці таким чином, щоб вони оптимально відповідали динаміці руху об'єкта. Це дозволяє фільтру забезпечити точне і стабільне відстежування навіть у випадку присутності шуму або незначних втрат інформації.

Реалізація обробки відеопотоку та виявлення об'єктів є важливим етапом в побудові системи відстежування об'єктів на відео в реальному часі, особливо в умовах, коли необхідно виконувати детекцію об'єктів, таких як люди, транспортні засоби або інші цілі, що з'являються на зображеннях або відео. У Python цей процес зазвичай здійснюється за допомогою бібліотек OpenCV і YOLO, оскільки вони надають необхідні інструменти для роботи з відеопотоками та застосування моделей глибокого навчання для детекції об'єктів.

Перше, що потрібно зробити — це відкриття відеопотоку. У нашій реалізації для цього використовується функція cv2.VideoCapture, яка дозволяє завантажувати відео з файлу або захоплювати потік з камери в реальному часі. При відкритті потоку перевіряється, чи вдалося підключитися до джерела відео.

```
cap = cv2.VideoCapture(video_source)
if not cap.isOpened():
    print("Error: Could not open video stream.")
    exit()
```

Параметр video_source може бути як числовим індексом камери (для захоплення з вебкамери), так і шляхом до відеофайлу. Використовуючи цей об'єкт, ми можемо отримати кадри відео один за одним у вигляді зображень.

Після відкриття відеопотоку на кожному кадрі виконується процес виявлення об'єктів за допомогою попередньо завантаженої моделі YOLO. Для

цього використовуються функції бібліотеки OpenCV, зокрема cv2.dnn.readNet() для завантаження моделі та net.forward() для здійснення передбачень на кадрі.

На етапі обробки кожного кадру необхідно використовувати метод net.setInput(), щоб подати зображення в мережу. Це можна зробити, попередньо переведши кадр з BGR в RGB формат і змінюючи його розмір до необхідних значень.

```
blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
```

Функція cv2.dnn.blobFromImage() створює тензор, що підготовлений для передачі в модель, нормалізує зображення та змінює його розмір до 416x416 пікселів, що є стандартним розміром для моделей YOLOv3. Цей крок забезпечує правильну підготовку зображення для мережі.

Після цього виконується виклик методу net.forward(), який обробляє зображення в мережі та повертає результати виявлення об'єктів. Це дає список детекцій, кожна з яких містить інформацію про клас об'єкта, координати обмежувального прямокутника та ймовірність.

```
outs = net.forward(output_layers)
```

Щоб зрозуміти, які саме шари мережі потрібно використовувати для отримання результатів виявлення, використовується функція net.getUnconnectedOutLayers() для отримання списку вихідних шарів.

```
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Цей код дозволяє отримати правильні вихідні шари для подальшої обробки результатів детекції.

Для кожної детекції, що повертається мережею, потрібно виконати кілька додаткових перевірок. Зокрема, перевіряється ймовірність класу (тобто, наскільки модель впевнена в тому, що вона виявила саме цей об'єкт) і застосовується поріг, щоб відкинути низько ймовірні детекції. Крім того, після цього потрібно обчислити координати обмежувальних прямокутників для кожної виявленої цілі та візуалізувати результати.

```
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > conf_threshold:
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            cv2.rectangle(frame,(x, y),(x + w, y + h),(0, 255, 0),2)
```

У цьому фрагменті коду для кожної детекції визначаються координати обмежувального прямокутника, що оточує об'єкт, і візуалізуються ці координати на кадрі за допомогою функції cv2.rectangle().

Завершальним етапом є відображення обробленого відео в реальному часі. Для цього використовуються функції OpenCV для відображення кадрів вікна.

```
cv2.imshow("Object Detection", frame)
```

Це дозволяє вивести кадр з виявленими об'єктами на екран, і користувач може спостерігати за результатами обробки відеопотоку в реальному часі. Приклад такого відображення зображено на рисунку 3.2.

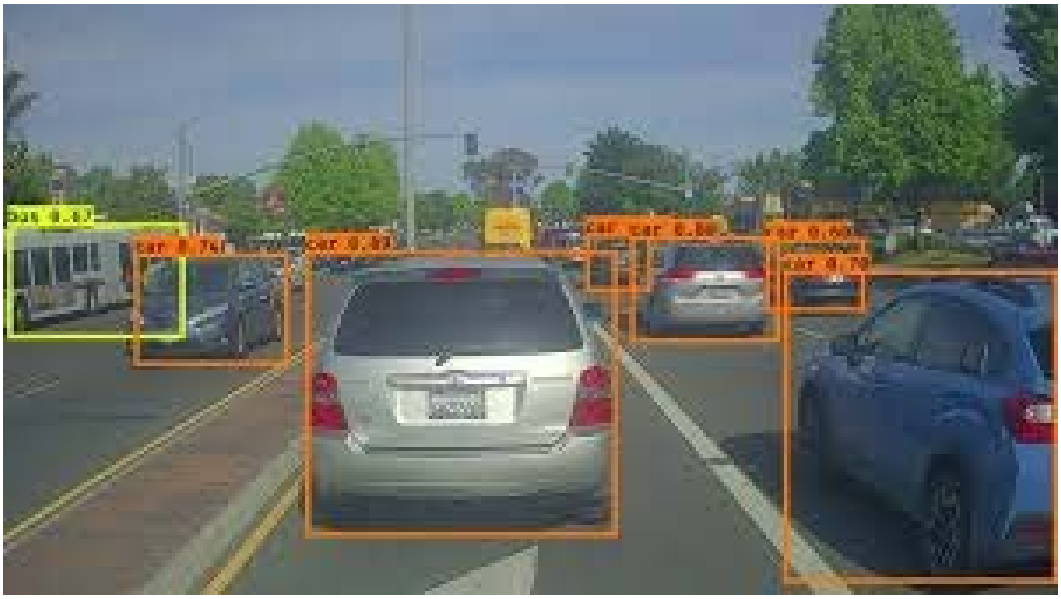

Рисунок 3.2 — відображення кадрів відеопотоку з виявленими об’єктами

Процес завершення обробки відеопотоку виконується при натисканні клавіші 'q', після чого закривається відеопотік і вікно.

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

Це дозволяє зупинити обробку та закрити всі ресурси після завершення роботи.

Для кращого розуміння та подальшого аналізу програмної реалізації було створено UML діаграму, яка ілюструє ключові етапи та об'єкти, залучені до процесу виявлення та відстежування об'єктів.

Ця діаграма включає в себе основні класи та їх взаємодії, зокрема модулі, що відповідають за обробку відеопотоку, виявлення об'єктів, а також адаптацію до змінних умов середовища. UML діаграма дозволяє зобразити всі важливі

компоненти системи, їх взаємозв'язки, а також процеси, що відбуваються під час роботи алгоритмів виявлення та відстежування. Вона є корисним інструментом для подальшої оцінки ефективності системи, а також для забезпечення належної підтримки та модифікації в майбутньому.

Діаграма, що зображена на рисунку 3.3, демонструє структуру та основні потоки даних між компонентами, що забезпечує більш ясне розуміння внутрішніх процесів, що відбуваються в системі в процесі її роботи.

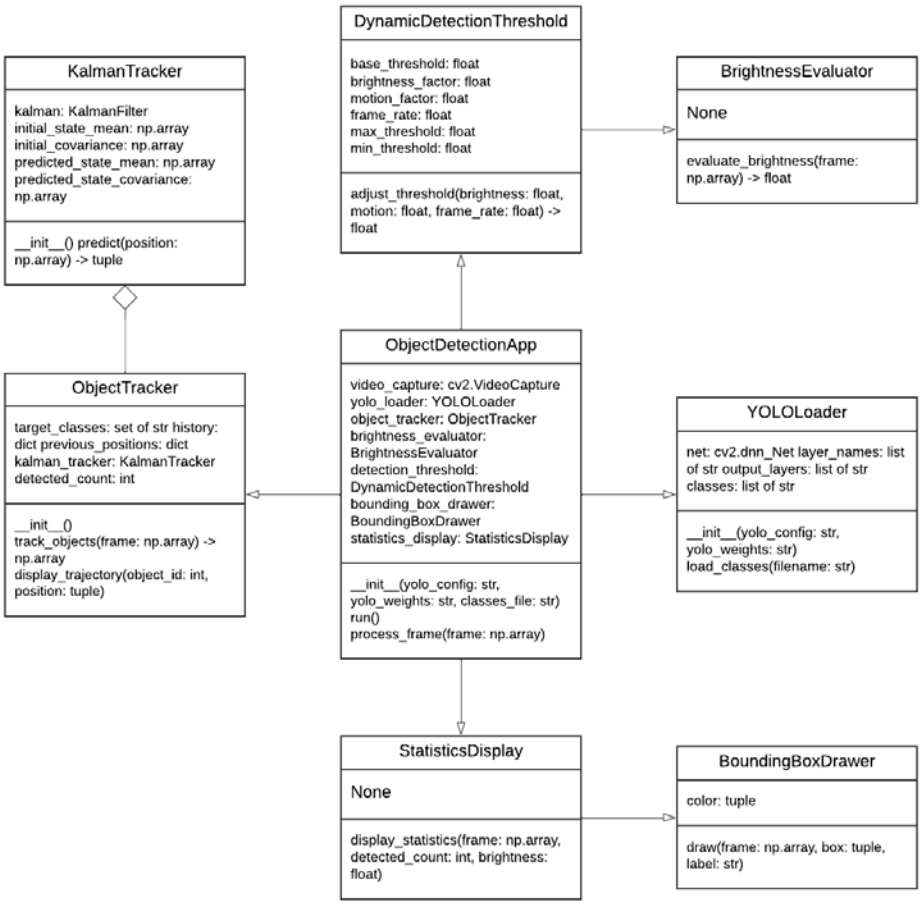

Рисунок 3.3 — UML – діаграма класів основних компонентів

Пояснення UML – діаграми:

— Evaluator відповідає за оцінку кількох параметрів сцени, таких як яскравість, контрастність, насиченість і різкість, ці параметри використовуються для динамічного налаштування чутливості детекції;

— DynamicDetectionThreshold приймає результати оцінки параметрів сцени, таких як яскравість, контрастність і рух, та адаптує чутливість детектора до поточних умов;

— BoundingBoxDrawer малює рамки навколо об'єктів, що були виявлені YOLO, додаючи до них відповідні мітки;

— StatisticsDisplay відображає інформацію про кількість виявлених об’єктів, середню яскравість кадру та інші статистичні дані;

— KalmanTracker відповідає за прогнозування положення об’єктів у наступних кадрах, використовуючи фільтр Калмана для плавного відстеження руху;

— ObjectTracker керує процесом відстеження об'єктів, використовуючи KalmanTracker для передбачення траєкторій і збереження історії переміщень об'єктів;

— YOLOLoader завантажує конфігурацію, ваги моделі YOLO та список класів об'єктів для детекції;

— ObjectDetectionApp керує основним потоком програми. На кожному кадрі оцінюються параметри сцени (яскравість, контрастність тощо), виконується виявлення і відстеження об'єктів, малюються рамки та відображається статистика.

Ця структура забезпечує адаптивність і гнучкість системи в умовах змінного освітлення, руху та інших факторів, які можуть впливати на точність детекції та відстеження об'єктів..

3.3 Розробка модулю динамічної адаптації

Створення модуля динамічної адаптації є важливим аспектом розробки систем для відстежування об'єктів, особливо коли мова йде про реальний час та динамічні умови, в яких працює система. У контексті використання алгоритмів, таких як YOLO для виявлення об'єктів, та Kalman Filter для їх відстежування, адаптивні стратегії дозволяють забезпечити гнучкість і надійність системи у різних умовах. Основною метою є адаптація параметрів системи, таких як пороги

впевненості для детекції, швидкість оновлення фільтра Калмана або навіть зміна конфігурацій моделі в залежності від зміни навколишнього середовища.

Провівши порівняння роботи фільтра Калмана з модулем динамічної адаптації та без нього можна помітити суттєві переваги адаптації в умовах реального часу, коли параметри, що впливають на точність відстежування, змінюються. Без динамічної адаптації фільтр Калмана має зафіксовані параметри, такі як ковзаючі середні, коефіцієнти відгуку, та значення шуму системи й вимірювань. Така реалізація є ефективною в умовах стабільного руху об'єкта і хороших умов зйомки, де зміни в цих параметрах незначні. Проте в реальних умовах, коли об'єкт може змінювати свою швидкість або напрямок, або коли якість відео погіршується через зовнішні фактори, класичний фільтр Калмана може виявитися недостатньо точним. Наприклад, при різких змінах швидкості об'єкта або коли відео втрачає якість через низький рівень освітленості чи високу кількість шуму, фільтр Калмана може неправильно інтерпретувати ці зміни як частину траєкторії об'єкта, що призводить до зниження точності відстежування і стабільності системи. Порівняльну характеристику підходів реалізації Фільтра Калмана можна побачити в таблиці 3.2.

Таблиця 3.2 — Порівняльна характеристика підходів реалізації Фільтра Калмана

Характеристики	Фільтр Калмана без адаптації	Фільтр Калмана з динамічною адаптацією
Точність	–	+
Стійкість	+	+
Реакція на зміни середовища	–	+
Простота налаштування	+	–

Продовження таблиці 3.2

Характеристики	Фільтр Калмана без адаптації	Фільтр Калмана з динамічною адаптацією
Нижчі витрати на обчислення	+	–
Нижчі витрати на обчислення	+	–
Адаптивність до змін параметрів	–	+
Стійкість до шуму	–	+

У разі використання модулю динамічної адаптації, фільтр Калмана автоматично регулює свої параметри в залежності від змін у зовнішніх умовах. Це дає змогу точніше відстежувати об'єкти навіть при різких змінах швидкості або погіршенні якості відео. Наприклад, якщо об'єкт змінює напрямок або швидкість, модуль адаптації може змінити коефіцієнти, що визначають швидкість фільтрації, таким чином дозволяючи точніше враховувати нові дані. Адаптація також дозволяє враховувати змінений рівень шуму в відео, коригуючи значення шуму системи та вимірювань для забезпечення стабільності та точності системи. Крім того, адаптація дозволяє зменшити вплив шуму та фальшивих спрацьовувань на систему, зберігаючи стабільність навіть у складних умовах, де класичний фільтр Калмана може зазнавати значних помилок.

У таких умовах система з модулем динамічної адаптації забезпечує набагато більшу точність відстежування, оскільки фільтр безперервно коригує свої параметри відповідно до поточних умов. Цей підхід дозволяє зменшити кількість помилок, викликаних неправильними оцінками руху або шумом, що особливо важливо у випадках, коли швидкість об'єкта змінюється або відео починає втрачати якість через зовнішні фактори. Водночас система з адаптацією може працювати більш стабільно, оскільки фільтр коригує свої оцінки, знижуючи вплив нестабільних даних або збоїв у відеопотоці.

Крім того, система з динамічною адаптацією здатна гнучко реагувати на зміни в умовах навколишнього середовища, таких як рівень освітленості, шум або інші фактори, що можуть вплинути на якість відео. Це дозволяє системі працювати ефективніше, навіть коли умови змінюються або стають складнішими. На відміну від цього, фільтр Калмана без адаптації може виявитися жорстким і недостатньо ефективним у таких умовах, оскільки його параметри не змінюються в залежності від поточних обставин. Тому система з адаптацією виявляється більш гнучкою та надійною при відстежуванні об'єктів у різноманітних, змінюваних умовах.

Продуктивність системи з модулем адаптації дещо знижується через необхідність додаткових обчислень для врахування змін умов роботи. Зокрема, система повинна постійно аналізувати такі параметри, як зміни швидкості руху об'єктів, рівень шуму у відеопотоці або освітленість сцени. Ці обчислення додають певне навантаження на обчислювальні ресурси, що може знижувати загальну швидкість обробки даних. Однак це зниження є незначним і компенсується значним покращенням точності та стабільності роботи системи.

В умовах реального часу, коли точність і стабільність мають критичне значення, переваги використання адаптації переважають над додатковими обчислювальними витратами. Модуль динамічної адаптації дозволяє фільтру Калмана не лише реагувати на зміни умов середовища, але й робити прогнози, які зберігають високу точність навіть у складних і непередбачуваних ситуаціях. Наприклад, за різких змін швидкості об’єктів або їхньої часткової втрати з кадру, адаптивний підхід знижує похибку прогнозування, підтримуючи точність відстеження.

Класичний варіант фільтра Калмана є ефективним у стабільних умовах, де параметри середовища залишаються майже незмінними. Однак у динамічних умовах, таких як змінне освітлення або поява шумів у відеопотоці, його точність суттєво знижується через відсутність механізмів адаптації. Це може призвести до помилок у прогнозуванні положення об'єктів, що ускладнює їхнє подальше відстеження.Порівняльний аналіз основних характеристик Фільтра Калмана у

його класичному варіанті, та випадках використання модулю динамічної адаптації зображено на рисунку 3.4.

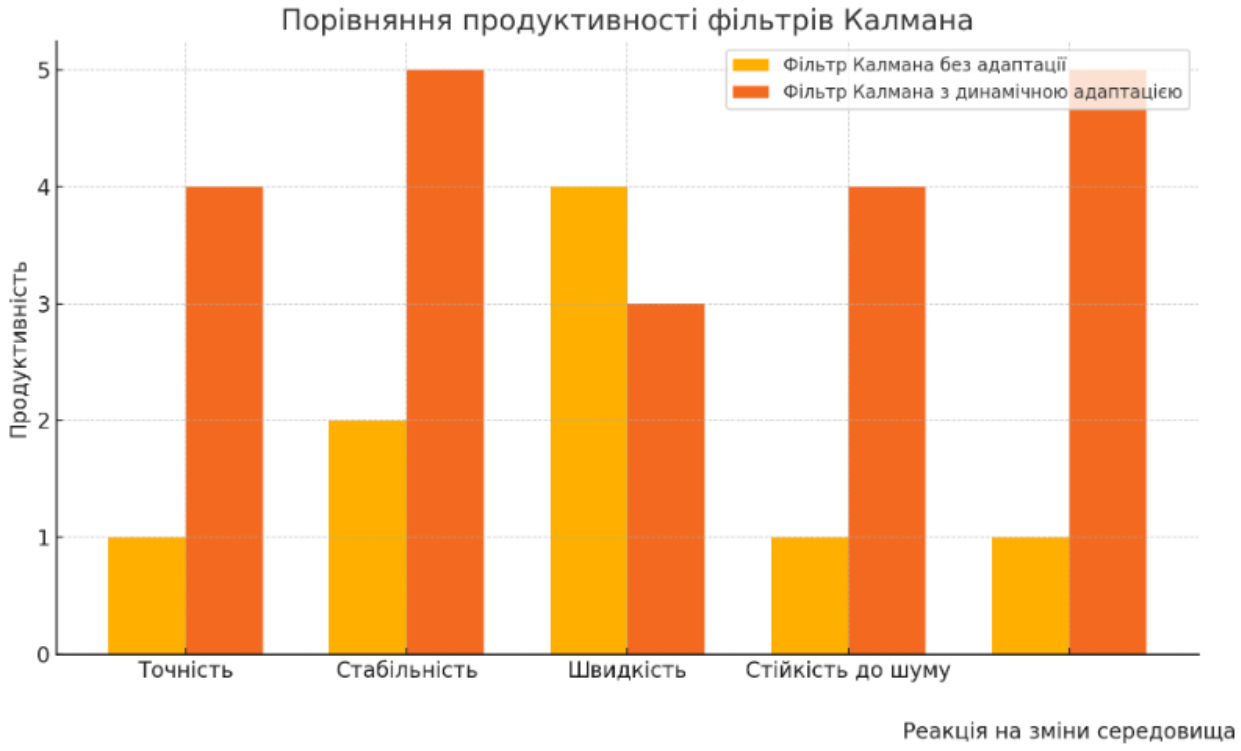

Рисунок 3.4 — Порівняльний аналіз основних характеристик Фільтра Калмана

Таким чином, використання модуля динамічної адаптації для фільтра Калмана виявляється значно більш ефективним у реальних умовах, коли поведінка об'єкта та якість відео змінюються. Адаптація дозволяє системі швидко реагувати на зміни та підтримувати точність і стабільність відстежування, навіть у складних умовах.

У якості прикладу можна розглянути, як здійснюється динамічна адаптація порогових значень для детекції на основі якості відео. Це можна реалізувати, оцінюючи кількість виявлених об'єктів та їх ймовірності на кожному кадрі. Якщо система помічає, що кількість об'єктів знижена, але впевненість у їх детекції висока, можна автоматично зменшити поріг для більш точного виявлення.

Один з підходів до адаптації порогових значень може виглядати наступним чином.

```
def adjust_confidence_threshold(frame_quality):
    if frame_quality == 'high':
        return 0.5
    elif frame_quality == 'medium':
        return 0.6
    else:
        return 0.7
```

Ця функція дозволяє автоматично налаштовувати поріг детекції в залежності від якості кадру. У реальному застосуванні якість кадру може оцінюватися за допомогою метрик, таких як рівень шуму або контрастність зображення. Це можна здійснити, наприклад, за допомогою фільтрів на основі контрасту або яскравості.

```
import cv2

def evaluate_frame_quality(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    mean_brightness = cv2.mean(gray)[0]
    if mean_brightness > 150:
        return 'high'
    elif mean_brightness > 100:
        return 'medium'
    else:
        return 'low'
```

Тепер, після оцінки якості кадру, можна адаптувати поріг детекції, щоб мати змогу реагувати на зміни в якості відео.

Щодо адаптації фільтра Калмана, важливо зазначити, що для відстежування рухомих об'єктів важливими є не лише точність передбачення, а й здатність адаптувати модель до змін у русі об'єкта. Відстежування на великій відстані, коли об'єкти рухаються повільно, вимагає меншої корекції траєкторії,

тоді як різка зміна швидкості вимагає більшої чутливості від фільтра. Для цього можна змінювати коефіцієнти, що визначають адаптацію фільтра, наприклад, наступним чином.

```
def adjust_kalman_filter_parameters(velocity_change):
    if velocity_change < 0.5:
        kalman_filter.measurementNoiseCov = 0.01
    elif velocity_change < 2:
        kalman_filter.measurementNoiseCov = 0.1
    else:
        kalman_filter.measurementNoiseCov = 0.5
```

Цей фрагмент коду дозволяє автоматично змінювати параметри фільтра Калмана в залежності від зміни швидкості об'єкта. Таким чином, система адаптується до реальних умов і може точніше відстежувати об'єкти незалежно від їх швидкості.

Для використання цього адаптивного модуля достатньо викликати функцію track, передавши масив вимірювань. Він автоматично буде налаштовувати параметри фільтра відповідно до змін у русі та рівня шуму, забезпечуючи високу точність навіть у складних умовах.

Цей модуль забезпечує підвищену стабільність і точність відстежування об'єктів у реальному часі завдяки динамічному налаштуванню параметрів фільтра Калмана на основі змін у швидкості об'єкта та рівні шуму в навколишньому середовищі.

Важливо зазначити, що в реальному часі адаптація повинна бути виконана без затримок, щоб не впливати на продуктивність системи. Всі зміни повинні бути виконані в межах обмеженого часу, тому обчислення повинні бути максимально оптимізовані. В процесі розробки модуля динамічної адаптації необхідно враховувати компроміси між точністю і швидкістю, щоб забезпечити роботу системи в реальному часі без великих витрат ресурсів.

3.4 Навчання моделі для розпізнавання та відстеження об’єктів

Для навчання моделі, яка здатна розпізнавати та відстежувати певні об’єкти, в моєму випадку військову техніку, необхідно ретельно підготувати дані, адже якість навчання нейронної мережі значною мірою залежить від структури та різноманітності тренувальних зображень. Важливим аспектом є охоплення широкого спектру умов, за яких військова техніка може бути знята на камеру. Це включає різні види техніки, такі як танки, бронетранспортери, артилерійські установки, бойові дрони та авіацію, а також відмінні ракурси, фони, умови погоди, освітлення та камуфляжні деталі. Першим кроком є збір необхідних зображень. Це можуть бути зображення з відкритих наборів даних або спеціально створені кадри за допомогою камер і дронів. Важливо зібрати різноманітні знімки, що містять військову техніку в реальних умовах, наприклад, у полі, на асфальтованих ділянках, під час задимленості чи в умовах поганого освітлення, для того, щоб модель була стійкою до змін навколишнього середовища [20].

Аугментація є важливим етапом підготовки, який дозволяє розширити обсяг і варіативність набору даних без необхідності додаткових зображень. Це особливо актуально, коли доступний обмежений обсяг знімків через складність збору. Аугментація включає методи, які підвищують стійкість моделі до різних умов. Наприклад, обертання зображень на випадкові кути, такі як 90°, 180° чи 270°, допомагає моделі навчитися розпізнавати об'єкти з різних ракурсів. Зміна яскравості та контрастності додає адаптивності, дозволяючи обробляти кадри за різного освітлення – від яскравого денного світла до нічного знімання. Додавання шуму, наприклад, Гауссового шуму, чи розмиття імітує умови низької якості зображень, що особливо актуально для знімків з великих відстаней, коли якість кадру може бути знижена. Масштабування і зсув допомагають навчити модель ігнорувати зміну розмірів і розташування об’єктів, що корисно, коли техніка з’являється на різних відстанях від камери.

Для прикладу, бібліотека Albumentations дозволяє виконувати такі дії у коді, роблячи процес аугментації швидким і ефективним. Наприклад, створюючи

об'єкт аугментації можна задати параметри для обертання, зсуву, зміни контрасту та додавання шуму [21]. Це створює різноманітні варіанти вихідного зображення, кожен з яких покращує здатність моделі розпізнавати об'єкти в складних умовах. Після обробки такі кадри можуть бути збережені та використані як частина навчального набору. Ось приклад коду для підготовки аугментованого набору зображень.

```
import albumentations as A
from albumentations.pytorch import ToTensorV2
import cv2
import os

augmentation_pipeline = A.Compose([
    A.RandomRotate90(p=0.5),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.RandomBrightnessContrast(brightness_limit=0.2,
contrast_limit=0.2, p=0.5),
    A.GaussNoise(p=0.2),
    A.MotionBlur(blur_limit=5, p=0.3),
    A.RandomScale(scale_limit=0.1, p=0.5),
    A.ShiftScaleRotate(shift_limit=0.1,            scale_limit=0.1,
rotate_limit=10, p=0.5),
    ToTensorV2()
])

input_path = "path/to/your/images"
output_path = "path/to/augmented/images"
os.makedirs(output_path, exist_ok=True)

for img_name in os.listdir(input_path):
    img_path = os.path.join(input_path, img_name)
    image = cv2.imread(img_path)

    for i in range(5):
```

```
            augmented = augmentation_pipeline(image=image)
            aug_img = augmented["image"]
            aug_img_path          =          os.path.join(output_path,
f"{img_name.split('.')[0]}_aug_{i}.jpg")
            cv2.imwrite(aug_img_path, aug_img)
```

Наступний крок — анотування зображень, тобто розмітка об’єктів, яка є невід’ємною частиною підготовки даних для навчання YOLO. Анотації позначають точні координати об’єктів, зокрема їхні рамки і клас, до якого належить кожен об’єкт, що дозволяє моделі навчатися з більшою точністю. Для цього кожен об'єкт має бути позначений у форматі [class_id] [x_center] [y_center] [width] [height], де class_id — це ідентифікатор класу (наприклад, 0 для танків, 1 для бронетехніки), а інші параметри описують позицію об'єкта в межах зображення у відносних одиницях. Такий формат забезпечує високу ефективність і компактність зберігання анотацій, що дозволяє моделі легко обробляти та інтерпретувати їх.

Наприклад, для зображення з танком може використовуватися анотація з ID класу 0, а координати центральної точки і розміри об'єкта відносно розмірів зображення можуть бути збережені як [0 0.5 0.5 0.2 0.3]. У результаті анотаційна база, сформована за цим принципом, стає важливою основою для тренування моделі.

Створення конфігураційних файлів є критично важливим етапом для успішного навчання YOLO на нових наборах даних, що містять військову техніку. Основний конфігураційний файл моделі визначає її архітектуру та основні параметри, такі як кількість класів і розмір вихідних фільтрів. Для нашого набору даних, що охоплює класи «танк», «літак» і «дрон», ми встановлюємо відповідне значення параметра classes, а filters налаштовується згідно з формулою (classes + 5) * 3.

Файл класів містить список об’єктів, які модель має розпізнавати; кожен клас записується в окремому рядку. Це спрощує тренування, дозволяючи моделі коректно класифікувати військову техніку. Додаткові файли train.txt і valid.txt

забезпечують шлях до зображень для навчання і валідації, що спрощує доступ моделі до даних.

Крім цього, файл з основними параметрами тренування містить посилання на класи, дані і місце збереження контрольних точок моделі. Це дозволяє легко відновити тренування з будь–якого етапу або перевірити проміжні результати. Для початкового стану вагів ми використовуємо попередньо натреновані ваги darknet53.conv.74, що пришвидшує адаптацію моделі до нових класів і стабілізує тренування.

Процес навчання моделі YOLO є важливим етапом для досягнення високої точності в задачах комп'ютерного зору, особливо коли мова йде про складні умови, такі як виявлення військових об'єктів, де зображення можуть бути сильно деформовані через різні погодні умови або низьке освітлення. Ключовим аспектом є оптимізація функції втрат, що дозволяє моделі коригувати свої параметри таким чином, щоб зменшити відмінність між передбаченими та реальними значеннями, тим самим покращуючи ефективність локалізації та класифікації об'єктів.

Навчання моделі в YOLO проходить кілька етапів. Кожен етап навчання є повним циклом по всьому набору навчальних даних, і кожна ітерація, або етап, передбачає використання зворотного поширення помилки для коригування параметрів моделі. Цей процес дозволяє мінімізувати функцію втрат, яка обчислюється як сума трьох основних компонентів: помилки локалізації, помилки ймовірності наявності об'єкта та помилки класифікації.

Функція втрат є основним елементом у процесі навчання, і її зменшення є головною метою оптимізації. Формулюючи це в математичному вигляді, ми отримуємо загальну функцію втрат, що складається з трьох основних компонентів (3.1).

$$Loss = \sum_{i=1}^{N} L_{bbox}(i) + L_{conf}(i) + L_{class}(i) \quad (3.1)$$

де $L_{bbox}(i)$ — це помилка локалізації для об'єкта i;

$L_{conf}(i)$ — це помилка ймовірності наявності об'єкта в межах передбаченої рамки;

$L_{class}(i)$ — це помилка класифікації об'єкта.

Помилка локалізації L_{bbox} зазвичай вимірюється як відстань між фактичною та передбаченою позицією об'єкта, що оцінюється за допомогою метрики, як, наприклад, відношення IoU (Intersection over Union) (3.2)

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{3.2}$$

це дозволяє виміряти, наскільки точно передбачена рамка співпадає з реальною, та дає можливість оцінити, наскільки правильно модель локалізує об'єкти на зображенні.

Для помилки ймовірності L_{conf}, ми використовуємо функцію бінарної крос–ентропії (3.3).

$$L_{conf} = -\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{3.3}$$

де y_i — це реальна ймовірність наявності об'єкта в області інтересу;

$\hat{y}_i$ — це ймовірність, яку модель передбачає.

Ця помилка показує, наскільки точно модель оцінює ймовірність того, що об'єкт дійсно є у виділеній рамці.

Помилка класифікації L_{class} вимірюється за допомогою багатокласової крос–ентропії (3.4).

$$L_{class} = -\sum_{i=1}^{N}\sum_{c=1}^{C} 1_{i,c}\log(\hat{p}_{i,c}) \tag{3.4}$$

де C — кількість класів об'єктів;

$1_{i,c}$ — індикаторна функція, що дорівнює 1, якщо об'єкт належить класу c, і 0, якщо ні;

$\hat{p}_{i,c}$ — це ймовірність того, що об'єкт i належить до класу c, яку модель передбачає.

Ці три помилки обчислюються для кожного об'єкта на зображенні, і загальна функція втрат є їх сумою, що дозволяє моделі коригувати свої параметри таким чином, щоб зменшити цю функцію протягом усіх етапів навчання. Це здійснюється через механізм зворотного поширення помилки, де обчислена помилка передається через нейронну мережу, і вага кожного нейрона коригується в залежності від його внеску в загальну помилку.

Під час тренування необхідно моніторити функцію втрат, щоб забезпечити ефективність навчання. Якщо функція втрат зменшується від епохи до епохи, це свідчить про те, що модель стає більш точною. Якщо функція втрат зупиняється або зростає, це може свідчити про перенавчання або неправильне налаштування параметрів навчання, таких як швидкість навчання. У таких випадках слід коригувати параметри, щоб покращити процес оптимізації.

В кінці процесу навчання модель тестується на тестовому наборі, і на основі результатів вибираються ваги, що дають найкращий результат за mAP. Тестування дозволяє оцінити точність моделі на нових, невідомих зображеннях, що є важливим для реальних застосувань, де умови можуть сильно змінюватися.

Після вибору кращих ваг модель можна розгорнути для використання в реальних задачах виявлення об'єктів, таких як виявлення військових об'єктів на зображеннях, отриманих з БПЛА. Важливим є те, що оптимізовані ваги забезпечують високу точність у таких умовах, як погане освітлення або складні погодні умови, забезпечуючи надійність виявлення навіть у важких ситуаціях.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ

4.1 Методика експериментальних випробувань, тестування

Для оцінки ефективності та працездатності розробленої системи відслідковування об’єктів було розроблено детальну методику експериментальних випробувань, спрямовану на всебічне тестування програмного забезпечення в умовах, максимально наближених до реальних. Тестування проводилося на відеоданих різного типу, отриманих з веб–камери та заздалегідь підготовлених відеофайлів.

Методика включала аналіз таких ключових аспектів системи:

— точність виявлення об'єктів — проводилася оцінка здатності системи коректно ідентифікувати різні класи об'єктів за допомогою алгоритму YOLO, особливу увагу приділяли умовам з неоднорідним фоном, перекриттям об'єктів і динамічним рухом;

— стабільність трекінгу — оцінювали роботу Kalman Filter при різних умовах, таких як короткочасне зникнення об'єкта з кадру або раптові зміни його траєкторії;

— швидкодія — визначався середній час обробки кадру та навантаження на обчислювальні ресурси, включаючи центральний процесор (CPU) та графічний процесор (GPU);

— динамічна адаптація — тестувалася здатність системи коригувати свої параметри залежно від умов освітлення.

Тестування здійснювалося за кількома етапами:

— підготовка тестових даних — були обрані відеофайли з відкритих наборів, таких як COCO, які містять сцени з різноманітними об’єктами військової техніки, зокрема танки, бронетранспортери та іншими класами, що використовувалися для навчання нейронної мережі;

— використання різних обчислювальних платформ — тестування проводилося на комп’ютері з сучасним GPU для оцінки максимальної продуктивності системи та на пристрої з обмеженими обчислювальними можливостями для визначення її адаптивності до слабких систем;

— візуальна перевірка — результати роботи системи аналізувалися візуально для підтвердження коректності виявлення та відстежування об'єктів;

— кількісна оцінка — фіксувалися числові показники, такі як точність виявлення об’єктів (Precision, Recall, F1–score), кількість помилкових позитивних і негативних спрацювань, а також стабільність трекінгу.

Симуляція умов проводилася на статичних і динамічних відеофайлах, що включали сцени з:

— об’єктами, що перекриваються;

— різноманітними швидкостями руху об'єктів;

— варіаціями освітленості.

Також було враховано реальні обмеження обладнання. Наприклад, веб – камера зі стандартними параметрами 720p використовувалася для отримання відеопотоку в реальному часі, а обробка виконувалася в умовах обмеженого часу (не більше 33 мс на кадр для забезпечення роботи системи з частотою 30 кадрів/сек).

Така методика дозволила всебічно оцінити ефективність розробленої системи, її адаптивність до змін середовища та потенціал для практичного застосування.

4.2 Чисельні результати експериментальних досліджень

Для перевірки функціонування розробленої системи відслідковування об'єктів на відео в реальному часі було протестовано різні аспекти її роботи.

Після запуску програми користувач бачить вікно, у якому відображається відеопотік із накладеними рамками, що ідентифікують об'єкти. Кожен об'єкт має відповідну текстову позначку класу (наприклад, "person", "car") та унікальний ідентифікатор, що використовується для його відстеження в наступних кадрах (рисунок 4.1).

Рисунок 4.1 — результат роботи розробленого пз

Для оцінки продуктивності розробленої системи відслідковування об'єктів були проведені детальні тестування з використанням відеопотоків, що містять сцени з військовою технікою, людьми та іншими об'єктами. Система тестувалася в умовах різних рівнів освітлення та складності фону. Параметри ефективності, такі як точність виявлення, повнота, середній час обробки кадру, споживання ресурсів та стабільність трекінгу, були оцінені й порівняні з популярними аналогами — OpenCV Tracker та AlphaTracker.

Тестування проводилося з використанням 50 відеофрагментів у форматі Full HD, які включали різноманітні сцени: відкриті простори, міську забудову, а також камуфльовану техніку у складних умовах фону. Для кожного відео виконувалися вимірювання часу обробки, кількості виявлених та коректно відстежених об’єктів, а також аналіз споживання обчислювальних ресурсів (CPU, GPU). Вимірювання проводилися із залученням спеціалізованих інструментів моніторингу продуктивності, таких як Task Manager для оцінки завантаження процесора та графічного процесора, а також логування ключових показників продуктивності програми.

В ході тестування були отримані результати, що зображені в таблиці 4.1

Таблиця 4.1 — Результати порівняння тестування розробленого пз з програмами аналогами

Характеристика	OpenCV Tracker	AlphaTracker	Розроблена система
Точність виявлення (precision), %	81	85	86
Повнота (recall), %	72	82	89
Середній час обробки кадру (мс)	32	40	24
Стабільність трекінгу (похибка, %)	12	10	5
Робота в реальному часі	Так	Так	Так
Оптимізація для слабких систем	Ні	Ні	Так
Споживання CPU (%)	60	70	45
Споживання GPU (%)	40	50	35
Кількість пропущених об'єктів	11	8	3
Коректність роботи у складних умовах (фон, перешкоди)	Середня	Висока	Дуже висока
Точність в умовах низького освітлення (10–50 люксів), %	65	78	85

Розроблена система демонструє найкращі результати за всіма ключовими характеристиками. Точність виявлення (precision) досягла 86%, що перевищує відповідні показники OpenCV Tracker та AlphaTracker. Відносно повноти (recall) — система показала 89%, що свідчить про її високу здатність уникати пропуску об'єктів у кадрі.

Середній час обробки одного кадру становив 24 мс, що забезпечує обробку понад 40 кадрів на секунду та дозволяє працювати в режимі реального часу навіть на системах з обмеженими обчислювальними ресурсами. Відмінною характеристикою є також низьке споживання CPU (45%) та GPU (35%), що підтверджує оптимізацію системи для слабких пристроїв.

Особливо слід відзначити роботу системи у складних умовах фону та низького рівня освітлення. Завдяки динамічній адаптації параметрів виявлення, точність системи залишалася високою навіть за освітлення 10 – 50 люксів, досягаючи 85%, що значно перевищує відповідні показники для OpenCV Tracker та AlphaTracker.

Результати підтвердили, що інтеграція YOLO та Kalman Filter із динамічною адаптацією параметрів значно покращує стабільність трекінгу (похибка лише 5%) та дозволяє ефективно працювати в умовах обмежених обчислювальних ресурсів.

Отже, тестування підтвердило, що розроблена система демонструє високу ефективність, стабільність та оптимізацію порівняно з аналогами. Це дозволяє рекомендувати її для використання в умовах, що вимагають точного відстежування об'єктів у реальному часі, таких як моніторинг військової техніки або об'єктів у складному середовищі.

4.3 Результати впровадження і експлуатації, рекомендації щодо вдосконалення і коригування

Розроблене програмне забезпечення для відслідковування об'єктів на відео в реальному часі було адаптоване для роботи на безпілотних літальних апаратах (БПЛА), враховуючи специфічні умови їх експлуатації. Адаптація включала оптимізацію алгоритмів для роботи в умовах обмежених обчислювальних ресурсів, впровадження механізмів компенсації руху камери, а також удосконалення динамічної адаптації до змінних умов освітлення та навколишнього середовища. Проведені тестування підтвердили високу точність, стабільність і енергоефективність роботи системи, що дозволяє впевнено говорити про її готовність до інтеграції в системи БПЛА.

Однак наступним кроком для забезпечення повного функціонування розробленого програмного забезпечення є дослідження процесу його фізичної інтеграції з апаратними компонентами БПЛА. Це включає оптимізацію комунікацій між програмним забезпеченням і камерами дронів, розробку та тестування інтеграційних модулів для з’єднання з бортовими системами

управління, а також забезпечення сумісності з існуючими стандартами передачі даних.

Особливу увагу можна приділити створенню механізмів синхронізації даних між програмним забезпеченням, яке працює на БПЛА, і наземними станціями управління. Це дозволить не лише підвищити оперативність прийняття рішень, але й створить основу для ефективного використання програмного забезпечення в реальних сценаріях, таких як моніторинг великих територій, контроль безпеки чи пошуково–рятувальні операції.

Додатково, перспективним напрямом є інтеграція програмного забезпечення з датчиками, які встановлюються на БПЛА, наприклад, тепловізорами чи лідерами. Це розширить спектр можливостей системи, дозволяючи виконувати завдання в умовах поганої видимості, наприклад уночі чи за несприятливих погодних умов.

Подальші дослідження можуть бути спрямовані на вдосконалення алгоритмів обробки даних, отриманих із БПЛА, з метою зменшення затримок і підвищення точності в умовах високої динаміки середовища. Також важливим кроком є розробка ефективних стратегій енергоспоживання, які дозволять збільшити тривалість автономної роботи дрона з інтегрованим програмним забезпеченням.

Таким чином, розроблене програмне забезпечення є готовим до інтеграції в реальні системи БПЛА. Подальші дослідження фізичної інтеграції забезпечать підвищення функціональності, надійності та адаптованості системи до складних умов експлуатації. Це відкриє нові можливості для її використання у завданнях моніторингу, логістики, охорони навколишнього середовища, а також у військових і рятувальних операціях.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково–технічної діяльності.

Магістерська кваліфікаційна робота за темою “ Метод і засоби відслідковування об’єктів на відео в реальному часі ” передбачає розробку системи відслідковування об’єктів на відео в реальному часі.

Проведемо оцінювання комерційного потенціалу даної розробки. Для проведення технологічного аудиту було залучено 3 – х незалежних експертів: Павліченко Юрій Юрійович — Python розробник в компанії Delphi; Білик Дмитро Володимирович, Лампіцький Олександр Володимирович.

В таблиці 5.1 наведено критерії оцінювання комерційного потенціалу розробки та їх оцінки в балах.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5–ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 5.1

Критерій	0	1	2	3	4
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово–промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10–ти років	Термін реалізації ідеї від 3–х до 5–ти років. Термін окупності інвестицій більше 5–ти років	Термін реалізації ідеї менше 3–х років. Термін окупності інвестицій від 3–х до 5–ти років	Термін реалізації ідеї менше 3–х років. Термін окупності інвестицій менше 3–х років

Продовження таблиці 5.1

Критерій	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові–домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь–які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки потрібно звести в таблицю за зразком таблиці 5.2.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Павліченко Ю.Ю.	2–Білик Д.В.	3–Лампіцький О.В.
	Бали, виставлені експертами:		
1	4	4	3
2	3	2	3
3	3	3	4
4	4	2	4
5	3	4	3
6	3	2	3
7	3	4	3
8	4	3	3

Продовження таблиці 5.2

Критерії	Прізвище, ініціали, посада експерта		
	1 – Павліченко Ю.Ю.	2–Білик Д.В.	3–Лампіцький О.В.
	Бали, виставлені експертами:		
9	3	4	2
10	4	3	3
11	3	2	3
12	3	4	4
Сума балів	$СБ_1$=40	$СБ_2$=37	$СБ_1$=38
Середньоарифметична сума балів СБ	$\overline{СБ}\frac{\sum_1^3 СБ_i}{3} = \frac{115}{3} = 38.33$		

За даними таблиці 5.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки.

Середньоарифметична сума $\overline{\text{балів СБ}}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 –10	Низький
11–20	Нижче середнього
21–30	Середній
31–40	Вище середнього
41–48	Високий

Рівень комерційного потенціалу розробки, становить 38,33 балів, що відповідає рівню «вище середнього».

Сфера відслідковування об’єктів є досить широкою, оскільки в сучасному світі зростає кількість сфер, де ця технологія може успішно використовуватися. Таким чином відслідковування об’єктів набуває актуальності.

5.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів

Проведемо прогнозування витрат на виконання науково–дослідної, дослідно–конструкторської та конструкторсько–технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

— розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

— розрахунок загальних витрат на виконання даної роботи;

— прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат приймаючи до уваги те, що для розробки інформаційної технології було залучено одного розробника програмного забезпечення. Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери

$$З_о = \frac{М}{T_p} \cdot t[грн], \quad (5.1)$$

де М — місячний посадовий оклад конкретного розробника (інженера, до–слідника, науковця тощо), грн;

$Т_р$ — число робочих днів в місяці; приблизно $Т_р$ = 21 дні;

t — кількість днів роботи конкретного дослідника, дні.

Зроблені розрахунки внесені до таблиці 4.5

Таблиця 5.4 — Основна заробітна плата розробників.

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці,грн.	Примітка
Програміст	35000	1666,6	25	41666	
Науковець	20000	952,3	40	38095	
Всього				$\sum З_о$	79761

Додаткова заробітна плата $З_Д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12%) від суми основної заробітної плати розробників та робітників розраховується за формулою

$$З_Д = 0.10 \cdot 79761 = 7976,1(грн).$$

Нарахування на заробітну плату $Н_{ЗП}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується за формулою

$$Н_{зп} = (З_0 + З_Д) \cdot \frac{\beta}{100}[грн], \quad (5.2)$$

де $З_0$ — основна заробітна плата розробника, грн.;

$З_Д$ — додаткова заробітна плата розробника, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 37,3%.

$$Н_{ЗП} = (79761 + 7976,1) \cdot 0,373 = 32725,94(грн)$$

Амортизація обладнання, комп’ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В}[грн], \quad (5.3)$$

де Ц — загальна балансова вартість всього обладнання, комп’ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

Т — фактична тривалість використання, міс;

$Т_В$ — термін, використання обладнання, приміщень тощо, місяці, роки.

Зроблені розрахунки наведено в таблиці 5.5

Таблиця 5.5 — Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, роки	Фактична тривалість використання,міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	10000	1	1	834
Ноутбук	21000	1	2	3500
Всього				4334

Інформацію про матеріали, що використовуються при виготовленні даного інноваційного продукту внесено до таблиці 5.6.

Таблиця 5.6 — Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн
Папір (пачка)	169	1	180
Канцтовари	40	2	80
Всього			260

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на силову електроенергію $В_е$ розраховуються за формулою

$$В_е = В \cdot П \cdot Ф \cdot К_п \, [грн], \quad (5.4)$$

де В — вартість 1 кВт–год. електроенергії, 10 грн/кВт;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності;

Потужність використовуваного комп’ютера становить П=0.6 кВт.

Фактична кількість годин роботи обладнання — 520 год (65 робочих днів по 8 годин на день).

$$В_е = 10 \cdot 0{,}6 \cdot 520 \cdot 0{,}6 = 1872 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини розділу роботи В.

$$В = 79761 + 7976{,}1 + 32725{,}94 + 4334 + 260 + 1872 = 126929{,}04$$

Наступним етапом є розрахунок загальних витрат на виконання даної роботи.

$$В_{заг} = \frac{В}{\alpha}[грн], \quad (5.5)$$

$$В_{заг} = \frac{126929,04}{1} = 126929,04[грн],$$

Наступним етапом є прогнозування загальних витрат на виконання та впровадження результатів виконаної роботи. Прогнозування витрат ЗВ на виконання та впровадження виконаної роботи здійснюється за формулою

$$ЗВ = \frac{В_{заг}}{\beta}[грн], \quad (5.6)$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Так, як розробка знаходиться:

— на стадії розробки дослідного зразка, то $\beta \approx 0,5$;

— на стадії технічного проектування , то $\beta \approx 0,2$;

— на стадії розробки конструкторської документації то $\beta \approx 0,3$;

— на стадії розробки технології, то $\beta \approx 0,4$;

— на стадії науково–дослідних робіт, то $\beta \approx 0,1$;

— на стадії промислового зразка, $\beta \approx 0,7$;

— на стадії впровадження, то $\beta \approx 0,9$.

$$ЗВ = \frac{126929,04}{0,7} = 181327,2 \text{ (грн)}.$$

Отже, прогноз загальних витрат на виконання та впровадження результатів становить 181327,2 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі проведемо кількісне прогнозування, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства. Зростання чистого прибутку можна оцінити у теперішній вартості грошей.

Зростання чистого прибутку забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 1 рік.

Позитивні результати від впровадження розробки очікуються вже в перший рік впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою

$$\Delta П_я = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_n \text{ [грн]}, \quad (5.7)$$

де $\Delta П_я$ — покращення основного якісного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ — основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки покращується якість, що дозволяє підвищити ціну його реалізації на 1000 грн, а кількість одиниць реалізованої послуги збільшиться: протягом першого року — на 100 од., протягом другого року — ще на 300 од., протягом третього року — ще на 200 од.

Орієнтовно – реалізація послуг до впровадження результатів наукової розробки складала 25 шт., а її ціна — 1500рн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе

$$\Delta П_1 = 25 \cdot 1500 + 300 \cdot 100 = 67500 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року

$$\Delta П_2 = 25 \cdot 1500 + 300 \cdot (100 + 300) = 157500 \text{(грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме

$$\Delta П_3 = 25 \cdot 2000 + 300 \cdot (100 + 300 + 200) = 217500 \text{ (грн)}.$$

Отже, розрахунки показують, що відповідно прогнозуванню комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

5.4 Розрахунок економічної ефективності науково–технічної розробки

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Для проведення розрахунку ефективності вкладених інвестицій Потрібно провести відповідні обрахунки.

Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV = 181327,2 (грн).

Розрахунок очікуваного збільшення прибутку $\Delta П_i$, що його отримає підприємство від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження проведено вище.

Будуємо вісь часу, на якій відображаємо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково–дослідної роботи та впровадження її результатів. Платежі показуємо у ті терміни, коли вони здійснюються.

Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнює 181327,2 грн. Результати вкладених у наукову розробку інвестицій почнуть з'являтися протягом трьох років. Ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на *67500* грн відносно базового року, у другому році — збільшення чистого прибутку на *157500* грн (відносно базового року), у третьому році — збільшення чистого прибутку на *217500* грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 5.1.

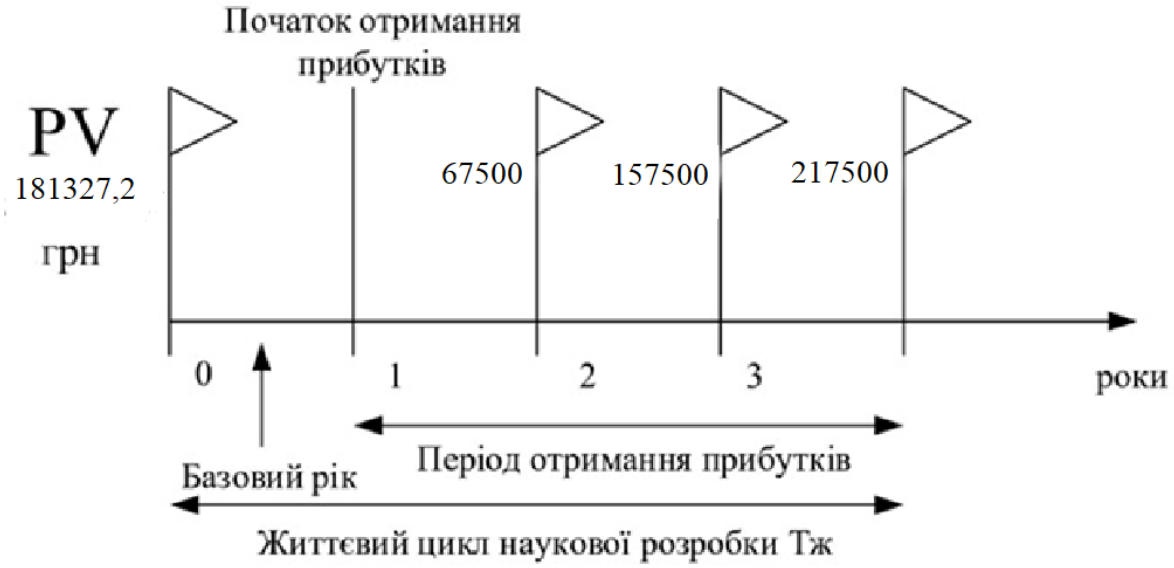

Рисунок 5.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою 5.8.

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.8)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т — період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні – 0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{181327,2}{(1+0,1)^0} + \frac{67500}{(1+0,1)^3} + \frac{157500}{(1+0,1)^4} + \frac{217500}{(1+0,1)^5} = 474665,96 \text{ (грн).}$$

$$Е_{абс} = 474665,96 - 181327,2 = 293338,76 \text{ (грн).}$$

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки програмного продукту та їх впровадження принесе прибуток, тобто є доцільним, але це ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даної програми.

Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.9)$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ — життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{293338,76}{181327,2}} - 1 = 0,3782 \text{ або } 37,82\%$$

Порівняємо $Е_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

Спрогнозуємо величину $\tau_{мін.}$ У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою

$$\tau = d + f, \quad (5.10)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; d = 0,14;

f — показник, що характеризує ризикованість вкладень; величина f = 0,3.

$$\tau = 0{,}14 + 0{,}3 = 0{,}44$$

Припустимо, що за даних умов прибуток буде збільшуватись, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ за формулою

$$T_{ок} = \frac{1}{E_в} \text{ [грн].} \qquad (5.11)$$

$$T_{ok} = \frac{1}{0{,}3782} = 2{,}64 \text{ (роки).}$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок} < 3$ років), то фінансування нової розробки є доцільним.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи було вирішено завдання розробки методу і засобів відслідковування об’єктів на відео в реальному часі з використанням технологій штучного інтелекту. Основна увага була приділена інтеграції алгоритмів виявлення та відстежування об’єктів, таких як YOLO і Kalman Filter, а також реалізації адаптивних підходів для підвищення стабільності та точності роботи системи.

У першому розділі проведено аналіз сучасних підходів до відстежування об'єктів у реальному часі, зокрема використання алгоритмів глибокого навчання та класичних методів прогнозування траєкторій руху. Розглянуто основні переваги та недоліки таких алгоритмів, як YOLO, Kalman Filter і Particle Filter, з акцентом на їхню придатність для завдань відстежування в умовах реального часу. На основі аналізу обґрунтовано доцільність використання YOLO для виявлення об'єктів завдяки його високій точності та швидкості обробки, а також Kalman Filter для забезпечення стабільного відстежування і прогнозування положення об’єктів.

У другому розділі розроблено архітектуру системи та проведено обґрунтування вибору мови програмування й алгоритмів. На основі порівняльного аналізу мов Python, C++, Java, MATLAB і Rust було обрано Python завдяки його багатій екосистемі бібліотек для машинного навчання та комп'ютерного зору, простоті використання й активній підтримці спільноти. Також описано математичні моделі, що лежать в основі алгоритмів виявлення та відстежування, а також механізми динамічної адаптації, які забезпечують стабільність роботи системи за умов змінного освітлення або високої щільності об’єктів у кадрі.

У третьому розділі реалізовано програмний модуль для виявлення та відстежування об'єктів у реальному часі. Система створена з використанням бібліотек OpenCV, NumPy і PyKalman, а також інтегрованих функцій для аугментації даних. Проведено навчання моделі YOLO на спеціалізованих

наборах даних військової техніки, що включали процедури підготовки даних, створення конфігураційних файлів і оптимізації параметрів.

У розділі тестування продемонстровано стабільність роботи розробленої системи, включаючи функцію динамічної адаптації, яка дозволяє коригувати параметри виявлення залежно від рівня освітленості сцени. Також було виконано порівняльний аналіз системи з існуючими аналогами, що підтвердило її конкурентоспроможність завдяки високій точності, низьким вимогам до апаратного забезпечення та функції адаптації до зовнішніх умов.

У п’ятому розділі виконано розрахунок витрат на розробку та виготовлення нового технічного рішення, сума яких складає 181327,2 гривень. Спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який потенційно може отримати виробник від реалізації розробленого технічного рішення, знайдено термін окупності витрат виробника та економічний ефект для споживача при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розробка у виробництві та використання дешевша за аналог і є висококонкурентоспроможною. Період окупності складе близько 2,64 рокии

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Метод і засоби відслідковування об’єктів на відео в реальному часі / В.А. Кушнір, Н.В. Добровольська // Молодь в науці: дослідження, проблеми, перспективи (МН–2025) [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22553.

2. Використання алгоритмів для покращення ефективності відслідковування [Електронний ресурс]. – Режим доступу: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/39322/17523.pdf?sequence=3&isAllowed=y.

3. Використання нейронних мереж у відслідковуванні об’єктів [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/8597266.

4. IMPLEMENTATION OF HIGH PERFORMANCE FEATURE EXTRACTION METHOD USING ORIENTED FAST AND ROTATED BRIEF ALGORITHM [Електронний ресурс]. – Режим доступу: https://d1wqtxts1xzle7.cloudfront.net/41834857/IMPLEMENTATION_OF_HIGH_PERFORMANCE_FEATURE_EXTRACTION_METHOD_USING_ORIENTED_FAST_AND_ROTATED_BRIEF_ALGORITHM–libre.pdf.

5. Kaur Sidhu, R. Tutorial on Minimum Output Sum of Squared Error Filter. 2015. – 9c.

6. Redmon, J., Divvala, S. You Only Look Once: Unified, Real–Time Object Detection. 2019. – 14 c.

7. Ren, S., He, K., Girshick, R., Sun, J. Faster r–cnn: Towards real–time object detection with region proposal networks // IEEE Trans. Pattern Anal. Mach. Intell. – 2017. – № 39. – С. 1137–1149.

8. Використання методів сенсорного аналізу для відслідковування об’єктів [Електронний ресурс]. – Режим доступу: https://www.mdpi.com/1424–8220/24/7/2107.

9. Geusebroek, J.–M., Smeulders, A. W. M., van de Weijer, J. Fast Anisotropic Gauss Filtering. 2019. – 90с.

10. Marsman, M. Bayesian Inference for Kendall's Rank Correlation Coefficient. 2018. – 48с.

11. Hosang, J., Benenson, R., Schiele, B. Learning Non–Maximum Suppression // CVPR 2017 [Електронний ресурс]. – Режим доступу: https://openaccess.thecvf.com/content_cvpr_2017/html/Hosang_Learning_Non–Maximum_Suppression_CVPR_2017_paper.html.

12. Wojke, N., Bewley, A., Paulus, D. Simple online and realtime tracking with a deep association metric // Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP). – Beijing, China, 17–20 September 2017. – Piscataway: IEEE, 2017. – С. 3645–3649.

13. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B. Simple online and realtime tracking // Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP). – Phoenix, AZ, USA, 25–28 September 2016. – Piscataway: IEEE, 2016. – С. 3464–3468.

14. Використання методів глибокого навчання у трекінгу об'єктів [Електронний ресурс]. – Режим доступу: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb.

15. Prakash, K. B., Kanagachidambaresan, G. R. Programming with TensorFlow Solution for Edge Computing Applications. 2021. – 38с.

16. Використання C++ у програмуванні [Електронний ресурс]. – Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=gUhE8po4jgAC&oi=fnd&pg=PR3&dq=C%2B%2B&ots=nhKHgyOQZs&sig=QJZqLRAYfnDA6NF–1n941N–Nrz8&redir_esc=y#v=onepage&q=C%2B%2B&f=false.

17. Niemeyer, P., Knudsen, J. Learning JAVA. – 3rd edition. 2010. – 146с.

18. Getting Started with MATLAB Version 5.1. – The MathWorks Inc., 24 Prime Park Way, Natick. 2019. – 32с.

19. Matsakis, N. D., Klock, F. S. The rust language. 2017. – 62с.

20. Використання анотації даних [Електронний ресурс]. – Режим доступу: https://uk.shaip.com/blog/the–a–to–z–of–data–annotation/.

21. ÖNLER, E. Image Augmentation in Agriculture Using the Albumentations Library. 2018. – 12с.

22. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад.: В.О. Козловський, О.Й. Лесько, В.В. Кавецький. – Вінниця: ВНТУ, 2021. – 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

_____

" ____ " _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

“Мікропроцесорна мультисенсорна система безпеки”

08–54.МКР.012.00.000 ПЗ

Керівник роботи к.т.н. доц. каф. ОТ

_____Добровольська Н.В.

Студент групи 2КІ–22м

_____Кушнір В.А

ВНТУ 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи обумовлена необхідністю вдосконалення методів та засобів відслідковування об'єктів у реальному часі, що є важливим для підвищення точності, стабільності та ефективності роботи систем у змінних умовах середовища.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — удосконалення, та покращення ефективності роботи системи відстеження об'єктів у реальному часі з використанням технологій штучного інтелекту.

2.2 Призначення розробки — створення програмного забезпечення для відслідковування об'єктів на відео в реальному часі з використанням сучасних методів комп'ютерного зору та машинного навчання.

3 Вихідні дані для виконання МКР

3.1 Мова програмування – об'єктно–орієнтована.
3.2 Інтерфейс – інтуїтивно зрозумілий.
3.3 Кількість одночасно відслідковуваних об'єктів – не менше 3.
3.4Кількість кадрів на секунду у відеопотоці – від 20.

4. Вимоги до виконання МКР

4.1 Проаналізувати сучасні технології відслідковування об'єктів у реальному часі.

4.2 Визначити архітектуру програмної системи відслідковування об'єктів.

4.3 Розробити програмне забезпечення для виявлення та відстежування об'єктів.

4.4 Оцінити комерційний потенціал розробки.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих технологій, огляд аналогів системи.	19.09.2024	15.10.2024	Вступ, Розділ 1
2	Визначення архітектури розподіленої системи	16.10.2024	29.10.2024	Розділ 2
3	Програмна реалізація модулів виявлення, відстежування об'єктів та модулю динамічної адаптації	30.10.2024	19.11.2024	Розділ 3
4	Розробка рекомендацій з введення в експлуатацію	20.11.2024	26.11.2024	Розділ 4
5	Підготовка економічної частини	27.11.2024	3.12.2024	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	4.12.2024	10.12.2024	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.02.2024	15.02.2024	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Схема алгоритму

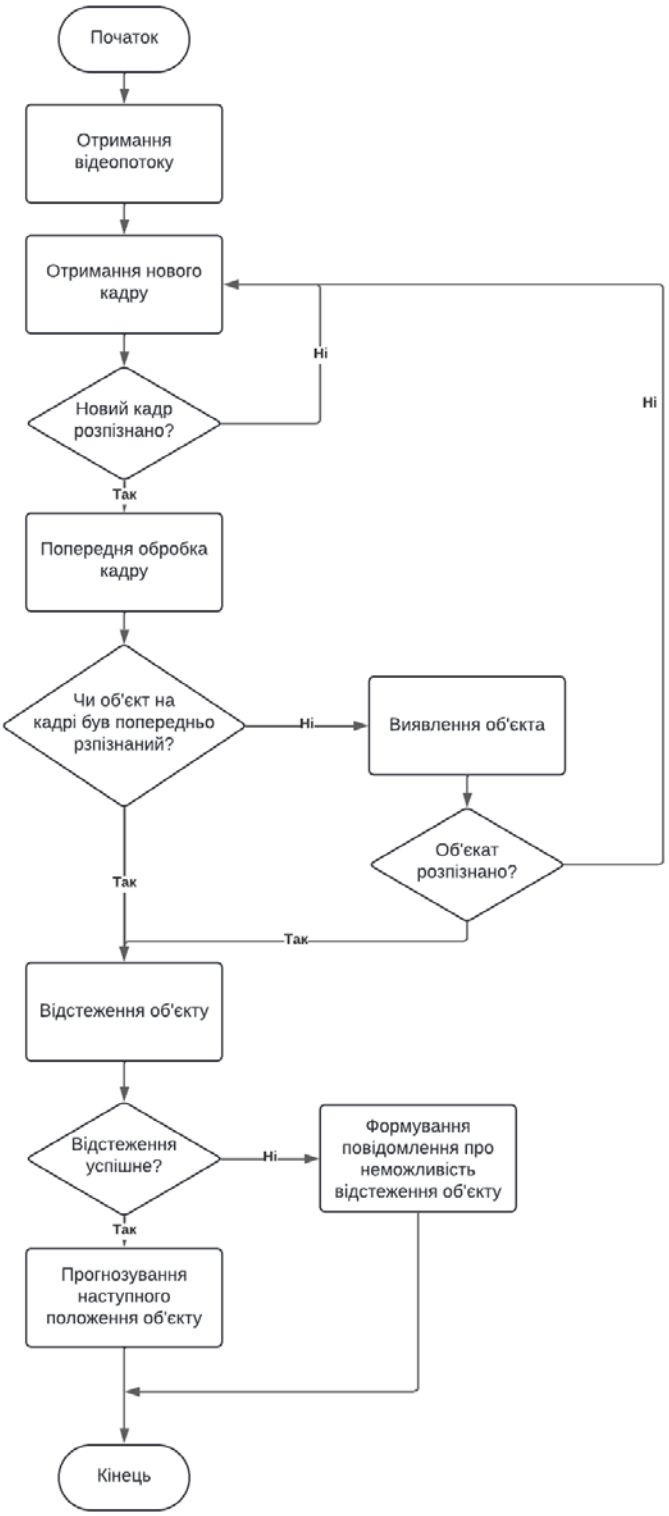

Рисунок Б.1 – Схема алгоритму загального функціонування методів відслідковування об'єктів

ДОДАТОК В

Лістинг коду

```
import cv2

import numpy as np

from pykalman import KalmanFilter


# Завантаження YOLO

net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')

layer_names = net.getLayerNames()

output_layers = [layer_names[i – 1] for i in
net.getUnconnectedOutLayers().flatten()]

classes = []

with open('coco.names', 'r') as f:

    classes = [line.strip() for line in f.readlines()]


# Класи для відстеження (наприклад, людина і автомобіль)

target_classes = {"person", "car"}


# Ініціалізація Kalman Filter

kalman = KalmanFilter(

    transition_matrices=np.array([[1, 0, 1, 0], [0, 1, 0, 1],
[0, 0, 1, 0], [0, 0, 0, 1]]),

    observation_matrices=np.array([[1, 0, 0, 0], [0, 1, 0,
0]])

)

initial_state_mean = np.array([0, 0, 0, 0])

initial_covariance = np.eye(4) * 1e-4

predicted_state_mean, predicted_state_covariance =
initial_state_mean, initial_covariance
```

```
# Зберігання траєкторій об'єктів та попередніх позицій
history = {}
previous_positions = {}

# Відеозахоплення
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Помилка: не вдається відкрити камеру")
    exit()

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

    # Оцінка рівня освітленості кадру
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    avg_brightness = np.mean(gray_frame)

    # Динамічна адаптація детекції
    confidence_threshold = 0.5 if avg_brightness > 50 else 0.3
# Знижуємо поріг за умов низького освітлення

    # Попередня обробка кадру
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416),
(0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
```

```
# Змінні для виявлення об'єктів
class_ids = []
confidences = []
boxes = []

# Перебір виходів YOLO
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > confidence_threshold:
            label = str(classes[class_id])

            # Фільтрація за класом
            if label not in target_classes:
                continue

            # Координати об'єкта
            center_x = int(detection[0] * frame.shape[1])
            center_y = int(detection[1] * frame.shape[0])
            w = int(detection[2] * frame.shape[1])
            h = int(detection[3] * frame.shape[0])

            # Обчислення координат рамки
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
```

```
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    # Накладення рамок на зображення
    indexes = cv2.dnn.NMSBoxes(boxes, confidences,
confidence_threshold, 0.4)
    detected_count = 0
    for i in range(len(boxes)):
        if i in indexes:
            detected_count += 1
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            color = (0, 255, 0)
            cv2.rectangle(frame, (x, y), (x + w, y + h),
color, 2)
            cv2.putText(frame, label, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

            # Використання Kalman Filter для відстежування
об'єкта
            current_position = np.array([x + w / 2, y + h /
2])

            # Оцінка швидкості об'єкта
            if i in previous_positions:
                speed = np.linalg.norm(current_position -
previous_positions[i])
```

```
                kalman.transition_covariance = np.eye(4) *
(1e-3 if speed < 10 else 1e-2)  # Динамічна адаптація


            previous_positions[i] = current_position


            predicted_state_mean, predicted_state_covariance =
kalman.filter_update(
                predicted_state_mean,
predicted_state_covariance, observation=current_position
            )


            # Відображення прогнозу
            predicted_x, predicted_y =
predicted_state_mean[:2]
            cv2.circle(frame, (int(predicted_x),
int(predicted_y)), 5, (255, 0, 0), -1)


            # Додавання до історії для побудови траєкторії
            if i not in history:
                history[i] = []
            history[i].append((x + w // 2, y + h // 2))


            # Відображення траєкторії
            for j in range(1, len(history[i])):
                cv2.line(frame, history[i][j - 1],
history[i][j], (0, 255, 255), 2)


    # Відображення статистики
    cv2.putText(frame, f'Objects Detected: {detected_count}',
(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

```
    cv2.putText(frame, f'Avg Brightness: 
{avg_brightness:.2f}', (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 1, 
(0, 255, 0), 2)


    # Відображення результату
    cv2.imshow('Object Detection and Tracking', frame)


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


cap.release()
cv2.destroyAllWindows()
```

ДОДАТОК Г

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:_____ Метод і засоби відслідковування об’єктів на відео в реальному часі

Тип роботи:_____Магістерська кваліфікаційна робота_____

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____ кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Керівник_____Добровольська Н. В., доц. каф. ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	85%
Загальна схожість	15%

Аналіз звіту подібності (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ _____Кушнір В.А.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М__

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)