Вищий навчальний заклад

«Університет економіки та права «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Розробка системи розпізнавання об’єктів у системі відеоспостережень

Виконав _____

(Підпис)

Совєтніков Олександр Іванович

Науковий керівник

Чернозубкін Ігор Олександрович

_____

_____

(Резолюція «До захисту»)

Попередній захист:

_____

(Висновок: “До захисту в екзаменаційній комісії ”)

Голова циклової комісії _____ _____ _____

(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Голова циклової комісії _____ Леонід УВАРОВ
(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛ коректно ІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Совєтніков Олександр Іванович

1. Тема роботи Розробка системи розпізнавання об’єктів у системі відеоспостережень затверджена наказом по університету від «___» _____ 202__ р. № _____

2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:

а) цільова аудиторія – тип користувачів системи (охоронні компанії, приватні особи, державні установи тощо), об’єкти для розпізнавання (люди, автомобілів, тварини тощо) та важливі функції (автоматичне виявлення, ідентифікація, сповіщення тощо);

б) функціональність - типи об'єктів для розпізнавання (люди-обличчя, автомобілі-номерні знаки, марки, тварини, інші предмети), виявлення руху в кадрі та його аналіз, відстеження переміщення об'єктів в кадрі, ідентифікація розпізнаних об'єктів (наприклад, порівняння обличчя з базою даних; у даному випадку – яка база даних, доступ до бази даних), автоматичне сповіщення про виявлення певних об'єктів або подій (наприклад, вторгнення на територію), зберігання відеозаписів та даних про розпізнані об'єкти, можливість аналізу цих даних;

в) технічні вимоги – тип камер відеоспостереження (IP-камери, аналогові камери), технічне обладнання для обробки відео (сервер, комп'ютер), програмне забезпечення для розпізнавання об'єктів (бібліотеки машинного навчання, спеціалізоване ПЗ), вимоги до точності та швидкості розпізнавання;

г) кращі практики та існуючі розробки в області систем розпізнавання об'єктів для відеоспостереження.

4. Зміст пояснювальної записки

а) Розділ 1 Теоретична частина. Аналіз кращих практик та існуючих розробок в області систем розпізнавання об'єктів для відеоспостереження, обґрунтування функціональності та технічних вимог.

б) Розділ 2 Проектування та розробка. Визначення архітектури системи (централізована, децентралізована, хмарна) та її компонентів, методів взаємодії між собою компонентів системи. Алгоритми розпізнавання (машинне навчання – глибоке навчання, нейроні мережи, моделі машинного навчання (CNN, RNN)). Методи обробки відео (покращення якості зображення, виділення ключових кадрів тощо) та стиснення відео для зберігання. База даних для зберігання інформації про розпізнані об'єкти та події.

в) Розділ 3 Експериментальна частина. Тестування системи (на реальних відеозаписах, на тестових даних) та критерії успіху тестування (точність розпізнавання, швидкість обробки). Оцінка ефективності системи (надійність, продуктивність). Інструкції користувачам (адміністраторам, програмістами – опис архітектури системи, принципів роботи, алгоритмів, специфікації на обладнання та програмне забезпечення, інструкції з встановлення та налаштування системи, користувачам – пояснення щодо використання системи, налаштування параметрів, вирішення проблем).

5. Перелік графічного матеріалу:
Скріншоти існуючих рішень
Скріншоти інтерфейсу продукту
Схеми і таблиці щодо візуалізації аналізу
Блок-схеми алгоритмів
Діаграми щодо проектування продукту (наприклад, потоків даних, переходів станів, сутність-зв'язок, UML, бази даних тощо)

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ Ігор ЧЕРНОЗУБКІН
(підпис)

Завдання прийняв до виконання _____ Олександр СОВЄТНІКОВ
(підпис)

РЕФЕРАТ

Кваліфікаційна робота: 57 стор., 23 рис., 12 табл., 11 джерел.

Об’єкт дослідження – процес обробки відеопотоку з камер спостереження.

Мета роботи – розробка програмної системи розпізнавання об'єктів у системі відеоспостереження з використанням алгоритмів комп’ютерного зору, яка дозволяє виявляти людей у кадрі в реальному часі, фіксувати їх появу методом скріншоту, вести статистику подій та налаштування для кожної камери окремо.

Методи та засоби розробки – Python, модель глибокого навчання YOLO (You Only Look Once), бібліотеки OpenCV, Tkinter.

У кваліфікаційній роботі розглянуто існуючі методи та засоби автоматизованого відеоаналізу, обґрунтовано вибір технологій, описано архітектуру та алгоритми роботи системи, інтерфейс користувача, результати тестування та умови безпечної праці під час розробки.

Сфера застосування – системи відеоспостереження, автоматизовані системи безпеки, аналітика відеопотоку.

Ключові слова: відеоспостереження, розпізнавання об’єктів, YOLO, реальний час, автоматизація.

ABSTRACT

Qualification work: 53 pages, 23 sheets, 12 tables, 11 sources.

Object of research – the process of video stream processing from surveillance cameras.

Objective of the work – development of a software system for object recognition in a video surveillance system using computer vision algorithms, which allows detecting people in the frame in real-time, capturing their appearance by screenshot, tracking event statistics, and configuring settings for each camera separately.

Methods and tools for development – Python, deep learning model YOLO (You Only Look Once), OpenCV libraries, Tkinter.

The qualification paper reviews existing methods and tools for automated video analysis, justifies the choice of technologies, describes the system architecture and algorithms, user interface, testing results, and working safety conditions during development.

Field of application – video surveillance systems, automated security systems, video stream analytics.

Keywords: video surveillance, object recognition, YOLO, real-time, automation.

ЗМІСТ

ВСТУП

Актуальність задачі. Традиційні методи моніторингу, зокрема постійне спостереження за відеокамерами людиною, є малоефективними та мають низку недоліків. Людський фактор — втома, неуважність або засинання — часто призводить до втрати важливої інформації, особливо при обслуговуванні великої кількості камер. Це зумовлює актуальність впровадження інтелектуальних систем, здатних аналізувати відеопотік і розпізнавати об’єкти в реальному часі.

Мета кваліфікаційної роботи: розробка програмної системи розпізнавання об’єктів у системі відеоспостереження, яка дозволяє виявляти людей на відео, та фіксувати їх появу в кадрі методом скріншоту, вести статистику та налаштування для кожної камери окремо.

Об’єкт дослідження: процес обробки відеопотоку з камер спостереження.

Предмет дослідження: алгоритми розпізнавання об’єктів та методи автоматизації відеоаналізу.

Методи дослідження:

У роботі використано методи аналізу літературних джерел і технічної документації, метод системного проектування, методи моделювання програмних компонентів, експериментальне тестування результатів роботи системи.

Завдання роботи:

Для досягнення поставленої мети у роботі необхідно:

1. Провести огляд існуючих систем відеоспостереження та методів розпізнавання об’єктів
2. Обґрунтувати вибір технологій, що будуть використані у розробці
3. Реалізувати графічний інтерфейс користувача для налаштування параметрів системи

4. Розробити ядро системи, яке виконує обробку відео, розпізнавання об’єктів, збереження скріншотів, логування та ведення статистики

5. Провести тестування системи в різних умовах та оцінити її ефективність

Технічна база:

В якості технологічної основи були обрані: мова програмування Python та модель YOLO (You Only Look Once)

Графічний інтерфейс реалізовано як окрему програму, що створює конфігураційний файл, який потім використовується ядром системи. Це рішення дозволяє розділити налаштування та виконання програми, що спрощує користування системою.

Структура роботи:

Кваліфікаційна робота складається з 3-х розділів: аналітичний розділ з аналізом технологій, проектний — з описом архітектури та реалізації, опис алгоритмів роботи системи, опис реалізації програмного забезпечення, тестування та оцінка ефективності.

Практичне значення одержаних результатів:

Результатом роботи є готова до використання система розпізнавання об’єктів, яка не потребує серверного оточення та складних налаштувань, що робить її придатною для домашнього або локального використання. Програмний продукт може бути легко адаптований або масштабований відповідно до потреб користувача, що забезпечує можливість його подальшого вдосконалення та впровадження в реальні системи відеоспостереження.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

1.1 Актуальність автоматизації систем відеоспостереження

В сучасному світі безпека є однією з найголовніших потреб, а системи відеоспостереження — є однією із її складових. Вона дозволяє фіксувати правопорушення та швидко ідентифікувати зловмисника або стати доказовою базою у випадку несчастного випадку або якоїсь технічної несправності. Сьогодні відеоспостереження стало інструментом контролю у банківській, медичній та охоронних сферах життя.

Проте традиційні системи відеоспостереження мають суттєвий недолік — людський фактор, він може спричинятись рядом не контрольованих факторів: недосипання, недоїдання, неуважність в конкретну хвилину або банальне відволікання. Це може провокувати не менший список проблем від найменшого втрата якоїсь інформації, до найгіршого не своєчасної реакції на якусь подію вище перелічене особливо актуальне для систем з великою кількістю камер, хоча і менші системи від цього не застраховані.

З огляду на вище перелічені недоліки традиційних систем відео спостереження, зростає попит на інтелектуальні системи відеоспостереження, що здатні без впливу оператора аналізувати зображення та виявляти важливі події. Такі системи можуть оперативно реагувати на появу людей, транспорту або інших об'єктів, в купі з ПЗ це дає великий простір для модернізації та налаштування реакцій, наприклад вмикати тривогу якщо помічено людину в зачиненому магазині.

Інтеграція алгоритмів комп’ютерного зору у системи відеоспостереження відкриває нові можливості для автоматизації охорони, реагування в режимі реального часу та зменшення витрат на обслуговування приклад такої автоматизації наведено вище. Застосування моделей глибокого навчання, таких як YOLO, дає змогу значно підвищити точність розпізнавання об'єктів навіть на недорогому обладнанні, що актуально для домашніх систем відеоспостереження.

Таким чином, розробка програмної системи, здатної автоматично розпізнавати об'єкти у відеопотоці з камер спостереження, є актуальним і практично значущим завданням у галузі інформаційних технологій та безпеки.

1.2 Класифікація систем відеоаналітики

Системи відеоаналітики — це програмно-апаратні комплекси, які автоматично аналізують відеопотік для виявлення подій, об’єктів, змін у середовищі або нестандартної поведінки. В залежності від складності обробки та функціоналу такі системи можна умовно класифікувати на три основні категорії.

1.2.1.Системи фіксації руху

Алгоритм роботи цих систем досить примітивний в порівнянні з іншими видами систем. Алгоритм фіксації руху працю наступним чином:

- З відеопотоку отримується статичне зображення, назвемо його «кадр 1», він буде еталонним
- Цей кадр зберігається (наприклад в масиві розміри якого дорівнюють розширенню камери)
- Отримується наступний кадр («кадр 2») з відеопотоку
- За допомогою циклу порівнюється «кадр 1» та «кадр 2» якщо відхилення у пікселях, вони ж елементи масиву, більше за вказаний поріг то відбувається фіксація руху і спрацьовує механізм – це може бути відправка повідомлення або щось інше.
- Якщо ж руху не помічено то «кадр 2» стане еталонним і цикл повториться

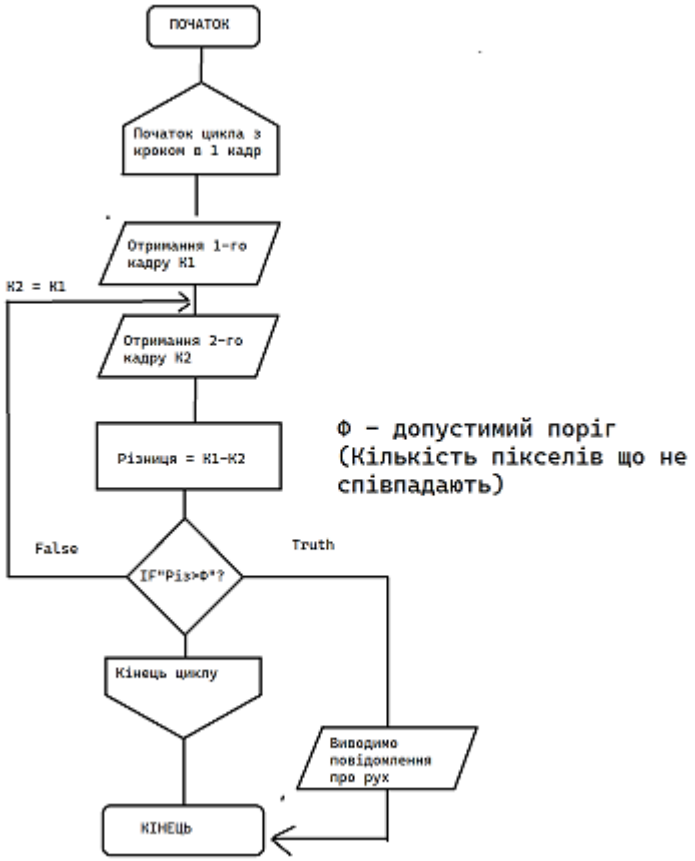

Рисунок 1.1 – Алгоритм фіксації руху

Переваги :

- Простота реалізації
- Швидка реакція
- Низькі вимоги до ресурсів

Недоліки:

- Велика кількість помилок, якщо поріг зависокий то можна повільно пройти, а якщо занизький то буде працювати на вітер, невеликі коливання та інше.

Прикладами таких систем є :

- Monitor(Linux) – це система з відкритим вихідним кодом що дозволяє керувати декількома камерами та налаштовувати реакції на рух
- MotionEye OS — прошивка для Raspberry Pi з функцією запису відео при русі, та сама реакція на рух

- Також багато виробників камер «вшивають» у свої камери подібний функціонал

1.2.2. Системи виявлення об’єктів

Ці системи здатні не лише виявляти рух, але й класифікувати об'єкти — наприклад, людина, автомобіль, тварина. Це суттєво зменшує кількість помилкових спрацювань та дозволяє створювати умовні реакції на певні типи об'єктів.

Принцип дії: використання алгоритмів комп’ютерного зору (як-от YOLO, SSD, Haar Cascades) для ідентифікації об’єктів у кадрі. Більш детально про алгоритм роботи YOLO буде описано нижче.

Переваги:

- Вища точність (тому що аналізується певний сектор пікселів а не весь кадр)
- Гнучке налаштування
- Можливість аналізу в реальному часі

Недоліки:

- Більше використання ресурсів
- Для специфічних задач вимагає навчання моделі

1.2.3.Системи розпізнавання

Найбільш інтелектуальні системи, що виявляють не тільки об’єкти, а ще й можуть ідентифікувати людей за обличчям, номери автомобілів і навіть деякі патерни поведінки.

Принцип дії: поєднання глибоких нейронних мереж, баз даних та механізмів обробки зображень для ідентифікації.

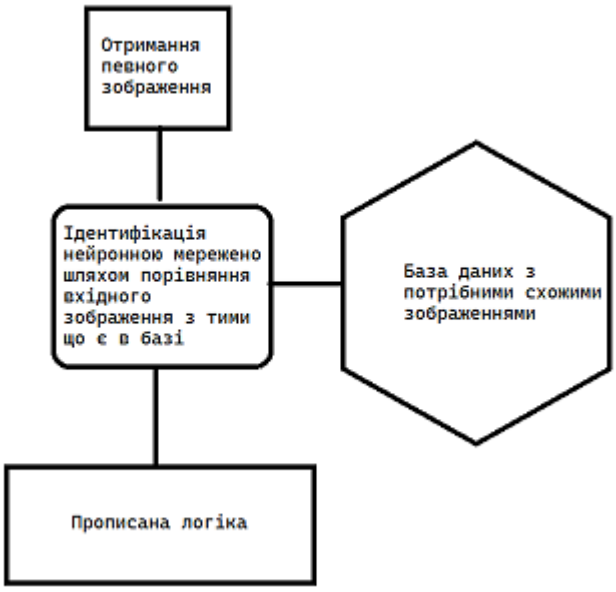

Рисунок 1.2 – Схематична робота систем розпізнавання.

Переваги:

- Високий рівень автоматизації
- Можливість інтеграції з іншими системами
- Запис подій в подробицях (якщо вдалось ідентифікувати людину то її дані та час скоєння злочину або входу в те чи інше місце)

Недоліки:

- Високі вимоги до ресурсів

Прикладами таких систем є:

- Face++ - це API що дає можливість ідентифікувати та перевіряти обличчя порівнюючи його з базою відомих обличь
- Openalpr – це утиліта що розпізнає номерні знаки автомобілів (база її обробки OpenCV)

У висновку можна підсувати наступне, сучасні системи відеоаналітики розвиваються в напрямку підвищення автономності, точності та швидкості роботи. Перехід від простого виявлення руху до повноцінного розпізнавання об’єктів і персон дозволяє значно розширити функціональність систем відеоспостереження та знизити залежність від людського фактору. Це

підтверджує доцільність використання інтелектуальних підходів, таких як YOLO, у практичних проектах.

1.3 Аналіз сучасних програмних рішень

На сьогоднішній день існує багато програмних рішень у сфері відеоспостереження, їх відмінності полягають у функціональності, інтерфейсних рішеннях та вартості ПЗ. У цьому підпункті буде розглянуто декілька популярних рішень а в кінці приведу порівняльну таблицю.

1.3.1. MotionEye [1]

Це веб-інтерфейс для найтування й керування системою відеоспостереження. Вона була розроблена на основі Linux бібліотеки Monitor. Найчастіше таку зв’язку використовують з одно-платними комп’ютерами типу Rassberry PI. Основні налаштування: детекція руху, збереження відео та знімків, зони спостереження.

Переваги:

- Відкритий код, безкоштовне використання
- Простота налаштування
- Економічність в плані ресурсів АЗ

Недоліки:

- Примітивність
- Використання алгоритму фіксації руху

1.3.2. ZoneMinder [2]

Це більш комплексе рішення. В додачу до попереднього присутня підтримка як IP-камер так і аналогових через карту захоплення, є можливість розшити функціонал плагінами, а також має розширені функції моніторингу, виявлення руху та запису.

Переваги:

- Достатня функціональність
- Розширені трігери

- Підтримка плагінів, зокрема продвинуті

Недоліки

- Складність встановлення й налаштування
- Потребує Linux системи з графічною оболонкою

1.3.3. Ivideon [3]

На відміну від інших рішень, що потребують встановлення ПЗ на ЕВМ, Ivideon це хмарна система, що дозволяє зберігати дані як у хмарі так і на локальному сховищі. Підтримує багато функцій найцікавіші з яких це детекція руху, надсилання повідомлень, дистанційний доступ з мобільного додатку.

Переваги:

- Простий інтерфейс користувача
- Перегляд через хмару
- Базові функції безкоштовні

Недоліки:

- Обмежений функціонал у безкоштовній версії
- Залежність від з’єднання з мережею Інтернет

1.3.4. DeepStack [4]

Досить цікаве рішення, його особливість це те, що це AI-сервіс з відкритим кодом, він підтримує локальне розпізнавання об’єктів, осіб, номерних знаків тощо. Також його особливістю є можливість інтеграції з іншими системами.

Преваги:

- Комп’ютерний зір на локальному сервері
- Можливість інтеграції з іншими системами

Недоліки:

- Відсутність інтерфейсу
- Налаштування через API

1.3.5. Milestone XProtect [5]

Це професійна платформа, для створення великих систем відеоспостереження, її особливостями є налаштування для більшої частини

сучасних камер, велика система аналітики, модулі розпізнавання та захисту даних.

Преваги:

- Професійне ПЗ
- Можливість масштабування

Недоліки:

- Вартість ліцензії
- Необхідність потужне обладнання

1.3.6. Порівняльна таблиця

Таблиця порівняння: дивись Додаток 1

1.4 Аналіз технологій комп’ютерного зору

Аналізуючи існуючі програмні рішення можна поміти, що для будь якої аналітики так чи інакше потрібно використовувати комп’ютерний зір. Комп’ютерний зір або Комп'ютерне бачення (англ. Computer Vision) — теорія та технологія створення машин, які можуть проводити виявляння, відстежування та визначення об'єктів. В контексті відео-аналізу, комп’ютерний зір дозволяє автоматизувати процеси виявлення руху, ідентифікації об’єктів та подій.

В цьому підпункті будуть розглянуті основні технології які лежать в основі автоматизації відео-аналізу.

1.4.1. Виявлення руху (Motion Detection)

Це найпримітивніший та найдавніший метод відео-аналізу. Полягає він у виявлені зміни пікселів між кадрами і якщо ця зміна більша за вказаний поріг то фіксується рух. Алгоритм роботи цього методу детально описаний в пункті 1.2, підпункт 1.2.1

Приклади алгоритмів:

- Абсолютна різниця кадрів (*frame differencing*)
- Віднімання фону (*background subtraction*) він же Алгоритм Текномо–Фернандеса

- Алгоритм MOG2 (Посилання на курсову роботу MOG2)

1.4.2. Детекція об'єктів (Object Detection)

Це процес виявлення певних об'єктів у кадрі та побудова обмежувальних рамок навколо них (bounding boxes). Цей підхід дозволяє не лише фіксувати рух, а й розуміти, що саме з'явилося в зоні спостереження.

Приклади алгоритмів:

- YOLO (You Only Look Once)[6]
- SSD (Single Shot Multibox Detector)
- Faster R-CNN

Вище перелічені моделі ґрунтуються на глибоких згорткових нейронних мережах (CNN), які дозволяють досягти високої точності та працювати у реальному часі. CNN (Convolutional Neural Network) — це згорткова нейронна мережа, спеціалізована на обробці зображень. Вона імітує принципи роботи зорової кори людини, автоматично виявляючи характерні ознаки (наприклад, контури, форми, кольори) на різних рівнях глибини. Завдяки особливій структурі, CNN може ефективно розпізнавати об'єкти на зображеннях, не потребуючи ручного виокремлення ознак

1.4.3 Класифікація об’єктів (Object Classification)

Цей тип алгоритмів працює в парі з детекцією об’єктів, і як зрозуміло з назви класифікує об’єкт за певними характеристиками. Класифікація зазвичай відбувається на основі вхідного зображення або виділеної області.

Приклади методів:

- CNN-класифікатори (наприклад, MobileNet, ResNet)
- Вбудовані моделі з бібліотек, такі як TensorFlow Lite[9].

1.4.4. Відстеження об’єктів (Object Tracking)

Це процес відстеження виявленого об’єкта впродовж часу, тобто на наступних кадрах відео. Дає змогу будувати траєкторії, аналізувати швидкість руху, напрямок тощо. Якщо спростити то метод працює наступним чином:

Два попередніх методи виявляють об’єкт на кадрі формуючи рамку навколо об’єкту після чого аналізуються пікселі в цій рамці, і якщо пікселі схожого типу (колір, швидкість тощо) зміщаються то рамка зміщується в ту ж сторону, після чого знову перевіряється чи є в цій рамці цей об’єкт і таким чином відбувається відстеження об’єкту

Приклади методів:

- KCF (Kernelized Correlation Filters)
- CSRT (Discriminative Correlation Filter)
- DeepSORT — поєднання детекції та ідентифікації об’єктів

1.4.5. Розпізнавання (Recognition)

Розпізнавання – найвищий, на момент написання, вид відео-аналізу, виявляє не просто об’єкт чи його клас а індивідуальні особливості кожного об’єкту, наприклад обличчя у людей, або номер автомобіля, його тип, марку, або текст. Знову спрощуючи роботу методу він працює наступним чином, у об’єкта шукаються певні унікальні особливості, наприклад форма носу посадка очей у людей і особливості порівнюються з тими що є у базі даних.

Приклади методів:

- Розпізнавання обличь через FaceNet
- DlibOCR для номерних знаків
- Розпізнавання типу поведінки (на основі моделей Recurrent Neural Networks)

1.5 Аналіз моделей розпізнавання

Сучасне розпізнавання об'єктів у відео здебільшого ґрунтується на глибокому навчанні, зокрема на згорткових нейронних мережах (CNN) – описано вище. Існує кілька потужних архітектур, що показають високу точність та швидкість роботи в реальних умовах. Серед них найбільш поширеними є YOLO, SSD та Faster R-CNN.

1.5.1. YOLO (You Only Look Once)[6]

Одна з найшвидших моделей розпізнавання об’єктів, на момент написання. Її особливість на фоні інших менш популярних моделей в тому, що вона обробляє зображення 1 проходом, тобто детекція та класифікація відбуваються одночасно що пришвидшує роботу всього методу навіть на слабкому АЗ.

Особливості:

- Швидкість роботи
- Вибір версій (кожна з яких дає свій рівень точності та вимоги до АЗ)
- Відкритий код, що дає змогу навчити модель на власних даних

1.5.2. SSD (Single Shot MultiBox Detector)

Модель схожа на YOLO так само обробляє зображення 1 проходом, але працює на іншій архітектурі. Її особливість в тому що вона створена для мобільних пристроїв. Також працює швидше за інші менш популярні моделі, але не має такої гнучкості у виборі версії та навчанні своїми даними

Особливості:

- Середня точність
- Покращена продуктивність на мобільних пристроях
- Менша гнучкість в навчанні

1.5.3. Faster R-CNN

Це двоетапна модель: спочатку виконується виявлення потенційних об'єктів (регіонів), а потім класифікація кожного з них. Вона є однією з найточніших, але поступається у швидкості іншим підходам.

Особливості:

- Висока точність
- Повільність

Таблиця 1.1—Порівняння моделей

Назва моделі	Швидкість	Точність	Простота використання
YOLO	Висока	Висока	Висока
SSD	Висока	Середня	Висока
Faster R-CNN	Низька	Дуже висока	Низька

Після аналізу найпопулярніших моделей розпізнавання об’єктів можна зробити наступні висновки: модель Faster R-CNN, попри високу точність, не підходить для систем відеоаналітики в реальному часі через низьку швидкість обробки. SSD є хорошим аналогом YOLO, однак їй бракує гнучкості, активної спільноти та різноманіття реалізацій.

На відміну від неї, YOLO позбавлена цих обмежень. Вона має відкритий код, активно підтримується розробниками, існує у багатьох версіях (v3, v4, v5, v8) і дозволяє гнучке навчання на власних наборах даних. Завдяки балансу між точністю, швидкістю та простотою інтеграції, модель YOLO є найоптимальнішим вибором для реалізації даної системи.

1.6 Технічні вимоги та технології для розробки

Після аналізу існуючих рішень, методів та моделей формується перелік функціональних та технічних вимог до розроблювального продукту.

1.6.1. Вимоги до продукту

Вимоги до продукту були розділені на функціональні та технічні вимоги

Функціональні вимоги:

1.Розпізнавання людей в реальному часі

Система має обробляти відеопотік в режимі реального часу та визначати наявність людей в кадрі за допомогою технологій комп’ютерного зору.

2.Реакція на появу людини

Система після виявлення людини має автоматично виконувати певну дію.

3.Ведення статистики

Система має вести статистику появ людей у кадрі, статистика повинна містити час виявлення та підрахунок кількості виявлених людей

4.Графічний інтерфейс користувача

Система має мати графічний інтерфейс для налаштування параметрів обробки, збору статистики, перегляду статистики, налаштування реакцій, перегляд лог-файлу

5.Збереження конфігурації

Система має зберігати свої налаштування в окремому файлі щоб після перезапуску не доводилось знову налаштовувати систему

6.Логування

Система має збирати дані про свій стан, роботу, та критичні помилки в роботі

Технічні вимоги:

1.Робота без підключення до мережі Інтернет

Система має повноцінно функціонувати в автономному режимі

2.Можливість роботи на середньостатистичному ПК без дискретної відеокарти

Система має бути оптимізована для роботи на ПК без дискретної відеокарти або бюджетному ноутбуці

3.Інтуїтивність встановлення та налаштування

Встановлення та первинне налаштування повинні бути інтуїтивно зрозумілі настільки щоб пересічний користувач зміг встановити та налаштувати

1.6.2 Вибір технологій

Після формулювання вимог, потрібно визначитись з технологіями які дадуть можливість їх виконати. Після аналізу інструментів був обраний наступний стек технологій:

Таблиця 1.2—Вибір технологій для створення продукту

Компонент	Обрана технологія	Причина вибору
Мова програмування	Python[9]	Легкий синтаксис, багата екосистема, зручність обробки відео
Модель детекції	YOLOv5 / YOLOv8[10]	Висока швидкість, точність, робота у реальному часі, відкритий код
Обробка відео	OpenCV[7]	Потужна бібліотека для захоплення, обробки, збереження відео та зображень
Інтерфейс користувача	Tkinter[8]	Вбудована GUI-бібліотека Python, не потребує сторонніх модулів
Налаштування системи	ConfigParser / JSON	Простота читання/запису конфігурації, доступність для людини
Архітектура	Модульність	Зручність для обслуговування та масштабування

Отже представлений стек технологій є найбільш оптимальним з урахуванням вимог, що висуваються до продукту. Використання Python, YOLO та супутніх бібліотек дозволяє реалізувати ефективну, автономну та доступну систему відео-аналітики для широкого кола користувачів.

1.7 Висновки за розділом 1

Після розгляду та аналізу сучасних системи відеоспостереження, методів комп’ютерного зору та моделей розпізнавання об’єктів. Були визначені основні недоліки існуючих рішень, вони стали основою для формування вимог до власної системи. Аналіз та порівняння показало доцільність використання моделі YOLOv5 як компромісного рішення між швидкістю та точністю. А також було обґрунтовано вибір інших технологій, що лежать в основі проекту.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ

В цьому розділі будуть детально описані архітектура та реалізація продукту.

2.1 Загальна архітектура

Для розробки продукту була вибрана архітектура яка ділиться на 2 модулі, модуль графічного інтерфейсу в цьому буде налаштовуватись інтерфейс користувача та конфігурація продукту, а також ядро системи відео-аналітики, його задачі обробка потокового відео та його обробка. Детальніше про ці модулі:

Графічний інтерфейс налаштування (GUI) – програма, що дозволяє користувачу задавати параметри для кожної камери спостереження, вибирати модель розпізнавання, частоту оновлення, шляхи збереження скріншотів тощо. Після налаштування створюється конфігураційний файл.

Ядро системи відеоаналітики (детектор) – програма, яка виконує розпізнавання об'єктів у відеопотоці згідно з заданими параметрами, використовує модель YOLO, обробляє відео з кількох камер, фіксує появу людей та зберігає скріншоти й статистику.

Таким чином запропонована архітектура легко може масштабуватись, обслужуватись або змінюватись. Наприклад замінити модель розпізнавання при цьому не чіпаючи інтерфейс або навпаки змінити інтерфейс не чіпаючи модель і її логіку.

2.2 Взаємодія компонентів

Після загального опису архітектури ось так буде виглядати схема взаємодії компонентів:

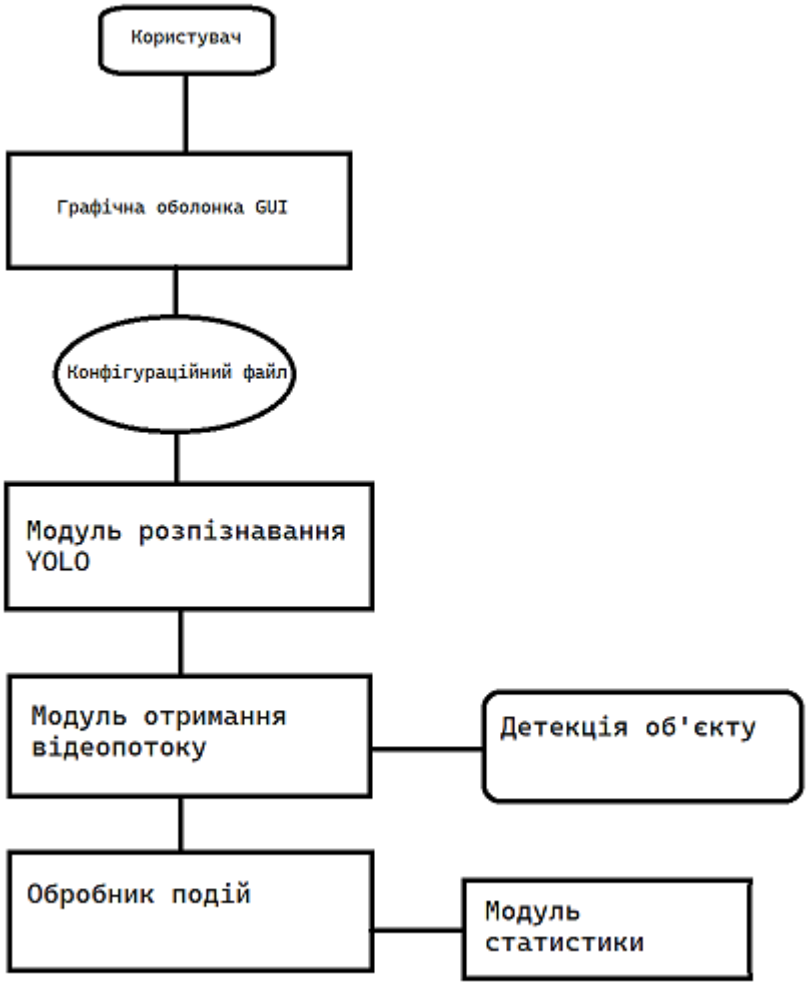

Рисунок 2.1 – Схема взаємодії компонентів

Нижче наведена пояснювальна таблиця:

Таблиця 2.1 —Таблиця функцій компонентів

Модуль	**Опис**
GUI	Надає можливість змінити налаштування без особливих навичок
Модуль розпізнавання	Обробляє відеопотік виявляючи об’єкти
Модуль отримання відеопотоку	Отримує відеопотік для обробки
Обробник подій	Реагує на появу людини, робить скріншоти та фіксує час
Модуль статистики	Веде статистику

Як це працює, користувач маючи графічну оболонку може або змінити налаштування або використати попередні, після чого запускаються модулі

отримання відеопотоку і модуль розпізнавання, після того як модуль розпізнавання, виявив людини обробник подій зберігає цей кадр фіксуючи час, після чого модуль статистики виконує свою роботу.

2.3 Технічні характеристики середовища розробки

Для програмної реалізації системи було обрано досить слабке АЗ, для тестування на слабкому обладнані, а також платформи, що забезпечують зручність розробки, широкий функціонал та кросс платформеність.

2.3.1. Основне ПЗ для розробки

Таблиця 2.2 «Основні програмні інструменти розробки»

Компонент	**Технологія**
Мова програмування	Python
Середовище розробки	Visual Studio Code
Основні бібліотеки	opencv-python, ultralytics, tkinter, json, os, datetime
Робота з моделлю	ultralytics
Робота з GUI	tkinter
Система керування пакетами	pip
Запуск системи	Запуск через консоль / десктопний ярлик

2.3.2. Обладненя на якому здійснювалась розробка

Таблиця 2.3 «Апаратне забезпечення для розробки та тестування»

Компонент	**Характеристика**
Центральний процесор	Intel Core I5-8400U (1.6 Ghz, 4core)
Оперативна пам’ять	8Gb DDR4
Графічний процесор	Intel UHD Graphics 620 (вбудований)
Накопичувач	SSD 256 GB

Система була протестована без використання дискретного графічного процесора (GPU), що доводить її придатність для запуску на середньостатистичному побутовому ПК або ноутбуці.

2.4 Опис графічного інтерфейсу користувача

Як було визначено у вимогах до програмної реалізації, для полегшення користування та налаштування для рядового користувача була розроблена графічна оболонка. Розробка була здійснена за допомогою Tkinter, що входить до складу Python, вона дозволяє створювати віконний інтерфейс, що сумісний з різними платформами.

2.4.1 Опис функцій GUI

Основними функціями GUI є:

- Вибір кількості камер
- Вказання параметрів розпізнавання
- Вибір директорії для зберігання скріншотів та відео
- Налаштування обробки (опрацювання кожного n-го кадру для налаштування продуктивності)
- Перегляду статистики
- Запуск ядра системи (модуль детекції)

2.4.2 Опис інтерфейсу користувача

При запуску програмного продукту у користувача відкривається вікно з кнопками налаштування, а саме:

- Кнопка вибору кількості камер
- Кнопка налаштування обробки
- Кнопка налаштування реакцій на появу людини
- Кнопка налаштування каталогу скріншотів
- Кнопка перегляду та налаштування статистичних даних
- Кнопка перегляду логів
- Кнопка запуску ядра

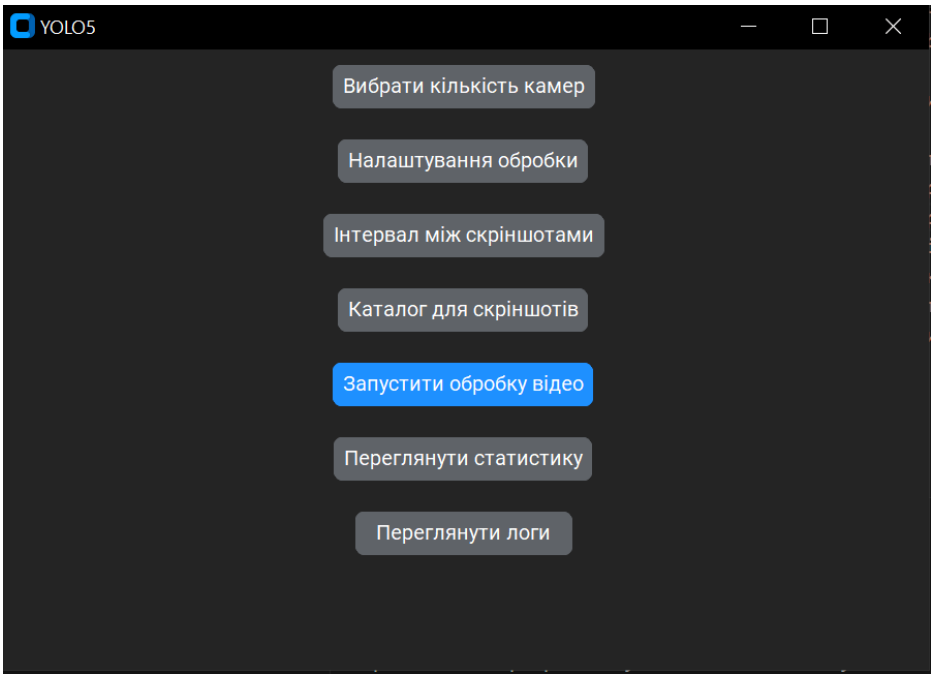

Рисунок 2.2 – Головне меню

Детальніше про роботу кожної кнопки:

Кнопка вибору кількості камер – вікно де користувач вводить кількість підключених камер

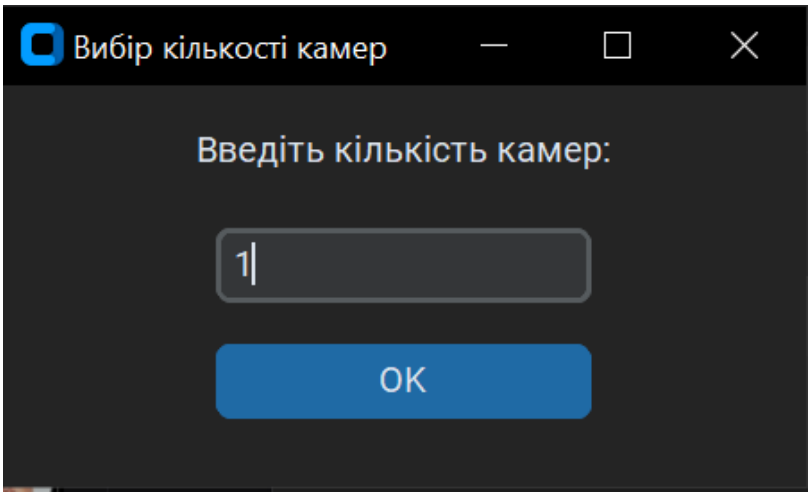

Рисунок 2.3 – Кнопка вибору кількості камер

Кнопка налаштування обробки – віно де користувач може ввести який кадр буде оброблятись (корисно для пришвидшення роботи на дуже слабкому АЗ), чекбокс який дозволяє обробляти кожен кадр що передає камера та налаштування порогу розпізнавання.

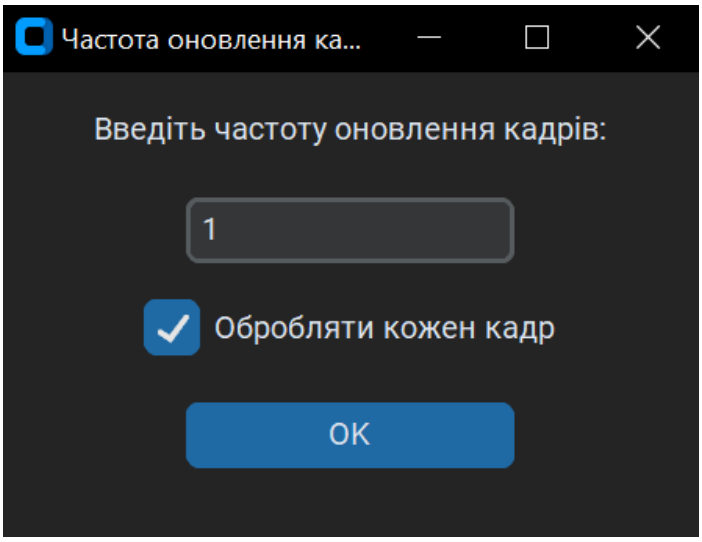

Рисунок 2.4 – Кнопка налаштування обробки

Кнопка налаштування реакцій – дозволяє задати інтервал між скріншотами після виявлення об'єкта.

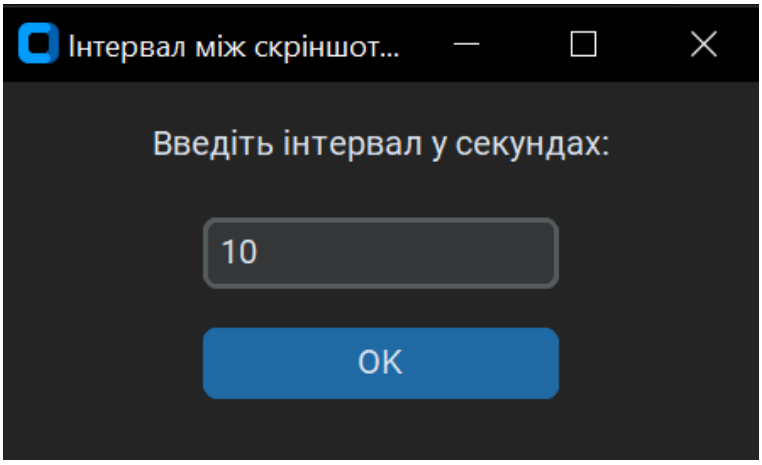

Рисунок 2.5 – Кнопка налаштування реакцій

Кнопки налаштування каталогу – Вибір директорії в яку будуть зберігатись скріншоти

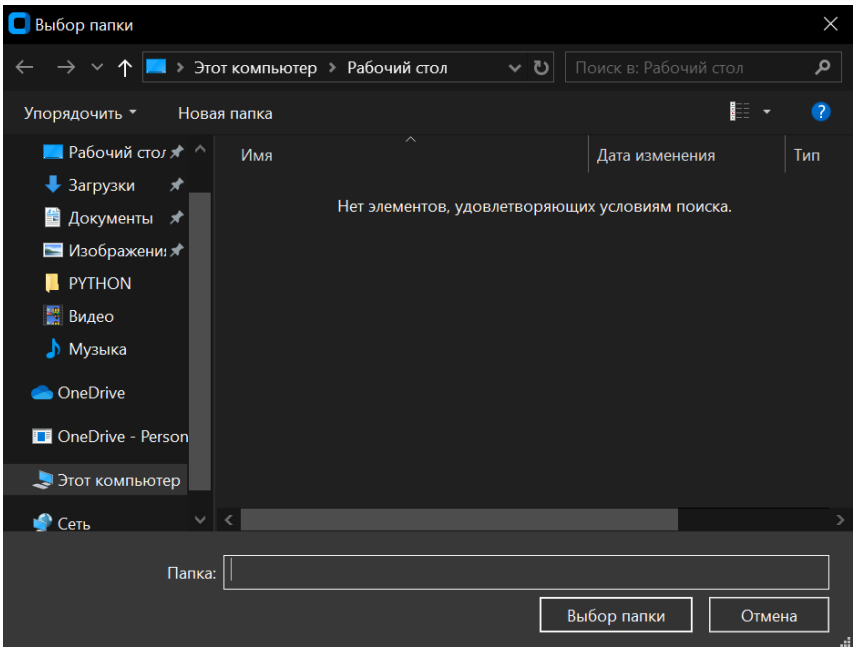

Рисунок 2.6 – Кнопка налаштування каталогу збереження скірншотів

Кнопка перегляду статистики – статистика кількості виявлених об’єктів за день, і за весь час.

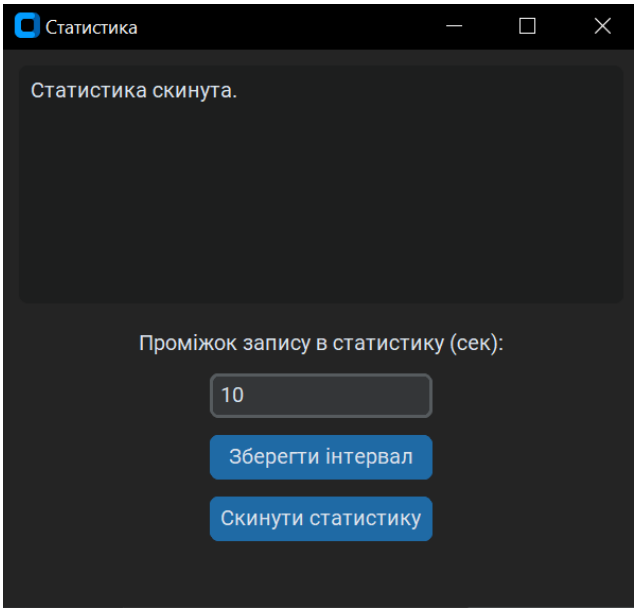

Рисунок 2.7 – Кнопка перегляду статистики

Кнопка перегляду логів - відкриває журнал логів

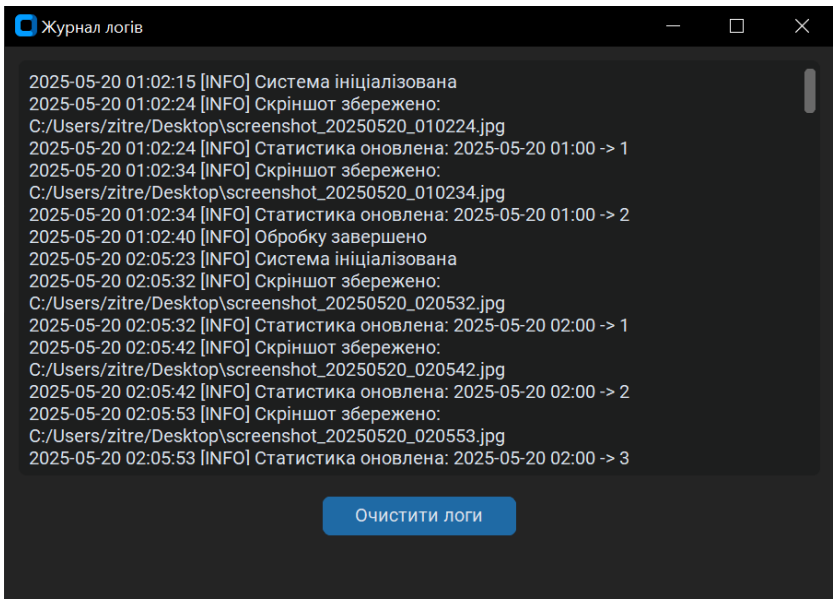

Рисунок 2.8 – Кнопка перегляду логів

Кнопка запуску ядра - виконує запуск основної програми розпізнавання відповідно до збережених налаштувань.

2.5 Робота ядра системи

Основний компонент системи це ядро, саме воно відповідальне за обробку відеопотоку, виявлення об’єктів, на виконання реакцій. Робота ядра реалізована мовою Python та його бібліотек та різноманітних модулів.

2.5.1. Архітектура ядра

Ядро складається з модулів, а саме:

- Модуль зчитування конфігураційного файлу – завантажує налаштування з конфігураційного файлу
- Модуль підключення камер – ініціалізує відео потоки відповідно до обраної кількості камер
- Модуль обробки кадрів – отримує кадри з відеопотоку, пропускаючи n-і кадри для підвищення продуктивності (якщо це вказано в конфігураційному файлі)
- Модуль розпізнавання об’єктів – використовує модель YOLO для аналізу отриманих кадрів

- Модуль реакцій – при виявлені об'єктів запускає відповідну реакцію, наприклад скріншот
- Модуль ведення логів - фіксує події, помилки та виявлення у файл журналу
- Модуль завершення роботи – корректно завершає роботу

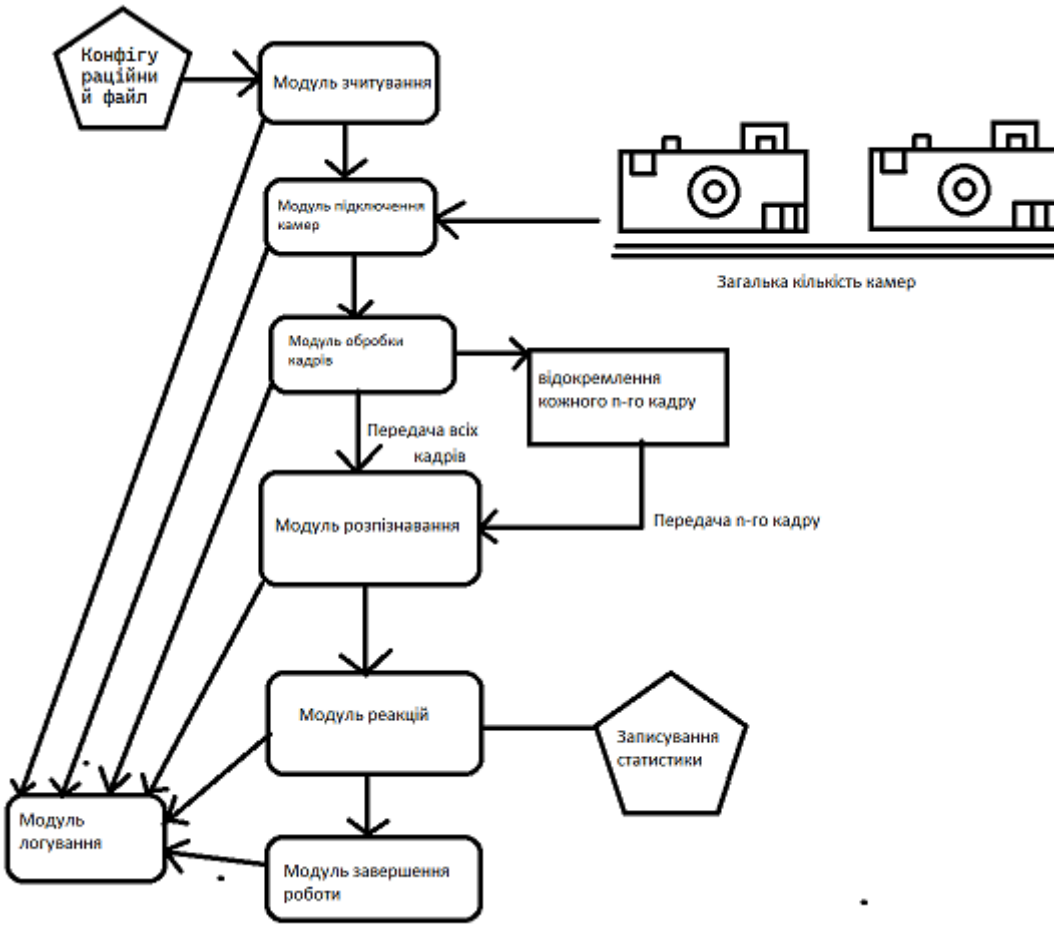

Рисунок 2.9 – Схема роботи ядра системи

2.5.2 Опис циклу роботи ядра

1. Після запуску ядра відбувається зчитування конфігураційного файлу
2. Для кожної камери створюється свій потік
3. Кожен потік:
    1. Зчитує кадр з потоку

2. Якщо кадр підпадає під умову конфігураційного файлу (кожен n-й кадр або кожен), він передається до моделі YOLO
3. YOLO розпізнає об’єкти на кадрі
4. Якщо виявлено заданий об’єкт (наприклад людина):
    1. Створюється скріншот
    2. Збільшується лічильник статистики
    3. Фіксація в логах
4. Система продовжує роботу до тих пір поки користувач не завершить роботу

2.6 Робота з моделлю YOLO

Для розробки програмного продукту, а саме задачі розпізнавання об’єктів (людей) у відеопотоці була використана модель YOLO, вона забезпечує високу швидкість обробки та достатню точність.

2.6.1. Про використану версію

В рамках розробки була використана YOLOv5, як оптимальне рішення для систем з обмеженими ресурсами. YOLOv5 розроблена командою Ultralytics і реалізована на базі бібліотеки PyTorch, що значно спрощує інтеграцію у Python-середовище.

2.6.2 Інтеграція YOLOv5 у Python

Модель не є частиною Python тому її треба клонувати, з Git-Hub, у своє середовище наступним чином:

```
git clone https://github.com/ultralytics/yolov5
cd yolov5
pip install -r requirements.txt
```

Рисунок 2.10 Клонування моделі у своє середовище

2.6.3 Завантаження моделі

YOLOv5s завантажується з офіційного репозиторію автоматично під час першого запуску наступним чином:

```python
from yolov5 import YOLOv5
import torch

# Приклад завантаження через torch.hub
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
```

Рисунок 2.11 – Завантаження моделі

2.6.4. Приклад використання моделі

Наведений приклад є демонстрацією базового функціоналу моделі та ядра, а саме отримати відеопотік та розпізнати об’єкти і відокремлення для обробки n-го кадру.

YOLO5.py > ...

```python
import cv2
from ultralytics import YOLO
# Завантаження моделі
model = YOLO('yolov5s.pt')
# Зчитування відео
cap = cv2.VideoCapture(0)
frame_count = 0  # Лічильник кадрів
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame_count += 1
    # Обробляємо кожен 10-й кадр
    if frame_count % 10 == 0:
        # Обробка кадру
        results = model(frame)  # Розпізнавання

        # Якщо є розпізнані об'єкти, відображаємо їх на кадрі
        if results:
            frame_with_boxes = results[0].plot()

    # два вікна:
    cv2.imshow('Original Frame', frame)
    if frame_count % 10 == 0:
        cv2.imshow('Frame with Objects', frame_with_boxes)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Рисунок 2.12 – Базовий функціонал моделі та ядра

2.6.5. Особливості роботи з моделлю

YOLOv5 видає результат у вигляді об'єкта з полями:

- results.xyxy: координати рамок [x1, y1, x2, y2, confidence, class]
- results.names: відповідність номерів класів до назв (наприклад, "person", "car" тощо)

Це дозволяє легко реалізувати додаткову логіку: запис скріншотів, підрахунок статистики, реакцію на появу людини тощо.

2.7 Обробка відеопотоку

Для розробленої системи одним з найважливіших компонентів є отримання та обробка відеопотоку перед його передачею до моделі розпізнавання.

2.7.1.Підключення до камер

Відеопотік може надходити як від локальної веб-камери, так і з мережевої IP-камери. Для цього використовується бібліотека OpenCV, яка дозволяє зручно працювати з будь-яким типом відеопотоку. Типова конструкція для отримання відеопотоку:

```
import cv2

# Підключення до локальної камери
cap = cv2.VideoCapture(0)

# Або до IP-камери
# cap = cv2.VideoCapture("rtsp://192.168.1.100:554/stream")
```

Рисунок 2.13 – Типова конструкція для отримання відеопотоку

2.7.2. Обробка відеопотоку

Відеопотік – це швидка передача кадрів (24, 30, 60 кадрів в секунду), для оптимізації на слабкому АЗ, можна дати системі час на обробку, тобто аналізувати не всі кадри а певну кількість, це дозволяє кратно зменшити навантаження на систему.

Такий механізм реалізується наступним чином:

```python
frame_counter = 0
process_every_n_frame = 3  # налаштовується в GUI

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

    if frame_counter % process_every_n_frame == 0:
        # Передача кадру до YOLO
        results = model(frame)
        annotated = results.render()[0]
        cv2.imshow("Processed Frame", annotated)
    else:
        cv2.imshow("Skipped Frame", frame)

    frame_counter += 1

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

Рисунок 2.14 – Механізм обробки кожного n-го

2.7.3 Механізм реакції на об’єкт

В рамках цього проекту основним об’єктом буде клас (людина “person”). При виявленні людини система:

- Зберігає скріншот у вибрану директорію
- Записує інформацію до лог-файлу
- Оновлює статистику в GUI

```python
for detection in results.xyxy[0]:
    class_id = int(detection[5])
    class_name = model.names[class_id]

    if class_name == "person":
        # Збереження скріншоту
        timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        filename = f"screenshots/person_{timestamp}.jpg"
        cv2.imwrite(filename, frame)
        break
```

Рисунок 2.15 – Механізм обробки появи людини

2.8 Збереження результатів розпізнавання

Однією з функцій системи є збереження результатів розпізнавального модуля для подальшого перегляду аналізу та ведення статистики. Це забезпечує зручність для користувача в разі якихось подій.

2.8.1. Збереження скріншотів

Після того як система виявляє об’єкт (в моєму випадку це людина), система робить скріншот він же кадр у форматі .jpg. Зберігання відбувається у заданій користувачем директорії. Ім’я файлу формується як «назва камери, дата, година» з розширенням .jpg. Приклад :

«camera1_2025-05-13_14-23-41.jpg»

А програмна реалізація цього механізму виглядає наступним чином:

```python
from datetime import datetime
import cv2
import os

def save_frame(frame, camera_name, output_dir):
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    filename = f"{camera_name}_{timestamp}.jpg"
    path = os.path.join(output_dir, filename)
    cv2.imwrite(path, frame)
```

Рисунок 2.16 – Механізм збереження кадру з датою

2.8.2. Логи та статистика

Кожен факт виявлення об’єкта записується в лог-файл (events.log). Лог містить текстовий запис із зазначенням дати, часу, камери та типу виявленого об’єкта.

Приклад формату в лог-файлі :

[2025-05-13 14:23:41] Camera: camera1 | Object: person

Механізм запису в лог-файл:

```python
def log_event(camera_name, object_type, log_file="events.log"):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(log_file, "a") as f:
        f.write(f"[{timestamp}] Camera: {camera_name} | Object: {object_type}\n")
```

Рисунок 2.17 – Механізм запису в лог-файл

2.8.3 Статистичні дані

Крім збереження в лог-файлі всіх виявлень людини, також реалізовано збір та збереження статистики виявлень у форматі .json, де для кожної камери фіксується кількість виявлених об’єктів за день та за весь час.

Приклад змісту файлу статистики:

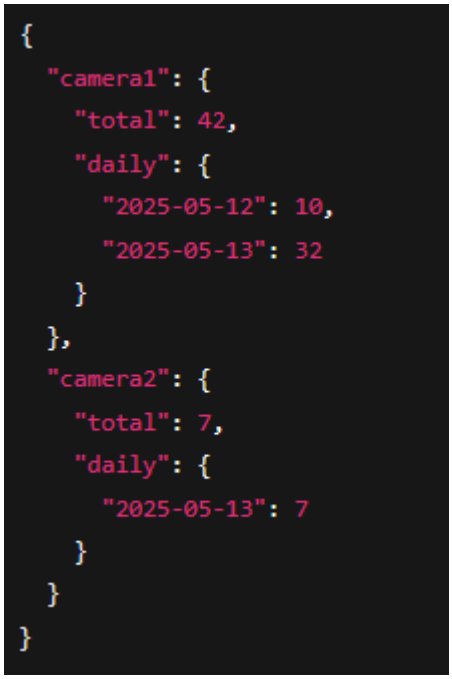

Рисунок 2.18 – Приклад статичного файлу

Механізм збору та запису в статистичний файл:

```python
import json

def update_stats(camera_name, stats_file="stats.json"):
    today = datetime.now().strftime("%Y-%m-%d")

    if not os.path.exists(stats_file):
        stats = {}
    else:
        with open(stats_file, "r") as f:
            stats = json.load(f)

    if camera_name not in stats:
        stats[camera_name] = {"total": 0, "daily": {}}
    stats[camera_name]["total"] += 1
    stats[camera_name]["daily"][today] = stats[camera_name]["daily"].get(today, 0) + 1

    with open(stats_file, "w") as f:
        json.dump(stats, f, indent=2)
```

Рисунок 2.19 – Механізм збору та запису в статистичний файл

2.9 Реакції на виявлення об’єктів

Для підвищення інформативності системи та забезпечення мінімального реагування навіть без постійного нагляду за відео, реалізовано прості реакції на виявлення об’єктів. Основна ідея – створення скріншотів моменту появи людини (або іншого об’єкта), що дозволяє вести журнал подій.

2.9.1. Основний функціонал

Основна, реалізована, функція – збереження скріншотів/кадрів, на якому виявлено заданий об’єкт. При розробці архітектури були визначені наступні характеристики зображення:

- Рамку відмітку об’єкта
- Клас об’єкта
- Впевненість (рейтинг від 0.5 до 1 наскільки модель вважає цей об’єкт правдивим, наприклад людина 0,75 це означає що найймовірніше це людина)
- Назву камери

- Час та дату виявлення

Скріншоти зберігаються в папці вказаній в GUI.

2.9.2 Обмеження частоти знімків

Для зменшення кількості схожих чи однакових знімків, а також зменшення навантаження на диск, був розроблений механізм мінімального інтервалу між знімками. Наприклад, якщо налаштовано інтервал у 10 секунд, то новий скріншот буде створено не раніше, ніж через 10 секунд після попереднього.

Реалізація механізму виглядає наступним чином:

```python
import time
from datetime import datetime
import cv2
import os

last_screenshot_time = 0
screenshot_interval = 10  # секунд

def save_screenshot(frame, camera_id, obj_label):
    global last_screenshot_time
    current_time = time.time()

    if current_time - last_screenshot_time < screenshot_interval:
        return  # ще не минув інтервал

    last_screenshot_time = current_time
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    filename = f"screenshots/cam_{camera_id}_{obj_label}_{timestamp}.jpg"

    if not os.path.exists("screenshots"):
        os.makedirs("screenshots")

    cv2.imwrite(filename, frame)
```

Рисунок 2.20 – Реалізація механізму мінімального інтервалу між кадрами

2.10 Логування

Основною метою логування є надання інформації, що дозволить проаналізувати помилки чи проблеми у роботі програми.

В межах проекту було реалізовано логування в форматах JSON, як інтерфейс для майбутнього масштабування, та .txt форматі для читання людиною.

2.10.1. Логування в текстовому форматі

Простий текстовий лог зберігає кожну подію у вигляді одного рядка

А ось так реалізується записування в текстовий файл лога:

```
import logging

logging.basicConfig(
    filename="log.txt",
    level=logging.INFO,
    format='[%(asctime)s] [%(levelname)s] %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

logging.info("Camera: camera1 | Object detected: person")
```

Рисунок 2.21 – Записування логу в текстовому файлі

2.10.2. Логування у форматі JSON

Вибір цього формату полягає в тому, що він дуже зручний для подальшої обробки або відправки.

Запит в JSON файл реалізується так:

```python
import json
from datetime import datetime

def write_json_log(event_type, event, camera=None, obj=None, filename="log.json"):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    log_entry = {
        "timestamp": timestamp,
        "type": event_type,
        "event": event
    }
    if camera:
        log_entry["camera"] = camera
    if obj:
        log_entry["object"] = obj

    try:
        with open(filename, "r") as f:
            logs = json.load(f)
    except (FileNotFoundError, json.decoder.JSONDecodeError):
        logs = []

    logs.append(log_entry)

    with open(filename, "w") as f:
        json.dump(logs, f, indent=2)
```

Рисунок 2.22 – Записування логу в JSON файл

2.10.3 Перегляд логів

Перегляд логів реалізовано теж 2 способами, переглядом текстового файлу, або в самому GUI в спеціально відведеній для цього кнопці.

2.11 Висновки за розділом 2

В цьому розділі було детально описано реалізацію всієї системи: архітектуру, взаємодію між модулями, логіку роботи ядра, графічний інтерфейс та супутні механізми. Також було описано структуру даних, формат логування та методику збору та зберігання статистики. Реалізація виконана, з урахуванням можливого масштабування, та адаптації під різні задачі. Всі функціональні вимоги були реалізовані згідно з поставленими вимогами.

РОЗДІЛ 3 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТИ

3.1 Мета тестування

Метою тестування є перевірка прац еспроможності всієї системи, перевірка на наявність критичних помилок, виявлення не значних помилок, перевірка функціональних можливостей що зазначені у вимогах, перевірка стабільності роботи під навантаженням, а також оцінка коректності, точності та швидкості виявлення об’єктів. Окрему увагу буде приділено перевірці та оцінці роботи на слабкому обладнанні та експерименти з налаштуваннями оптимізації.

3.2 Методика тестування

Тестування було умовно поділене на:

- Функціональне – перевірка кожної функції GUI, обробки відеопотоку, збереження скріншотів, збереження логів.
- Сценарне тестування – тестування в умовах наближених до реальних (перший запуск, перше налаштування перший запуск ядра, повторний запуск)
- Навантажувальне тестування – запуск в найгірших умовах для системи. (зменшена кількість ОЗУ, погане освітлення, швидка зміна об’єктів у відео-потоці).
- Оцінка точності – оцінка точності розпізнавання при різному освітленні та відстані до об’єкту.

3.3 Функціональне тестування

Як було визначено основними функціями системи є: запуск GUI, налаштування кількості камер, налаштування обробки, налаштування збереження скріншотів, збереження логів, збереження статистики, запуск ядра системи. Тож нижче наведені результати тестування.

Таблиця 3.1 —Перевірка основних функцій в GUI

№	Функція	Очікуваний результат	Фактичний результат	Статус
1	Запуск GUI	Програма запускається без помилок	Успіх	Пройдено
2	Налаштування кількості камер	Інформація зберігається у конфігураційний файл	Успіх при відсутності конфігураційного файлу він створюється	Пройдено
3	Налаштування обробки	Інформація зберігається у конфігураційний файл	Успіх при відсутності конфігураційного файлу він створюється	Пройдено
4	Налаштування збереження скріншотів	Інформація зберігається у конфігураційний файл	Успіх при відсутності конфігураційного файлу він створюється	Пройдено
5	Збереження логів	Інформація зберігається у конфігураційному файлі	Успіх при відсутності конфігураційного файлу він створюється	Пройдено
6	Налаштування статистики	Інформація зберігається у конфігураційному файлі	Успіх при відсутності конфігураційного файлу він створюється	Пройдено
7	Запуск ядра	Файл ядра запускається без помилок	Успіх	Пройдено

Таблиця 3.2—Тестування роботи системи під час виконання

№	Перевірка	Очікуваний результат	Фактичний результат	Статус
1	Виявлення об’єкту	Система виявляє об’єкти та відбувається робота пов’язаної логіки	Розпізнавання працює коректно	Пройдено
2	Записування у лог-файл	Записується інформація у текстовому та JSON файлі	Інформація записується в файли коректно	Пройдено
3	Збереження статистики	Після кожного інтервалу інформація зберігається у відповідному файлі	Статистика записується коректно	Пройдено
4	Перевірка інтервалу скріншотів	Скріншот не створюється раніше заданого інтервалу	Інтервали дотримані	Пройдено
5	Робота при зниженому освітлені та його відсутності	При зменшеному освітленні, точність розпізнавання зменшується а при відсутності світла вона не відбувається	При зниженому освітлені: точність знизилась При відсутності не відбулась	Пройдено
6	Втрата відео-потоку	Робота не припиняється та з’являється відповідний запис в лог-файлі	У лог-файлі зафіксовано втрату з'єднання	Пройдено
7	Робота системи понад 1 годину	Система не дає збоїв, продовжує роботу	Система працює стабільно	Пройдено
8	Реакція на появу людини	Система створює скріншот	Скріншот збережено	Пройдено

Функціональне тестування показало, що система працює правильно. Усі передбачені функції працюють справно та стабільно і відповідають всім заданим вимогам.

3.4 Сценарне тестування

Суть сценарного тестування в тому щоб перевірити роботи системи «від початку і до кінця», тобто пройти повний цикл роботи, так як це буде робити користувач. Такий спосіб тестування дозволить перевірити повний цикл роботи, та виявити помилки на «стиках» модулів.

Для цього тестування були визначені наступні параметри:

Таблиця 3.3—Параметри запуску системи

Кількість камер	**1 камера**
Обробка кадрів	Кожен кадр
Каталог для збереження скріншотів	Робочий стіл
Проміжок запису в статистику	10 секунд
Проміжок між скріншотами	30 секунд

Рисунок 3.1 – Схема сценарного тестування

Таблиця 3.4—Сценарне тестування

Крок	**Дія**	**Очікуваний результат**	**Фактичний результат**	**Статус**
1	Запуск GUI	Графічний інтерфейс запускається без помилок	GUI відкривається без помилок	Пройдено
2	Заповнення параметрів	Введені дані відображаються в полях	Дані відображаються в полях	Пройдено
3	Запуск ядра	Після натискання відповідної кнопки запускається обробка відео-потоку	Обробка запускається	Пройдено
4	Перевірка оброки згідно до параметрів	Обробка відбувається згідно до параметрів	Обробка відбувається згідно до параметрів	Пройдено
5	Завершення роботи тестувальником	Програма коректно закривається без помилок	Програма закрилась без помилок	Пройдено

6	Повторний запуск GUI	Графічний інтерфейс запускається без помилок	GUI відкривається без помилок	Пройдено
7	Повторний запуск ядра	Після натискання відповідної кнопки запускається обробка відео-потоку	Обробка запускається	Пройдено
8	Повторна перевірка обробки згідно до параметрів	Обробка відбувається згідно до параметрів	Обробка відбувається згідно до параметрів	Пройдено

Сценарне тестування показало, що система проходить повний цикл роботи, а також те що всі її компоненти коректно працюють між собою, дані передаються між модулями без втрат, що свідчить про те що система готова до реального використання.

3.5 Навантажувальне тестування

Суть навантажувального тестування в тому щоб перевірити роботу системи в найнесприятливіших умовах. Для цієї перевірки були обрані наступні фактори: низька освітленість, зменшення ОЗУ, втрата відеопотоку з його подальшим відновленням, підключення сторонніх програм(що впливає на завантаженість ОЗУ, ЦП, відео-ядра системи).

Для достовірності та порівняльного аналізу буде проведений «еталонний» замір споживання ресурсів після чого буде проведене тестування, з одним несприятливим фактором для розуміння його впливу на систему, а також примітка з варіантами вирішення або покращення, після чого буде проведене тестування з усіма несприятливими факторами, важливо перед початком тестування сказати що кожен з тестів буде проводитись близько години для збору якомога більш точних даних.

Умови для перевірки:

Таблиця 3.5—Умови тестування

Параметри запуску	**Що були і в сценарному тестуванні**
Апаратне забезпечення	Описане у Таблиці 2.3
Зовнішнє оточення	Світла спальна кімната, 1 людина що час від часу рухається по кімнаті
Час тестування	1 година

Еталонне тестування без навантаження:

Таблиця 3.6—Еталонне тестування

Показник	**Значення**	**Примітка**
Завантаженість ЦП	68%	еталон
Завантаженість ОЗУ	536 МБ	еталон
Завантаження відео ядра	11%	еталон
Середній FPS	29	еталон
Кількість зафіксованих людей	360	еталон
Помилки в логах	0	еталон

Тестування зі зниженим освітленням:

Таблиця 3.6—Тестування зі зниженим освітленням

Показник	**Значення**	**Примітка**
Завантаженість ЦП	68%	еталон
Завантаженість ОЗУ	536 МБ	еталон
Завантаження відео ядра	11%	еталон
Середній FPS	29	еталон
Кількість зафіксованих людей	341	Через знижене освітлення та

		розширення камери в певні моменти людина не розпізнається
Помилки в логах	0	еталон

Для покращення розпізнавання пропонується: змінити модель на більш досконалу, змінити камеру з більшою роздільною здатністю або на камеру нічного бачення.

Тестування зі зменшеною кількістю ОЗУ:

Таблиця 3.7—Тестування зі зменшеною кількістю ОЗУ

Показник	Значення	Примітка
Завантаженість ЦП	68%	еталон
Завантаженість ОЗУ	536 МБ	еталон
Завантаження відео ядра	11%	еталон
Середній FPS	24	Менше еталону
Кількість зафіксованих людей	360	еталон
Помилки в логах	0	еталон

В ході тестування робота програми порушена не була, але були помітні гальмування системи, що може негативно впливати на роботу системи відеоспостереження, тож не рекомендується використовувати систему на базі ОС Windows 10 64х з меш ніж 6 Gb ОЗУ, можливим вирішенням цієї проблеми є оптимізація самої ОС.

Тестування з втратою відеопотоку:

Для цього тесту на 15 хвилині роботи було відключено камеру на 10 хв після чого знову підключено

Хід тестування:

До відключення камери обробка відбувалась в межах еталону, після відключення камери обробка завершилась і в лог-файлі з’явилось повідомлення

про не можливість зчитати кадр, після підключення камери потрібно знову вмикати роботу ядра.

Для усунення цієї проблеми пропонується видалити обробку виключення про не можливість зчитати кадр, та створити обробку очікування підключення камери.

Тестування з стороннім ПЗ:

Перед запуском системи буде відкрито браузер Firefox та буде відкриватись по 1 вкладці з відеороликом в якості 1080p результати будуть записуватись в таблицю, нові вкладки будуть відкриватись до тих пір поки не буде завантажено всю ОЗУ або до критичної помилки.

Кількість вкладок	Результат
1	Було помічено загальні підвисання: браузеру та ОС, FPS в ядрі програми близько 5-10, але фіксування продовжувалось в межах еталону
2	Після того як була заповнена ОЗУ, вся програма(GUI, ядро) була зачинена самою ОС

Результати показали, що оперативна пам’ять є одним із най вразливіших ресурсів системи. При її перевантаженні Windows самостійно приймає рішення про завершення процесів згідно з власними алгоритмами. Пріоритет критичних процесів не враховується програмно, тому неможливо примусити систему закривати інші процеси замість програми розпізнавання. Тож не рекомендується запускати програми, що можуть використовувати велику кількість ОЗУ, це може призвести до закриття системи та втрати статистичних даних.

Проведення загально навантажувального тесту:

Як було сказано вище останнім тестом буде включення всіх несприятливих факторів.

Хід тестування:

Після включення всіх несприятливих факторів, програма запускає GUI але при запуску ядра, Windows примусово зачиняє програму розпізнавання, що не дає змогу провести тестування.

Висновок навантажувального тестування:

Після проведення всіх тестувань можна сказати, що система розпізнавання може працювати на досить обмеженому АЗ, але за умови, що буде достатня кількість ОЗУ так як вона є найбільш вразливим місцем.

Зальною рекомендацією буде використання або окремого АЗ для системи розпізнавання та системи відеоспостереження, щоб уникнути проблем з перевантаженням ОЗУ або запускати ці системи у ізольованому середовищі.

Також під час цього тестування була виявлена проблема пов’язана з відсутністю механізму очікування підключення камер, що є некритичною помилкою, але все ж таки вагомою, впровадження такого механізму значно підвищить надійність, стабільність та автономність роботи системи.

3.6 Оцінка точності розпізнавання

Метою оцінки точності є оцінка роботи моделі YOLOv5, з метою оцінити стабільність та кількість хибних спрацювань, а також обґрунтувати доцільність або не доцільність використання цієї моделі у схожих проектах.

Оцінка буде формуватися після створення специфічних умов, а саме точність розпізнавання на різній відстані, при змінах освітлення, швидкості руху об’єкту, частковий показ об’єкту на кадрі.

Оцінка формується з: індексу довіри, фактичного розпізнавання за формулою:

*(Індекс довіри)*50+ фактичне розпізнавання (відбулось 50, не відбулось 0)*, тобто максимальний бал 100.

Оцінка розпізнавання об’єкту при різній відстані до нього:

Відстань	Оцінка розпізнавання
1м	95
3м	94,5
5м	85

Висновок: модель показує чудові результати на різній відстані до об’єкту

Оцінка розпізнавання об’єкту при різному освітленні:

Примітка: тестування відбувалось на відстані 3м до об’єкту.

Освітлення	Оцінка розпізнавання
Сонячне світло	95
Світло кімнатної люстри	95
Світло настільної лампи	90

Висновок: модель також показує чудовий результат навіть при поганому освітленні.

Оцінка розпізнавання об’єкту при різній швидкості об’єкту:

Примітка: тестування відбувалось на відстані 3м до об’єкту, освітлення: кімнатна люстра.

Швидкість	Оцінка розпізнавання
Об’єкт стоїть нерухомо	95
Ходьба	93.5
Пробіжка	93

Висновок: навіть про русі об’єкту модель показує чудові результати в рамках погрішності, не велика різниця між бігом та ходьбою пояснюється алгоритмом роботи, а саме не обробкою змазаного зображення а цілого кадру де досить чітко видно об’єкт.

Оцінка розпізнавання об’єкту при різній швидкості об’єкту:

Частина об’єкту що видно	Оцінка розпізнавання
Людина до поясу	95

Половина торсу, обличчя, та рука	90
Рука від плеча до кінчиків пальців	80
Кисть	87,5

Висновок: Модель показала чудовий результат навіть якщо людина не повністю в кадрі.

Висновок тестування:

Результати тестування показують, що система на основі YOLOv5 забезпечує високу точність розпізнавання об’єктів у більшості типових умов. Точність і повнота знаходяться в межах 85–95%, що дозволить використовувати систему для відеоспостереження з високим рівнем достовірності. У складних умовах можливе часткове зниження точності, що може бути компенсовано оптимізацією параметрів моделі або заміною її на більш потужну версію.

3.7 Висновки за розділом 3

В цьому розділі було проведено: функціональне, сценарне, навантажувальне тестування, а також була проведена власна оцінка точності. Отримані результати свідчать про стабільну та коректну роботу системи, високий рівень точності а також можливість роботи на середньому обладнанні. Виявлені вузькі місця, а саме критичність обсягу ОЗУ та відсутність автоматичного перепідключення — не є перешкодою для базового використання системи, але можуть бути усунені в наступних версіях.

ВИСНОВКИ

В рамках кваліфікаційної роботи, було розроблена та протестована система розпізнавання об’єктів в системах відеоспостереження. Метою даної роботи було створення програмного забезпечення, що буде: розпізнавати людей в реальному часі з камер відеоспостереження, зберігати скріншоти з виявленими об’єктами, вести статистику та мати зручний графічний інтерфейс для налаштування параметрів.

Під час виконання роботи було досягнуто:

- Було проаналізовано наявні програмні продукти на ринку, після чого визначено їхні недоліки та функціональні обмеження. На основі отриманих висновків було сформовано вимоги до власної системи, яка поєднує необхідний функціонал і усуває виявлені недоліки.
- Проведено аналіз сучасних алгоритмів комп’ютерного зору. На основі проведеного аналізу було обґрунтовано вибір моделі YOLOv5, що поєднує в собі швидкість та точність.
- Реалізований досить зручний графічний інтерфейс користувача, для налаштування параметрів системи, що робить програму доступною для користувачів без технічної підготовки.
- Створено ядро системи, що обробляє відеопотік та розпізнає в ньому об’єкти
- Роботи системи з конфігураційним файлом
- Проведено комплексне тестування, функціональне, сценарне, навантажувальне та оцінка точності. Отримані результати підтвердили працездатність системи у реальних умовах
- Виявлені вузькі місця системи та не критичні помилки

Отримані результати свідчать про те, що система відповідає всім функціональним вимогам і може використовуватись для базового відеоаналізу в побутових та напів-професіональних умовах.

Робота має практичне значення, оскільки може розгортатись на не складному обладнані, без використання серверних рішень та без залучення висококваліфікованих фахівців.

Після проведення розробки та тестування, були запропоновані такі варіанти масштабування:

Варіант 1. Open Source проект

Суть цього варіанту в тому щоб надати загальності проекту де будь хто може завантажити собі проект, та модифікувати його під свої потреби таким чином створивши Fork цього проекту. Відомими проектами такого типу є сімейство ОС Linux, у цієї ОС є неймовірна кількість Fork-ів, і кожен з них має свої переваги та недоліки.

Варіант 2. Створення «легкої» та «важкої» версії продукту

Суть цього варіанту в тому щоб:

Для «легкої» версії підібрати більш швидку мову програмування наприклад C++, та більш швидку модель розпізнавання і таким чином створити продукт що зможе розгортатись на дуже старому обладнані таким чином модернізувати старі системи відеоспостереження.

А для «важкої» версії додати ще декілька налаштувань, багатопоточну обробку, можливість запису відео, вибрати більш досконалу модель, створити більш кращий з боку дизайну GUI, таким чином створивши застосунок що вже сам буде системою відеоспостереження з розпізнаванням об’єктів для великих систем.

Академічна доброчесність

Під час виконання кваліфікаційної роботи частково використовувалися інструменти генеративного штучного інтелекту (зокрема, ChatGPT) для підготовки структуризації тексту та формування варіантів висновків. Усі технічні рішення, реалізація програмного забезпечення, тестування, аналіз та остаточне формулювання висновків виконано автором самостійно. Робота відповідає принципам академічної доброчесності.

Таким чином, мету кваліфікаційної роботи було досягнуто повністю, а всі завдання — реалізовано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Інтернет-ресурси:

1. MotionEye [Електронний ресурс]. – Режим доступу: **https://github.com/ccrisan/motioneye** (дата звернення: 23.04.2025).
2. ZoneMinder [Електронний ресурс]. – Режим доступу: **https://zoneminder.com** (дата звернення: 23.04.2025).
3. Ivideon [Електронний ресурс]. – Режим доступу: **https://www.ivideon.com** (дата звернення: 23.04.2025).
4. DeepStack [Електронний ресурс]. – Режим доступу: **https://deepstack.cc** (дата звернення: 23.04.2025).
5. Milestone XProtect [Електронний ресурс]. – Режим доступу: **https://www.milestonesys.com** (дата звернення: 23.04.2025).
6. YOLOv5. Ultralytics [Електронний ресурс]. – Режим доступу: **https://github.com/ultralytics/yolov5** (дата звернення: 26.04.2025).
7. OpenCV Documentation [Електронний ресурс]. – Режим доступу: **https://docs.opencv.org/** (дата звернення: 26.04.2025).
8. Tkinter – Python GUI library [Електронний ресурс]. – Режим доступу: **https://docs.python.org/3/library/tkinter.html** (дата звернення: 27.04.2025).
9. Python. Official Website [Електронний ресурс]. – Режим доступу: **https://www.python.org** (дата звернення: 26.04.2025).
10. Ultralytics Documentation [Електронний ресурс]. – Режим доступу: **https://docs.ultralytics.com/** (дата звернення: 03.05.2025).

Літературні джерела:

11. Пац О. Розробка системи виявлення об'єктів на основі YOLOv5 : кваліфікаційна робота бакалавра / О. Пац ; наук. керівник В. Литвин. – Львів : Львівський нац. ун-т ім. І. Франка, 2020. – 42 с. – Режим доступу: **https://ami.lnu.edu.ua/wp-content/uploads/2020/11/Pats.pdf** (дата звернення: 06.03.2025).

Додаток А Порівняльна таблиця сучасних програмних рішень

Назва	Ліцензія	АЗ	Кількість одночасно підключених камер	Користувацький інтерфейс	Складність налаштування (1–5)	Необхідність інтернет з'єднання
MotionEye	Відкрита	Raspberry Pi або ПК з Linux	до 4	Так	4	Ні
ZoneMinder	Відкрита	Linux, Apache, MySQL	10 та більше	Веб-інтерфейс	2	Ні
Ivideon	Умовно-безкоштовна	Windows, Android, iOS	1 (безкоштовно) 16 та більше (платно)	Так	5	Так
DeepStack	Відкрита	ПК, Python, Docker	Залежить від інтеграції	Ні (API/консоль)	3	Ні
Milestone XProtect	Комерційна	Windows	8–64+ залежно від ліцензії	Так	3	Ні