Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

МЕТОД ТА ЗАСОБИ МОНІТОРИНГУ ДІЯЛЬНОСТІ ПРАЦІВНИКІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав студент 2 курсу, групи 1КІ-24м
Спеціальності 123 — Комп'ютерна інженерія
_____ Данько І. П.

Керівник к.т.н., доц. каф. ОТ
_____ Богомолов С. В.
"12" 12 2025 р.

Опонент к.ф.-м.н., проф. каф. МБІС
_____ Шиян А. А.
"12" 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
_____
"17" 12 2025 р.

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н. Азаров О.Д.
"25" вересня 2025 р.

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

студенту **Даньку Івану Петровичу**

1 Тема роботи: Метод та засоби моніторингу діяльності працівників з використанням штучного інтелекту, керівник роботи_ к.т.н., доцент кафедри ОТ Богомолов С.В. затверджені наказом ВНТУ від 24.09.2025 року № 313

2 Строк подання студентом роботи 4.12.2025 р.

3 Вихідні дані до роботи: відеопотоки з камер спостереження в реальному часі. Основні технології: Python , OpenCV. Архітектура системи: розподілена, клієнт—серверна (edge computing).Методи аналізу: Motion Detection (як тригер) , YOLO (для ідентифікації об'єктів) , cvCamShift (для відстеження об'єктів).

4 Зміст текстової частини: аналіз методів та засобів створення автоматизованих систем моніторингу, дослідження методів та моделей аналізу відеопотоку (Motion Detection, MeanShift, CamShift, YOLO), проєктування та розробка автоматизованої системи моніторингу на основі ШІ (загальна архітектура, моделювання потоків даних, розробка клієнтського агента та серверної частини API), економічне обґрунтування розробки програмного продукту.

5 Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових кресл
загальна архітектура системи моніторингу ,моделювання потоків даних (DFD) в сис
розробка діаграми компонентів, приклади роботи методів аналізу відеопотоку (Mo
Detection, cvMeanShift, YOLO).

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця — 1 Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завда прий
1-4	Богомолов Сергій Віталійович к.т.н., доцент каф. ОТ		
5	Ратушняк Ольга Георгіївна к.т.н., доцент каф. ЕПВМ		
Нормоконтроль	Швець Сергій Ілліч асистент каф. ОТ		

7 Дата видачі завдання 25.09.2025

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця — 2 Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Приміт
1	Аналіз сучасного рівня інформаційних технологій розпізнавання об'єктів на зображеннях. Постановка задач дослідження		викона
2	Побудова моделей розпізнавання об'єктів на зображеннях на основі нейронної мережі та функціонування нейронної мережі		викона
3	Практичне застосування та оцінка ефективності розроблених моделей		викона
4	Підготовка економічної частини		викона
5	Апробація та/або впровадження результатів дослідження		викона
6	Оформлення пояснювальної записки, графічного матеріалу та презентації		викона
7	Попередній захист		викона
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків		викона

Студент _____ Данько І. П.

Керівник _____ к.т.н., доц. каф. ОТ Богомолов С. В.

АНОТАЦІЯ

УДК 004.931

Данько І. П. Метод та засоби моніторингу діяльності працівників з використанням штучного інтелекту. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2025 — 110 с.

Магістерська кваліфікаційна робота містить 9 рисунків, 9 лістингів та 12 таблиць.

Розроблено метод та засоби для автоматизованої системи моніторингу діяльності працівників, розглянуто принципи аналізу відеопотоків у реальному часі для ідентифікації, відстеження та аналізу поведінки персоналу, виконано програмну реалізацію системи мовою Python із застосуванням клієнт—серверної архітектури. Реалізовано каскадний конвеєр на базі бібліотеки OpenCV, що поєднує методи Motion Detection, YOLO та cvCamShift.

Ключові слова: моніторинг персоналу, штучний інтелект, комп’ютерний зір, YOLO, cvCamShift, Motion Detection, клієнт—серверна архітектура, Python, OpenCV.

ABSTRACT

Danko I. P. Method and means of monitoring employee activity using artificial intelligence. Master's qualification work in specialty 123 - Computer Engineering, Vinnytsia: VNTU, 2025 — 110p.

The Master's qualification work contains 9 figures, 9 listings, and 12 tables.

Developed method and means for an automated employee activity monitoring system, considered principles of real—time video stream analysis for identification, tracking, and analysis of personnel behavior, performed software implementation of the system in Python using client—server architecture. Implemented a cascade pipeline based on the OpenCV library, combining Motion Detection, YOLO, and cvCamShift methods.

Keywords:personnel monitoring, artificial intelligence, computer vision, YOLO, cvCamShift, Motion Detection, client—server architecture, Python, OpenCV.

ЗМІСТ

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ СТВОРЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ МОНІТОРИНГУ 10
1.1 Аналіз сучасних підходів до моніторингу діяльності працівників .. 10
1.2 Порівняльний аналіз технологій для створення систем моніторингу 14
1.3 Огляд проєктування автоматизованої системи моніторингу 18
1.4 Постановка задач дослідження 22
2 МЕТОДИ ТА МОДЕЛІ АНАЛІЗУ ВІДЕОПОТОКУ 28
2.1 Метод Motion Detection 28
2.2 Метод cvMeanShift (Метод середнього зсуву) 31
2.3 Метод cvCamShift (Безперервно—адаптивний середній зсув) 35
2.4 Метод YOLO 39
3 ПРОЄКТУВАННЯ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ 43
3.1 Методологія проєктування та реалізації системи моніторингу 43
3.1.1 Метод каскадної обробки відеопотоку як основа зменшення обчислювального навантаження 44
3.1.2 Обґрунтування вибору методів детекції та трекінгу об’єктів 46
3.3.2 Метод реалізації первинної обробки відеопотоку 50
3.3.3 Метод інтеграції нейромережевої детекції об’єктів 51
3.3.4 Засоби реалізації трекінгу та контролю заборонених зон 52
3.3.5 Метод формування подій та взаємодії з серверною частиною 54
3.4 Проєктування та реалізація серверної частини автоматизованої системи моніторингу 54
3.5 Оцінка ефективності та стабільності роботи автоматизованої системи моніторингу 58

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ 62
4.1 Методика експериментальних випробувань, тестування 62
4.2 Чисельні результати експериментальних досліджень, натурних випробувань 67
4.3 Результати впровадження і експлуатації, рекомендації щодо вдосконалення 72
5 ЕКОНОМІЧНА ЧАСТИНА 78
5.1 Оцінювання комерційного потенціалу розробки 78
5.2 Прогнозування витрат на виконання науково—дослідної роботи .. 85
5.3 Розрахунок економічної ефективності науково—технічної розробки 92
ВИСНОВКИ 97
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 99
ДОДАТОК А Технічне завдання 101
ДОДАТОК Б Протокол перевірки кваліфікаційної роботи 106
ДОДАТОК В Загальна архітектура системи моніторингу 107
ДОДАТОК Г Моделювання потоків даних (DFD) 108
ДОДАТОК Д Діаграма компонентів 109
ДОДАТОК Е Приклади роботи методів аналізу відеопотоку 110
ДОДАТОК Ж Лістинг коду 112

ВСТУП

Актуальність теми дослідження полягає в тому, що у сучасних умовах стрімкого розвитку цифрових технологій зростає потреба в автоматизації процесів контролю, безпеки та аналітики діяльності працівників у різних сферах — від промислових підприємств до освітніх і державних установ. Ефективність управління трудовими ресурсами безпосередньо залежить від якості моніторингу та об'єктивної оцінки робочої активності.

Завдяки розвитку технологій комп'ютерного зору та штучного інтелекту з'явилась можливість реалізувати системи, здатні автоматично розпізнавати події, відстежувати дії людей і аналізувати поведінкові показники без втручання людини.

Такі системи дозволяють вирішити низку задач: ідентифікацію працівників, фіксацію присутності, аналіз робочої активності, контроль дотримання техніки безпеки та інше. Використання алгоритмів комп'ютерного зору значно підвищує точність і швидкість обробки інформації, що робить процес моніторингу об'єктивним і надійним.

Отже, розроблення автоматизованої системи моніторингу діяльності працівників за допомогою алгоритмів комп'ютерного зору та штучного інтелекту є актуальним напрямом дослідження, що поєднує аналітичні, інженерні та програмні аспекти сучасних інформаційних технологій.

Метою є аналіз, розробка концепції та оцінка ефективності методів побудови систем моніторингу персоналу із застосуванням комп'ютерного зору та алгоритмів штучного інтелекту.

Для досягнення мети необхідно необхідно вирішити такі **задачі**:

— проаналізувати сучасні підходи до автоматизованого моніторингу персоналу;

— дослідити принципи роботи алгоритмів комп'ютерного зору та штучного інтелекту, що використовуються для відстеження діяльності людей;

— здійснити порівняльний аналіз технологій і методів реалізації подібних систем;

— розробити загальну структурну модель автоматизованої системи;

— визначити критерії оцінки ефективності та можливості інтеграції з існуючими інформаційними середовищами.

Об’єкт дослідження — процес автоматизованого моніторингу діяльності працівників.

Предмет дослідження є алгоритми комп’ютерного зору та штучного інтелекту, що забезпечують виявлення, ідентифікацію та аналіз поведінкових параметрів.

Методами дослідження є аналітичні методи, моделювання, порівняльний аналіз алгоритмів глибинного навчання та методів обробки зображень.

Новизна результатів полягає у формуванні узагальненої моделі автоматизованої системи моніторингу з адаптивним інтелектуальним ядром, здатним до самовдосконалення на основі зворотного зв’язку.

Основні результати пройшли **апробацію** та доповідались на науково—практичній конференції «Молодь у науці: дослідження, проблеми, перспективи (МН—2025)» та опубліковані у збірнику тез.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ СТВОРЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ МОНІТОРИНГУ

1.1 Аналіз сучасних підходів до моніторингу діяльності працівників

Глибинний аналіз еволюції засобів моніторингу дозволяє виділити фундаментальну проблему, притаманну раннім підходам, яку у науковій літературі класифікують як обмеження когнітивного ресурсу оператора. Дослідження у сфері інженерної психології свідчать, що ефективність візуального контролю з боку людини стрімко деградує в умовах монотонної діяльності.

Вже після двадцяти хвилин безперервного спостереження за екранами моніторів здатність оператора фіксувати цільові події знижується до критичного рівня, що призводить до пропуску значної частини інцидентів. Саме цей фактор став каталізатором переходу від парадигми пасивного спостереження до активної відеоаналітики, де роль людини зміщується від безпосереднього спостерігача до верифікатора подій, попередньо відібраних автоматизованою системою.

Водночас впровадження систем другого покоління, що базувалися на детермінованих алгоритмах обробки сигналів, виявило проблему так званого семантичного розриву між низькорівневими ознаками зображення та високорівневим розумінням сцени. Класичні методи, такі як детекція руху або перетин віртуальних ліній, оперують змінами інтенсивності пікселів і не здатні розрізняти контекст події.

Це призводить до генерації надмірної кількості хибних спрацювань у виробничих умовах, де наявні вібрації, зміни освітлення або рух технологічного обладнання. Відсутність семантичного розуміння сцени унеможливлює вирішення завдань, пов'язаних із оцінкою саме трудової активності, оскільки такі

системи не можуть диференціювати, наприклад, продуктивну роботу працівника за верстатом від простого перебування поруч із ним.

Сучасний етап розвитку, пов'язаний із впровадженням інтелектуальних систем на базі глибоких нейронних мереж, докорінно змінює саму філософію моніторингу, трансформуючи її з інструменту безпеки в інструмент бізнес-аналітики. Застосування згорткових нейронних мереж дозволяє виконувати детекцію та класифікацію об'єктів у складних умовах, забезпечуючи інваріантність до масштабу, ракурсу зйомки та перекриттів.

Це відкриває можливості для автоматизованого нормування праці, побудови теплових карт переміщення персоналу, аналізу ергономіки робочих місць та виявлення аномалій у технологічних процесах. Відеокамера в такій архітектурі перестає бути просто пристроєм фіксації, а перетворюється на універсальний інтелектуальний сенсор, що генерує структуровані дані для прийняття управлінських рішень.

Окремої уваги в контексті аналізу сучасних підходів заслуговує етико-правовий аспект впровадження автоматизованого моніторингу. Зростання обчислювальних потужностей та точності розпізнавання біометричних даних актуалізує питання захисту приватності працівників. Сучасні системи проєктуються з урахуванням принципів Privacy by Design, що передбачає впровадження механізмів динамічного маскування особистості на відеопотоці, шифрування каналів передачі даних та суворого розмежування прав доступу.

Такий підхід дозволяє збалансувати інтереси роботодавця щодо контролю ефективності та права працівників на конфіденційність, що є обов'язковою вимогою при інтеграції подібних рішень у корпоративну інфраструктуру відповідно до сучасних стандартів захисту даних.

Саме тому системи спостереження за працівниками зазнали значної еволюції — від простих відеокамер до інтелектуальних рішень на основі

алгоритмів комп’ютерного зору та штучного інтелекту, розвиток таких систем можна поділити на три основні етапи:

— пасивні системи;

— аналітичні системи;

— інтелектуальні системи.

Пасивні системи це — перше покоління систем спостереження, яке виконувало виключно функцію відеофіксації. Їх основне завдання — записувати зображення з камер спостереження для подальшого ручного перегляду.

Такі системи не мали аналітичних можливостей і повністю залежали від людини—оператора, який переглядав матеріали, оцінював ситуацію та приймав рішення.

Переваги цього підходу — простота, низька вартість і зрозуміла структура. Недоліки — значна залежність від людського фактора, неможливість оперативного реагування та потреба у великих витратах часу на перегляд відео.

Аналітичні системи це — наступний етап розвитку пов’язаний із появою алгоритмів обробки відео, які дозволили виявляти рух, зміни у кадрі, визначати контури об’єктів та розпізнавати обличчя. Такі системи почали застосовувати методи комп’ютерного зору на базовому рівні — наприклад, аналіз потоків кадрів у реальному часі, виявлення присутності людини або її активності.

Аналітичні системи могли автоматично сповіщати оператора про підозрілі події, рухи у заборонених зонах або відсутність працівника на робочому місці.

Проте рівень точності залишався недостатнім, адже алгоритми не розуміли контексту ситуацій, а лише реагували на зміни зображення.

Інтелектуальні системи це — сучасний етап розвитку моніторингових технологій характеризується впровадженням штучного інтелекту, машинного навчання та глибинних нейронних мереж.

Інтелектуальні системи здатні не лише фіксувати об’єкти, а й аналізувати поведінку людей, розпізнавати їхні дії зображено на рисунку 1.1 (робота за

комп’ютером, пересування, взаємодія з обладнанням), оцінювати ефективність праці, виявляти відхилення або небезпечні ситуації.

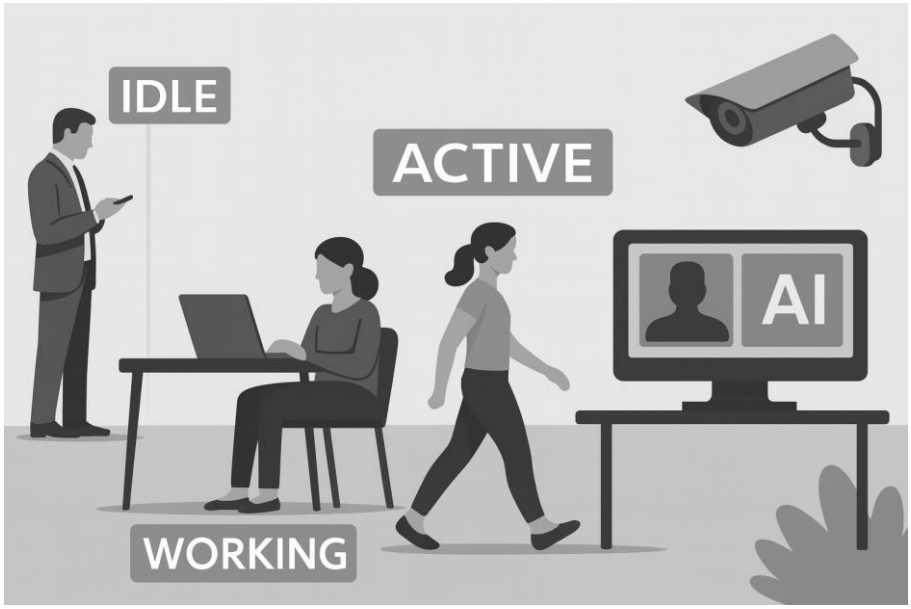

Рисунок 1.1 — Приклад аналізу поведінки

Такі системи інтегруються з корпоративними мережами, хмарними сховищами та аналітичними панелями, що дозволяє отримувати статистику в реальному часі. Інтелектуальні системи часто поєднують декілька технологічних підсистем:

— модулі комп’ютерного зору (розпізнавання осіб, виявлення рухів, аналіз дій);

— машинне навчання (класифікація поведінкових паттернів);

— аналітичні сервіси (звіти, графіки, прогнозування);

— засоби безпеки (шифрування, контроль доступу).

Завдяки цьому забезпечується повний цикл аналітики — від отримання даних до формування управлінських рішень. Використання алгоритмів комп’ютерного зору має низку переваг:

— об’єктивність (виключення людського чинника при оцінці поведінки персоналу);

— оперативність (система реагує миттєво, надсилаючи сповіщення про відхилення);

— аналітичність (можливість аналізувати довготривалі тенденції, формувати звіти);

— безпека (фіксація несанкціонованого доступу чи порушення техніки безпеки);

— оптимізація (покращення розподілу робочого часу та ресурсів).

Таким чином, сучасні автоматизовані системи моніторингу є результатом поєднання відеоаналітики, штучного інтелекту та хмарних технологій. Вони дозволяють підприємствам не лише контролювати, а й розуміти робочі процеси, виявляючи закономірності, неефективні дії та потенційні ризики. Перехід від пасивного спостереження до інтелектуального аналізу є ключовим кроком у розвитку корпоративної аналітики та управління персоналом.

1.2 Порівняльний аналіз технологій для створення систем моніторингу

Реалізація автоматизованих систем моніторингу діяльності працівників із використанням алгоритмів комп’ютерного зору та штучного інтелекту можлива за допомогою різних технологічних стеків, які відрізняються за продуктивністю, гнучкістю, масштабованістю та простотою інтеграції.

Python — найпопулярніше середовище для реалізації систем комп’ютерного зору та машинного навчання. Його відкритий екосистемний підхід забезпечує широкий набір бібліотек:

— OpenCV — для обробки зображень і відео, виявлення руху, трекінгу об’єктів;

— TensorFlow та PyTorch — для побудови, навчання і тестування нейронних мереж, включаючи згорткові архітектури (CNN), рекурентні (RNN) і трансформерні моделі;

— Mediapipe — для аналізу положення тіла, обличчя, жестів, що є особливо корисним при моніторингу активності людини.

C++ вирізняється високою швидкодією, що робить його придатним для створення систем, де важлива мінімальна затримка обробки відеопотоків у реальному часі. Його використання часто виправдане в задачах обробки великих обсягів даних із камер спостереження або вбудованих пристроїв (edge computing). Крім того, OpenCV має повноцінну реалізацію на C++, що дозволяє ефективно поєднувати швидкодію з гнучкістю налаштувань.

JavaScript (Node.js) застосовується для розробки вебінтерфейсів адміністрування, аналітики та візуалізації зібраних даних. У поєднанні з фреймворками React, Vue.js або Angular можна створювати інтерактивні панелі керування, які відображають стан робочих місць, рівень активності персоналу або події безпеки. Використання Node.js на сервері дає змогу швидко обробляти запити та передавати результати моніторингу в режимі реального часу через WebSocket.

Для виявлення та класифікації об’єктів у кадрі застосовуються сучасні архітектури глибинного навчання:

— YOLO (You Only Look Once) — висока швидкість та точність, ефективна для детекції людей і визначення їхніх поз у реальному часі;

— SSD (Single Shot MultiBox Detector) — збалансований варіант між продуктивністю та точністю для систем середньої складності;

— Faster R—CNN — глибока мережа з підвищеною точністю, яка використовується в системах, де важлива якість розпізнавання;

PoseNet або OpenPose — для аналізу скелетних моделей людини та відстеження рухів кінцівок.

Окрім вибору самої архітектури нейронної мережі, критично важливим інженерним аспектом розробки є вибір середовища виконання (Inference Engine), яке забезпечує запуск навченої моделі на цільовому обладнанні. Безпосереднє виконання моделей у середовищах навчання, таких як PyTorch або TensorFlow, часто є неефективним з точки зору утилізації апаратних ресурсів. Для оптимізації роботи системи в продакшн-середовищі доцільно використовувати спеціалізовані фреймворки прискорення інференсу. Зокрема, бібліотека NVIDIA TensorRT дозволяє виконувати глибоку оптимізацію графу обчислень, злиття шарів та квантування ваг (наприклад, до форматів FP16 або INT8), що забезпечує максимальну продуктивність на графічних процесорах NVIDIA.

Для систем, що базуються на центральних процесорах (CPU) або інтегрованій графіці від Intel, найбільш ефективним рішенням є використання інструментарію OpenVINO Toolkit, який дозволяє адаптувати нейромережеві моделі для роботи на стандартній офісній техніці з високою швидкістю.

Універсальним підходом до вирішення проблеми сумісності різних фреймворків є використання формату ONNX (Open Neural Network Exchange) та середовища виконання ONNX Runtime. Цей інструмент дозволяє уніфікувати процес розгортання моделей, забезпечуючи кросплатформність та підтримку різнорідного апаратного забезпечення без необхідності переписування коду.

Враховуючи вимоги до масштабованості та необхідність забезпечення роботи системи на різному обладнанні, використання подібних засобів оптимізації інференсу є обов’язковою умовою для досягнення стабільної частоти кадрів при обробці відеопотоку в реальному часі.

Вибір конкретного інструменту оптимізації залежить від апаратної конфігурації кінцевого вузла моніторингу, проте загальна стратегія розробки передбачає відділення логіки навчання моделі від логіки її виконання, що підвищує загальну надійність та ефективність програмного комплексу.

Важливим компонентом є вибір бази даних для збереження результатів аналізу відео.

— PostgreSQL — реляційна база з підтримкою геоданих і JSON, зручна для структурованого зберігання метаданих про події.

— MongoDB — документно—орієнтована система, яка добре підходить для зберігання неструктурованих результатів, зокрема даних нейронних моделей.

— InfluxDB — оптимізована для часових рядів, що дозволяє ефективно фіксувати зміни активності працівників у часі.

Для подальшої обробки результатів моніторингу використовуються інструменти аналітики та візуалізації, такі як Grafana, Kibana або інтегровані модулі в межах вебінтерфейсу.

З точки зору архітектури, системи моніторингу можуть бути централізованими — коли вся обробка даних здійснюється на центральному сервері. Такі системи простіші в адмініструванні, проте мають нижчу масштабованість і можуть створювати вузьке місце при збільшенні кількості камер, або роозподіленими (edge—based) — коли частина обробки (наприклад, попереднє розпізнавання або фільтрація кадрів) відбувається безпосередньо на пристроях або локальних вузлах. Це зменшує навантаження на центральний сервер і забезпечує швидшу реакцію системи. Така архітектура особливо актуальна для великих підприємств із кількома майданчиками спостереження.

Для забезпечення масштабованості системи доцільно використовувати хмарні платформи (AWS, Google Cloud, Azure) або оркестраційні технології (Docker, Kubernetes), які дозволяють гнучко керувати ресурсами, швидко розгортати нові модулі та оновлювати алгоритми без зупинки всієї системи.

Таким чином, вибір технологічного стеку залежить від пріоритетів системи:

— якщо важлива швидкість прототипування та гнучкість — доцільно обрати Python;

— якщо ключова вимога — висока продуктивність — C++;

— якщо потрібно інтерактивне керування та віддалений доступ — JavaScript/Node.js у поєднанні з вебфреймворками.

Поєднання цих технологій у єдиній архітектурі дозволяє створити комплексну, надійну та масштабовану систему моніторингу працівників, яка поєднує відеоаналітику, штучний інтелект та веб—інтерфейси управління.

1.3 Огляд проєктування автоматизованої системи моніторингу

Проєктування автоматизованої системи моніторингу діяльності працівників на основі технологій комп’ютерного зору та штучного інтелекту є складним міждисциплінарним процесом, який поєднує аналітику бізнес—потреб, інженерне моделювання, алгоритмічну оптимізацію, кібербезпеку та інтеграцію з корпоративною інфраструктурою.

Основна мета такого проєктування — створення системи, здатної в реальному часі виявляти, аналізувати та класифікувати поведінку працівників, забезпечуючи баланс між ефективністю контролю та збереженням приватності.

На початковому етапі формується концепція та обґрунтування доцільності впровадження системи. Для цього проводиться аналіз робочих процесів підприємства, визначаються зони спостереження, потенційні сценарії використання та очікувані показники ефективності, серед типових завдань:

— контроль присутності працівників на робочому місці;

— аналіз рівня активності та часу простоїв;

— відстеження виконання конкретних дій (робота за комп’ютером, рух по офісу, взаємодія з обладнанням);

— фіксація подій, що можуть мати значення для безпеки або оптимізації процесів (падіння, порушення трудової дисципліни, сторонні особи у заборонених зонах).

На цьому етапі розробляється специфікація вимог — документ, у якому фіксуються функціональні, нефункціональні та технічні характеристики майбутньої системи. Визначаються граничні умови: кількість камер, частота кадрів, якість відео, допустима затримка аналізу, цільова точність розпізнавання, пропускна здатність мережі та доступні ресурси серверної інфраструктури.

Після етапу формалізації вимог здійснюється розроблення архітектурної моделі, яка визначає загальну логіку взаємодії компонентів системи.

Зазвичай система має багаторівневу архітектуру, що включає такі основні модулі:

— модуль збору даних отримує відеопотоки з камер спостереження, сенсорів або інших пристроїв, здійснює базову синхронізацію та передачу потоків у систему обробки;

— модуль попередньої обробки виконує нормалізацію кадрів (зміна розміру, усунення шумів, стабілізація зображення, баланс білого), а також виділення ключових зон інтересу (ROI);

— модуль розпізнавання та класифікації — застосовує алгоритми комп’ютерного зору для виявлення осіб, фігур чи об’єктів, аналізує їхню поведінку за допомогою моделей глибинного навчання (наприклад, YOLO, OpenPose, DeepSort, MediaPipe BlazePose);

— модуль обробки подій та аналітики — на основі виявлених дій формує логічні висновки: “активний”, “пасивний”, “працює”, “відсутній”, “порушення”.

— модуль збереження та управління даними — відповідає за фіксацію результатів у базах даних, ведення журналів подій, архівування відео та метаданих;

— інтерфейс користувача (UI) — забезпечує візуалізацію даних, доступ до аналітики, сповіщень і статистики.

Важливим принципом є модульність та масштабованість: система має легко адаптуватися до зміни кількості камер, серверів чи користувачів. Для цього часто застосовують мікросервісну архітектуру або контейнеризацію (Docker, Kubernetes), що дозволяє гнучко керувати навантаженням.

Вибір алгоритмічних і програмних засобів, на цьому етапі визначаються технологічні рішення для реалізації кожного модуля.

Для розпізнавання об’єктів обираються згорткові нейронні мережі (CNN) або гібридні моделі, які поєднують глибоке навчання з традиційними методами (HOG, SVM).Алгоритми типу YOLOv8 або Detectron2 забезпечують високу швидкість обробки потокового відео, що є критично важливим для режиму реального часу.

Для оцінки поз та рухів використовуються архітектури OpenPose або BlazePose, які аналізують скелетну модель людини.

Залежно від вимог до точності, обчислювальних ресурсів та умов освітлення, можуть застосовуватися:

— кластерні обчислення для паралельної обробки кількох потоків;

— GPU—акселерація (CUDA, TensorRT) для прискорення інференсу нейронних мереж;

— edge—аналітика (обробка безпосередньо на пристрої) для зменшення навантаження на центральний сервер.

Оскільки система працює з відео— і біометричними даними, важливим аспектом є захист персональної інформації. На цьому етапі передбачаються такі заходи:

— шифрування потоків і збережених даних (наприклад, за допомогою AES—256 або TLS);

— обмеження доступу до відеоматеріалів відповідно до рівня прав користувача;

— ведення журналів аудиту для фіксації всіх дій користувачів;

— відповідність вимогам законодавства про захист персональних даних (GDPR, Закон України “Про захист персональних даних”).

Крім технічних заходів, розробляються етичні політики використання системи, щоб уникнути зловживань і забезпечити прозорість моніторингу для працівників.

Інтеграція з корпоративними сервісами має взаємодіяти з іншими інформаційними системами організації.

Інтеграційний рівень передбачає створення API або шлюзів для обміну даними із:

— HR—платформами (автоматичне ведення табеля робочого часу, облік запізнень, статистика присутності);

— CRM чи ERP—системами (вплив активності на продуктивність відділів);

— BI—платформами (візуалізація даних у вигляді графіків, дашбордів і KPI—аналітики);

— системами безпеки (автоматичне сповіщення охорони про інциденти).

Також проводиться налаштування взаємодії між серверами зберігання, аналітичними модулями та веб—інтерфейсом через стандартизовані протоколи — REST API, MQTT або WebSocket.

Тестування, оптимізація та розгортання — фінальний етап охоплює верифікацію системи в умовах, максимально наближених до реальної експлуатації.

Проводяться функціональні, навантажувальні та стрес—тести, що дозволяють перевірити стабільність системи під час одночасної роботи десятків камер.

Оцінюється точність класифікації дій, швидкість реакції на події, рівень помилкових спрацювань (False Positive Rate) та середній час обробки одного кадру, за результатами тестування здійснюється:

— коригування параметрів моделей;

— оптимізація коду;

— балансування навантаження між серверами;

— формування звіту про відповідність технічним вимогам.

Після цього відбувається розгортання у виробничому середовищі, налаштування резервного копіювання, моніторингу працездатності та системного журналювання подій.

1.4 Постановка задач дослідження

Після ґрунтовного узагальнення теоретичних засад, детального аналізу сучасних технологій комп’ютерного зору та критичного розгляду різноманітних архітектурних підходів до побудови автоматизованих систем моніторингу, постає об’єктивна необхідність сформулювати цілісний комплекс взаємопов’язаних наукових та прикладних задач. Їх послідовне розв’язання має забезпечити створення не просто програмного продукту, а ефективного інтелектуального рішення, здатного здійснювати відстеження діяльності працівників у реальному часі з високим рівнем достовірності та автономності.

Сутність дослідження полягає у розробленні комплексної моделі, яка здатна на системному рівні інтегрувати процеси попередньої обробки відеопотоків, алгоритмічне розпізнавання об’єктів, глибинне машинне навчання

та подальшу аналітичну обробку отриманих результатів у єдину, злагоджено працюючу екосистему.

Таке мультимодальне поєднання дозволяє вийти за межі простої фіксації фактів присутності людини на робочому місці та перейти до якісно нового рівня аналітики — оцінювання характеру активності, ідентифікації конкретних робочих дій, точного вимірювання часових інтервалів продуктивної роботи та періодів неактивності, а також виявлення специфічних поведінкових особливостей у межах контрольованого робочого простору.

Детальний аналіз вимог до сучасних систем моніторингу дозволив виявити ключове науково-технічне протиріччя, вирішення якого є необхідною умовою створення ефективного програмного продукту. Сутність цього протиріччя полягає у фундаментальному конфлікті між необхідністю забезпечення високої точності семантичного аналізу сцени, що досягається переважно за рахунок використання глибоких згорткових нейронних мереж, та жорсткими обмеженнями щодо швидкодії системи та доступних апаратних ресурсів.

З одного боку, об’єктивізація контролю вимагає застосування складних SOTA-архітектур (State-of-the-Art), здатних надійно розпізнавати об’єкти в умовах складного освітлення та часткових перекриттів. Однак використання таких моделей у їхньому базовому вигляді неминуче призводить до експоненційного зростання обчислювальної складності, що вимагає застосування дороговартісних серверних графічних прискорювачів та створення високошвидкісних каналів передачі даних. З іншого боку, економічна доцільність масового впровадження системи диктує необхідність використання недорогих периферійних обчислювальних пристроїв (Edge devices) та мінімізації навантаження на локальну мережу підприємства.

Спроба вирішити цю проблему шляхом простого зменшення розмірності моделей або використання класичних методів обробки зображень призводить до критичного зниження точності, зростання кількості помилок першого та другого

роду (хибних спрацювань та пропусків подій), що нівелює саму ідею інтелектуального моніторингу. Таким чином, виникає потреба у розробці та дослідженні нових підходів до організації обчислювального процесу, які дозволили б знайти оптимальний компроміс у трикутнику «точність — швидкість — ресурси». Це вимагає відходу від лінійного застосування алгоритмів на користь створення гібридних, адаптивних схем обробки даних, де інтенсивність обчислень динамічно змінюється залежно від інформаційної ентропії поточного кадру відеопотоку.

У цьому контексті передусім виникає нагальна необхідність у глибокій систематизації та класифікації існуючих алгоритмів і технологій комп'ютерного зору, які потенційно можуть бути імплементовані для аналізу кінематики руху людини, виявлення її координат і положення у тривимірному просторі, класифікації поточних станів та визначення типових патернів поведінки, що корелюють із виконанням посадових обов'язків. Ця частина дослідження охоплює не лише описовий, а й критичний порівняльний огляд методів, що базуються на сучасних згорткових нейронних мережах (CNN), методів розрахунку оптичного потоку для оцінки динаміки сцени, алгоритмів побудови та аналізу скелетних структур, а також механізмів багатооб'єктного трекінгу.

Особлива увага при цьому приділяється питанням досягнення оптимального балансу між точністю детекції та швидкодією алгоритмів, а також забезпеченню стійкості системи до таких дестабілізуючих факторів, як різкі зміни освітлення, складні ракурси та кути огляду, часткові перекриття об'єктів (оклюзії) та наявність візуальних перешкод у кадрі, оскільки саме ці фактори мають вирішальний вплив на надійність та стабільність системи в реальних умовах експлуатації.

Другою ключовою та логічно пов'язаною задачею є побудова детальної структурної моделі автоматизованої системи моніторингу. Ця модель має відобразити складну логічну та фізичну взаємодію між модулями збору

відеоданих, блоками попередньої фільтрації та нормалізації зображень, ядром аналітичного аналізу на базі нейромереж, підсистемою збереження структурованих метаданих та інтерфейсами користувацької візуалізації.

Розроблення такої розгалуженої структури дозволяє чітко демаркувати функціональні межі кожного компонента, формалізувати інформаційні потоки між ними, встановити необхідні алгоритмічні зв’язки та сформулювати конкретні вимоги до програмно-апаратної частини системи, включаючи обчислювальні потужності графічних прискорювачів та пропускну здатність мережі. Вже на етапі проєктування моделі критично важливо врахувати можливість подальшого масштабування та модернізації системи, що передбачає безболісне підключення нових камер, інтеграцію додаткових модулів специфічної аналітики або міграцію обчислень до хмарного середовища з використанням технологій паралельної обробки великих масивів даних.

Наступним важливим аспектом дослідження виступає експериментальна оцінка ефективності застосування методів штучного інтелекту безпосередньо для семантичного аналізу діяльності працівників. У цьому випадку йдеться не лише про визначення метрик точності розпізнавання (таких як mAP), а й про здатність алгоритмів адаптуватися до мінливих і непередбачуваних умов робочого середовища. Основною метою цього етапу є емпіричне встановлення оптимального компромісу між швидкодією системи (FPS), споживанням системних ресурсів та точністю класифікації дій персоналу.

Для цього здійснюється поглиблений порівняльний аналіз архітектур нейронних мереж різних поколінь, зокрема сімейства YOLO (v5—v8), SSD, Faster R-CNN, Detectron2 та спеціалізованих рішень на кшталт MediaPipe BlazePose, з метою визначення найбільш релевантної архітектури для специфічних задач моніторингу в обмежених корпоративних просторах. Окрім того, окремо аналізується доцільність та технічна можливість поєднання кількох різнорідних моделей у рамках ансамблевого підходу, що дозволить нівелювати

слабкі сторони окремих алгоритмів та суттєво підвищити загальну стабільність розпізнавання складних сцен.

Окреме і надзвичайно важливе місце в роботі займає дослідження проблем, пов’язаних із масштабованістю рішення, інформаційною безпекою та захистом персональних даних. У процесі розроблення системи моніторингу необхідно враховувати не лише суто технічні інженерні аспекти, а й численні етичні норми, правові регуляції та організаційні обмеження, що діють у сфері трудових відносин.

Спроєктована система повинна залишатися стабільною та відмовостійкою при лінійному або експоненційному збільшенні кількості джерел відеосигналу, одночасних користувачів і аналітичних запитів, не втрачаючи при цьому продуктивності та не допускаючи критичного перевантаження серверної інфраструктури. Питання безпеки охоплюють впровадження надійних протоколів шифрування відеопотоків, реалізацію багаторівневої системи розмежування прав доступу, ведення захищених журналів подій і регулярний аудит дій адміністраторів. Крім того, глибоко досліджується механізм автоматичної анонімізації зображень (наприклад, розмиття облич), що дозволяє зберігати аналітичну цінність зібраних даних для менеджменту без порушення приватності та розкриття особистості конкретного працівника.

Завершальним стратегічним напрямом роботи є формування пакету практичних рекомендацій щодо створення гнучких, адаптивних і масштабованих систем моніторингу, здатних ефективно функціонувати в різних сценаріях експлуатації — від невеликих офісних приміщень до великих промислових підприємств із розгалуженою мережею камер відеоспостереження. Розроблені рекомендації охоплюють обґрунтування принципів побудови архітектури, підходи до ефективного розподілу обчислювальних ресурсів між периферійними пристроями та сервером, стандарти інтеграції з існуючими корпоративними базами даних та CRM-системами, а також варіанти оптимізації «важких»

нейромережевих моделей для роботи на пристроях із обмеженими потужностями в рамках концепції граничних обчислень (edge computing).

Таким чином, представлене дослідження передбачає реалізацію комплексного науково-прикладного підходу, який гармонійно поєднує теоретичну систематизацію знань, алгоритмічне та математичне моделювання, інженерне проєктування архітектури та етап практичного тестування й валідації результатів. Підсумком цієї роботи має стати створення повнофункціональної інтелектуальної системи, що забезпечує об’єктивний, неупереджений моніторинг діяльності працівників, сприяє підвищенню загальної ефективності управління персоналом та дозволяє мінімізувати ризики й втрати, пов’язані з людським фактором.

У подальшій перспективі отримані наукові та практичні результати можуть слугувати надійним фундаментом для розроблення адаптивних аналітичних платформ нового покоління, здатних не лише фіксувати поточний стан справ, а й прогнозувати поведінкові тенденції та оптимізувати бізнес-процеси в робочому середовищі.

2 МЕТОДИ ТА МОДЕЛІ АНАЛІЗУ ВІДЕОПОТОКУ

2.1 Метод Motion Detection

У спроєктованій клієнт—серверній архітектурі метод Motion Detection відіграє критично важливу роль первинного механізму аналізу відеопотоку. Він не розглядається як основний засіб інтерпретації сцени, а використовується як початковий, обчислювально «легкий» тригер активності. Головним призначенням цього методу є оперативне відсіювання так званих «порожніх» кадрів, у яких відсутні будь-які значущі зміни. Такий підхід дозволяє суттєво оптимізувати використання обчислювальних ресурсів системи, оскільки у стані простою не відбувається залучення графічного процесора для виконання ресурсоємних алгоритмів детекції об’єктів, зокрема нейронної мережі YOLO.

Усі обчислення на даному етапі виконуються на центральному процесорі, що робить метод придатним для безперервної роботи у режимі реального часу.

Принцип роботи Motion Detection базується на аналізі змін між послідовними кадрами відеопотоку. Замість спроби розпізнати тип або клас об’єкта, алгоритм зосереджується виключно на факті наявності руху в сцені. Процес виявлення руху складається з кількох послідовних етапів, які формують єдиний конвеєр попередньої обробки даних та аналізу різниці між кадрами.

На початковому етапі система захоплює поточний відеокадр. Для зменшення впливу цифрового шуму та усунення дрібних, несуттєвих коливань яскравості зображення кадр перетворюється у відтінки сірого за допомогою функції cv2.cvtColor. Це дозволяє зменшити обсяг оброблюваної інформації, оскільки подальший аналіз виконується лише в одному каналі замість трьох. Після цього до зображення застосовується гаусівський фільтр розмиття cv2.GaussianBlur, який згладжує високочастотний шум та підвищує стійкість алгоритму до випадкових спотворень, спричинених особливостями матриці камери або компресією відеопотоку.

Оброблений поточний кадр $I_t(x, y)$ віднімається від аналогічно обробленого попереднього кадру $I_{t-1}(x, y)$. У результаті формується різницеве зображення, яке містить інформацію лише про ті пікселі, значення яких змінилися між двома послідовними моментами часу. Цей процес математично описується наступною формулою:

$$D(x, y) = |I_t(x, y) - I_{t-1}(x, y)|, \quad (2.1)$$

де $D(x, y)$ — це зображення, що містить лише пікселі, які змінилися.

На наступному етапі до зображення різниці застосовується порогова обробка за допомогою функції cv2.threshold. Метою цього кроку є відсікання незначних змін яскравості, які можуть бути спричинені мерехтінням освітлення, коливаннями експозиції або фоновим шумом. Усі пікселі, значення яких перевищують заданий поріг T , вважаються такими, що відповідають руху, тоді як решта ігноруються. У результаті формується бінарне зображення, що містить лише області потенційної активності.

Однак, отримане бінарне зображення часто містить шуми у вигляді окремих точок або розривів усередині рухомих об'єктів (наприклад, якщо одяг людини має однорідний колір, внутрішня частина може не виділитися як рух). Для усунення цих дефектів застосовується морфологічна операція дилатації (Dilation).

Суть операції полягає у "нарощуванні" світлих областей шляхом проходження структурним елементом (ядром) по зображенню. Якщо хоча б один піксель під ядром є білим, центральний піксель також стає білим. Це дозволяє об'єднати розрізнені фрагменти руху в суцільні плями (blobs), що значно покращує якість подальшого детектування контурів та гарантує, що людина буде розпізнана як єдиний об'єкт, а не сукупність окремих плям.

Подальший аналіз виконується шляхом пошуку контурів на бінаризованому зображенні з використанням функції cv2.findContours. Цей етап дозволяє об’єднати окремі пікселі руху у зв’язні області, що відповідають реальним об’єктам або їх частинам. Виявлені контури аналізуються з точки зору їх площі, що дає змогу відфільтрувати дрібні артефакти та зосередитися лише на значущих подіях у кадрі. На рис. 2.1 наведено приклад роботи методу Motion Detection, який демонструє виділення рухомих областей на відео.

Якщо площа знайдених контурів перевищує мінімально допустиме значення, клієнтський агент робить висновок про наявність значущого руху в сцені. У такому випадку система переходить до наступного, більш складного етапу аналізу, передаючи повноколірний кадр для детекції об’єктів за допомогою нейронної мережі YOLO.

Рисунок 2.1 — Приклад використання методу Motion Detection

Таким чином, Motion Detection виступає ефективним фільтром, який дозволяє зменшити кількість кадрів, що підлягають глибокому аналізу, забезпечуючи оптимальний баланс між швидкодією та точністю всієї системи.

З точки зору обчислювальної складності, запропонований алгоритм має лінійну залежність $O(N)$, де N — кількість пікселів у кадрі. Це вигідно відрізняє його від нейромережевих методів, складність яких залежить від глибини архітектури та кількості параметрів моделі. Саме така низька ресурсоємність дозволяє клієнтському агенту виконувати перевірку кожного кадру відеопотоку з мінімальною затримкою (менше 5–8 мс на кадр для роздільної здатності Full HD), активуючи "важкі" алгоритми лише у моменти реальної потреби.

2.2 Метод cvMeanShift (Метод середнього зсуву)

У каскадній системі моніторингу, що розробляється, алгоритм cvMeanShift виконує роль швидкісного агента супроводу об’єктів між послідовними кадрами відеопотоку. Основна мотивація використання цього методу полягає у необхідності зменшення кількості запусків ресурсомістких нейронних мереж. Постійний запуск повноцінної моделі детекції об’єктів із частотою відеопотоку, наприклад 25–30 кадрів за секунду, є надзвичайно неефективним з точки зору обчислювальних витрат і вимагав би використання високопродуктивних графічних процесорів для кожної камери спостереження. Такий підхід є економічно та архітектурно невиправданим для масштабованих систем відеомоніторингу.

З огляду на це, у роботі застосовується оптимізована схема обробки, за якої «важка» нейронна мережа використовується лише для початкової детекції об’єкта класу *person*. Після одноразового визначення координат об’єкта в кадрі, подальше відстеження його руху передається «легкому» класичному алгоритму cvMeanShift. Завдання цього алгоритму полягає у безперервному супроводі об’єкта з мінімальним навантаженням на центральний процесор, що дозволяє

зберігати високу частоту обробки відеопотоку. Приклад використання методу cvMeanShift наведено на рис. 2.2.

Рисунок 2.2 — Приклад використання методу cvMeanShift

Алгоритм MeanShift належить до класичних методів комп’ютерного зору та ґрунтується на ідеї пошуку локальних максимумів щільності у просторі ознак. У контексті задачі відстеження об’єктів таким простором ознак виступає колірний простір, а саме розподіл кольорів усередині області інтересу, що відповідає об’єкту. На відміну від методів, які використовують геометричні або динамічні моделі руху, MeanShift працює виключно зі статистичними характеристиками зображення, що робить його простим і водночас ефективним у відповідних умовах.

Процес роботи алгоритму починається з етапу ініціалізації, під час якого формується так званий «колірний відбиток» об’єкта. Коли система вперше виявляє об’єкт за допомогою нейронної мережі, cvMeanShift аналізує пікселі, що знаходяться всередині прямокутної області інтересу (Region of Interest, ROI). На основі цих пікселів обчислюється колірна гістограма, яка описує розподіл кольорів та їх відносну частку. Фактично така гістограма є статистичним представленням зовнішнього вигляду об’єкта та може розглядатися як його

«колірний паспорт».

Для підвищення стійкості алгоритму до змін умов освітлення обчислення гістограми виконується не у стандартному просторі BGR, а у просторі HSV (Hue, Saturation, Value). Поділ інформації про колір та яскравість дозволяє значно зменшити вплив тіней, відблисків та локальних змін освітлення, що є типовими для виробничих та офісних приміщень. Завдяки цьому трекер зберігає працездатність навіть у ситуаціях, коли загальна яскравість сцени змінюється, але колірні характеристики об’єкта залишаються відносно сталими.

Технічно процес пошуку базується на так званій "зворотній проєкції гістограми" (Histogram Back Projection). Суть цього методу полягає у трансформації вхідного відеокадру в одноканальне зображення ймовірностей, де яскравість кожного пікселя $p(x, y)$ відповідає ймовірності де яскравість кожного пікселя $P(x, y)$ відповідає ймовірності його належності до відстежуваного об’єкта.

Ця ймовірність розраховується як відношення частоти кольору пікселя у поточному кадрі до його частоти у еталонній гістограмі об'єкта.

Значення пікселя визначається шляхом проєкції його кольору на розраховану раніше гістограму-еталон. Таким чином, області зображення, що мають кольорову гаму, подібну до об’єкта, стають яскравими, а фон затемнюється. Саме це ймовірнісне зображення виступає ваговим полем для подальшого ітеративного пошуку центру мас.

Під час обробки наступних кадрів cvMeanShift не виконує повного сканування всього зображення. Аналіз обмежується локальною областю навколо попереднього положення об’єкта, що суттєво знижує обчислювальну складність. У межах цієї області обчислюється центр мас пікселів, які найбільше відповідають колірному відбитку об’єкта. Після цього вікно пошуку зсувається у напрямку нового центру, забезпечуючи оновлення положення об’єкта у кадрі.

Ітеративний процес зсуву вікна до області з максимальною щільністю ознак математично описується формулою:

$$x_{new} = \frac{\sum_i^x K\left(\frac{|x_i - x_{current}|}{h}\right)}{\sum_i^K \left(\frac{|x_i - x_{current}|}{h}\right)} \quad (2.2)$$

де x_i — координати пікселів усередині поточного вікна пошуку h;

K — ядрова функція, яка надає більшу вагу пікселям, ближчим до поточного центру $x_{current}$.

Процес повторюється кілька разів для кожного кадру, доки центр вікна не перестане зміщуватися, використання формули в коді наведено в лістингу 2.1

Лістинг 2.1 — Використання формули в коді

```
roi_hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(roi_hsv, np.array((0., 60., 32.)),
np.array((180., 255., 255.)))
roi_hist = cv2.calcHist([roi_hsv], [0], mask, [180], [0, 180])
cv2.normalize(roi_hist, roi_hist, 0, 255,
cv2.NORM_MINMAX)
```

У наведеному фрагменті коду функція cv2.calcHist виконує побудову гістограми виключно за каналом Hue (індекс [0]), ігноруючи насиченість та яскравість. Це є свідомим архітектурним рішенням, спрямованим на підвищення робастності трекера: навіть якщо освітлення об'єкта зміниться (зміниться Value) або камера змінить баланс білого (зміниться Saturation), відтінок об'єкта залишиться найбільш стабільною ознакою.

Функція cv2.normalize приводить значення бінів гістограми до діапазону [0, 255]. Ця операція є обов'язковою для коректної роботи алгоритму зворотної проєкції в OpenCV, оскільки він очікує, що ваги пікселів будуть представлені у 8-бітному форматі цілих чисел, сумісному зі стандартними форматами зображень.

До беззаперечних переваг методу cvMeanShift належить його висока швидкодія та обчислювальна ефективність. У порівнянні з глибокими нейронними мережами цей алгоритм є значно менш ресурсоємним, що дозволяє виконувати його на центральному процесорі без використання спеціалізованих апаратних прискорювачів.

Крім того, статистичний характер аналізу забезпечує відносну стійкість до цифрового шуму та незначних деформацій об’єкта під час руху.

Водночас, практичне застосування cvMeanShift виявляє низку суттєвих обмежень. Ключовим з них є повна залежність алгоритму від колірної інформації. У випадках низького контрасту між об’єктом і фоном, наприклад, коли працівник у синій уніформі знаходиться на тлі стіни подібного кольору, трекер може втратити об’єкт або помилково зміститися на фонову область. Аналогічні проблеми виникають і при різкій зміні зовнішнього вигляду людини, зокрема при додаванні або знятті елементів одягу, що суттєво змінюють колірний розподіл.

Найбільш критичним архітектурним недоліком базового алгоритму cvMeanShift є фіксований розмір вікна відстеження, який визначається лише на етапі ініціалізації та не змінюється під час роботи. За умови наближення об’єкта до камери його проєкція на зображенні збільшується, однак вікно трекера залишається сталим і починає охоплювати лише окремі фрагменти об’єкта.

Це поступово призводить до деградації якості супроводу і, зрештою, до його зриву. Протилежна ситуація спостерігається при віддаленні об’єкта, коли рамка стає надмірно великою та захоплює значну частину фону, унаслідок чого алгоритм втрачає здатність коректно відрізняти ціль від навколишнього середовища.

2.3 Метод cvCamShift (Безперервно—адаптивний середній зсув)

Аналіз практичного застосування алгоритму cvMeanShift у задачах

трекінгу об’єктів показує, що його ключові обмеження пов’язані не зі швидкодією або складністю реалізації, а з відсутністю механізмів адаптації до змін геометричних параметрів об’єкта у відеопотоці. За умов реальної експлуатації систем відеомоніторингу об’єкти, зокрема люди, рідко залишаються на фіксованій відстані від камери. Вони постійно переміщуються у просторі сцени, що призводить до зміни масштабу їх зображення, а також до варіацій орієнтації та проєкції на площину кадру. У таких умовах використання фіксованого вікна супроводу, характерного для класичного cvMeanShift, неминуче призводить до поступового зниження точності трекінгу та втрати об’єкта.

З огляду на зазначені недоліки, у межах даного проєкту було обрано алгоритм cvCamShift (Continuously Adaptive Mean Shift), який є логічним розвитком і розширенням методу середнього зсуву. Основною ідеєю cvCamShift є поєднання статистичного підходу до аналізу кольорових ознак із динамічною адаптацією геометричних параметрів області відстеження. У загальній архітектурі клієнтського агента цей алгоритм використовується як основний «легкий» компонент трекінгу, що забезпечує стабільний супровід об’єктів між моментами повторної детекції за допомогою нейронної мережі.

На початковому етапі роботи cvCamShift повністю наслідує принципи функціонування cvMeanShift. Після первинного виявлення об’єкта «важким» детектором YOLO система отримує координати області інтересу (Region of Interest, ROI), яка містить проєкцію об’єкта на зображенні. На основі пікселів цієї області формується колірна гістограма у просторі HSV, що використовується як статистичний опис зовнішнього вигляду об’єкта. Використання простору HSV дозволяє зменшити вплив змін освітлення та підвищити стабільність супроводу у сценах із неоднорідним або динамічним світловим середовищем.

Під час обробки наступних кадрів cvCamShift, як і MeanShift, визначає нове положення об’єкта шляхом пошуку центру мас пікселів, що найбільше

відповідають сформованому колірному відбитку. Однак на відміну від базового алгоритму, на цьому етапі процес не завершується. Після знаходження нового центру алгоритм переходить до детальнішого аналізу просторового розподілу кольорових ознак усередині області відстеження.

Ключовою особливістю cvCamShift є використання моментів зображення (Image Moments), які дозволяють оцінити геометричні характеристики області, що відповідає об’єкту. Моменти зображення є узагальненими числовими характеристиками, які описують не лише положення області у кадрі, але й її площу, форму та орієнтацію.

Зокрема, нульовий момент (M_{00}) характеризує сумарну «масу» області, тоді як перші моменти (M_{10}, M_{01}) використовуються для визначення координат центроїда. Математично моменти зображення та положення центру області описуються співвідношеннями:

$$M_{pq} = \sum_{x}^{\sum_{y}^{x^p}} y^q \, I(x, y). \tag{2.3}$$

$$\chi_{center} = \frac{M_{10}}{M_{00}}. \tag{2.4}$$

$$y_{center} = \frac{M_{01}}{M_{00}}. \tag{2.5}$$

На основі обчислених моментів алгоритм cvCamShift динамічно коригує параметри вікна відстеження, зокрема його ширину, висоту та орієнтацію. Це дозволяє області трекінгу змінювати свої розміри пропорційно до реального масштабу об’єкта у кадрі. У результаті рамка супроводу більш точно відповідає поточній проєкції об’єкта, що суттєво знижує ймовірність включення фонових пікселів до області аналізу.

Важливою перевагою такого підходу є підвищення стійкості трекінгу у випадках часткового перекриття об’єкта або зміни його пози. Завдяки адаптивній формі вікна cvCamShift зберігає коректне положення області відстеження навіть за складних траєкторій руху. Реалізація цього механізму в програмному коді наведена в лістингу 2.2.

Лістинг 2.2 — Використання формули в коді

```
ret, track_window = cv2.CamShift(back_proj,
track_window,(cv2.TERM_CRITERIA_EPS |
cv2.TERM_CRITERIA_COUNT, 10, 1))
```

На рис. 2.3 показано приклад застосування алгоритму cvCamShift у процесі супроводу об’єкта. Візуально можна спостерігати, як область відстеження автоматично змінює свої розміри при наближенні об’єкта до камери та зменшується при його віддаленні, зберігаючи коректне позиціонування. З точки зору архітектури системи відеомоніторингу, використання cvCamShift дозволяє досягти суттєвого компромісу між точністю та обчислювальною ефективністю.

Алгоритм залишається достатньо «легким» для виконання на центральному процесорі, водночас демонструючи значно вищу стабільність супроводу порівняно з класичним cvMeanShift. Алгоритм залишається достатньо «легким» для виконання на центральному процесорі, водночас демонструючи значно вищу стабільність супроводу порівняно з класичним cvMeanShift.

Завдяки зменшенню кількості втрат трекінгу система значно рідше змушена ініціювати повторну детекцію об’єкта за допомогою нейронної мережі YOLO, що позитивно впливає на загальну продуктивність, затримки обробки та масштабованість розробленого рішення.

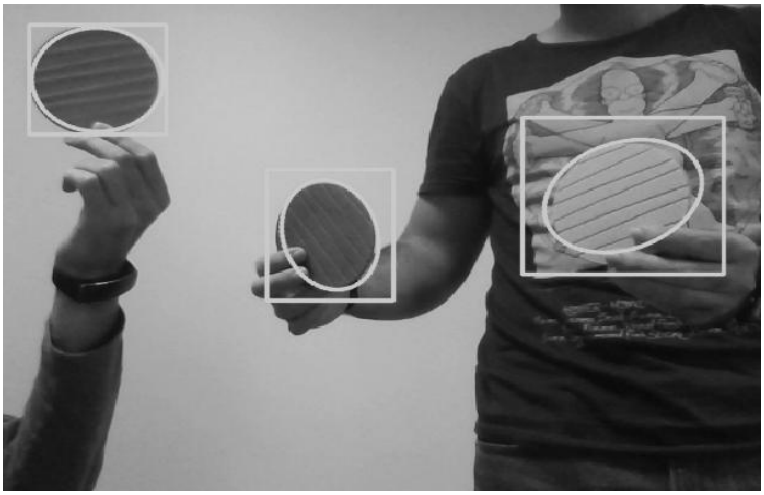

Рисунок — 2.3 Приклад використання методу cvCamShift

2.4 Метод YOLO

Метод YOLO (You Only Look Once) належить до класу сучасних нейромережевих підходів для задач виявлення та розпізнавання об’єктів на зображеннях і у відеопотоці та є одним із найрезультативніших рішень у цій галузі. Його концепція ґрунтується на принципово іншому підході до детекції об’єктів порівняно з традиційними двоетапними моделями, такими як R-CNN та його похідні. Замість послідовного виконання етапів генерації кандидатних областей і їх подальшої класифікації, YOLO здійснює повну детекцію за один прохід через нейронну мережу. Це дозволяє значно зменшити затримку обробки та забезпечити аналіз відеоданих у режимі реального часу.

У процесі роботи YOLO розглядає задачу детекції як задачу регресії. Вхідне зображення ділиться на сітку рівних клітинок, кожна з яких відповідає за прогнозування об’єктів, центр яких потрапляє в межі відповідної клітинки. Для кожної такої області нейронна мережа визначає координати обмежувальної рамки об’єкта (bounding box), коефіцієнт впевненості, що відображає ймовірність наявності об’єкта в межах рамки, а також імовірності належності до певних класів. Такий підхід дозволяє виконувати одночасну локалізацію та

класифікацію об’єктів у межах єдиного обчислювального процесу.

Навчання мережі YOLO здійснюється шляхом мінімізації спеціальної функції втрат, яка враховує помилки локалізації, класифікації та оцінки впевненості. Формально ця функція втрат буде подана у вигляді виразу (2.6).

$$L_{OSS} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \widehat{x_i})^2 + (y_i - \widehat{y_i})^2] \quad (2.6)$$

де xi, yi — координати центра передбаченої рамки;

$\widehat{x_i}, \widehat{y_i}$ — відповідні еталонні значення;

$\lambda coord$ коефіцієнт ваги;

Така формалізація дозволяє мережі одночасно оптимізувати точність визначення положення об’єкта та стабільність детекції.

Після отримання первинних результатів детекції застосовується процедура пригнічення немаксимумів (Non-Maximum Suppression, NMS). Її призначення полягає в усуненні дубльованих або надмірно перекривних рамок, які відповідають одному й тому ж об’єкту. У результаті залишається лише одна, найбільш імовірна рамка для кожного об’єкта, що підвищує наочність та точність кінцевого результату. Приклад використання цього підходу в програмній реалізації наведено в лістингу 2.3.

Лістинг 2.3 — використання формули в коді

```
for detection in output:
scores = detection[5:]
class_id = np.argmax(scores)
confidence = scores[class_id]
if class_id == 0 and confidence > conf_threshold:
box = detection[0:4] * np.array([width, height
, width, height])
(centerX, centerY, boxW, boxH) =
box.astype("int")
```

Архітектурно YOLO базується на згортковій нейронній мережі (Convolutional Neural Network, CNN), яка автоматично виділяє суттєві візуальні ознаки зображення, починаючи від простих країв і текстур до складних семантичних патернів. Для навчання таких моделей використовуються великі набори розмічених зображень, що дозволяє досягати високої точності навіть у складних сценах із великою кількістю об’єктів та фонових деталей. На рис. 2.4 наведено приклад використання методу YOLO для детекції об’єктів у відеопотоці.

Однією з ключових переваг YOLO є висока швидкість обробки, яка дозволяє аналізувати десятки кадрів за секунду навіть на обладнанні середнього класу. Саме ця характеристика зумовила широке застосування алгоритму в системах відеоспостереження, автономного транспорту, робототехніки, медичних системах аналізу зображень та рішеннях для розумних міст. Крім того, YOLO відзначається гнучкістю, оскільки здатний працювати з об’єктами різних категорій і коректно обробляти різні зображення з різними просторовими розмірами.

Водночас, незважаючи на численні переваги, алгоритм YOLO має і певні обмеження. Зокрема, його точність зменшується при роботі з дрібними або щільно розташованими об’єктами, оскільки поділ зображення на сітку не завжди дозволяє коректно локалізувати всі деталі сцени. Крім того, етап навчання моделі потребує значних обчислювальних ресурсів і, як правило, вимагає використання потужних графічних процесорів.

У сценах із великою кількістю подібних об’єктів також можливі помилки ідентифікації, пов’язані з перекриттям рамок і неоднозначністю класифікації. Розвиток сімейства YOLO призвів до появи вдосконалених версій, зокрема YOLOv4 та YOLOv5, які спрямовані на зменшення зазначених недоліків.

У цих моделях використовуються оптимізовані архітектури, такі як CSPDarknet, а також покращені методи післяобробки результатів детекції.

Рисунок 2.4 — Приклад використання методу YOLO

Це дозволяє підвищити точність, стабільність та ефективність роботи алгоритму в реальних умовах експлуатації, що робить YOLO доцільним вибором для інтеграції у складні каскадні системи аналізу відеопотоку.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Методологія проєктування та реалізації системи моніторингу

Проєктування автоматизованої системи моніторингу діяльності працівників ґрунтується на поєднанні методів комп'ютерного зору, штучного інтелекту та інженерних підходів до розподіленої обробки даних. Особливістю даного завдання є необхідність забезпечення аналізу відеопотоку в реальному часі за умов обмежених обчислювальних ресурсів, що характерно для виробничих і офісних середовищ. У зв'язку з цим методологія розробки системи орієнтована не лише на досягнення високої точності розпізнавання, але й на оптимізацію швидкодії, стабільності та масштабованості програмного рішення.

В основі проєктування лежить принцип каскадної обробки даних, за яким складні та ресурсоємні методи аналізу застосовуються лише у випадках, коли це обґрунтовано з точки зору інформаційної цінності кадру. Такий підхід дозволяє значно зменшити середнє навантаження на систему без втрати функціональних можливостей моніторингу.

Першим етапом проєктування є формування вимог до автоматизованої системи моніторингу, які визначають логіку вибору методів і програмних засобів. Система повинна забезпечувати безперервний аналіз відеопотоку з камер спостереження та виявляти присутність працівників у контрольованих зонах з мінімальною затримкою. При цьому важливою вимогою є можливість функціонування системи в умовах змінного освітлення, часткових перекриттів об'єктів, а також фонових рухів, що не мають відношення до діяльності персоналу.

Окрему увагу приділено вимогам до продуктивності. Оскільки система орієнтована на тривалу безперервну роботу, вона не повинна перевантажувати центральний процесор або графічний адаптер, особливо у періоди відсутності

активності в кадрі. Це обмеження безпосередньо вплинуло на вибір методів аналізу відеопотоку та обумовило відмову від постійного використання нейромережевих моделей для кожного кадру.

Крім того, система повинна мати гнучкі засоби налаштування, які дозволяють адаптувати її до конкретних умов експлуатації без внесення змін у програмний код. До таких параметрів належать джерело відеопотоку, порогові значення впевненості детекції, геометрія заборонених зон та адреси серверних сервісів. Це забезпечує можливість масштабування системи та її використання на різних об’єктах спостереження.

3.1.1 Метод каскадної обробки відеопотоку як основа зменшення обчислювального навантаження

Для виконання завдання моніторингу в реальному часі в роботі застосовано метод каскадної обробки відеопотоку. Його сутність полягає у послідовному використанні декількох методів аналізу з різною обчислювальною складністю, де кожен наступний етап активується лише за наявності відповідних умов. Математично ефективність запропонованого каскадного методу можна описати через загальний час обробки одного кадру T_{frame}, який залежить від стану сцени. Ця залежність виражається формулою (3.1):

$$\mathrm{T_{frame}} = \mathrm{T_{MD}} + \alpha(\mathrm{T_{YOLO}} + \mathrm{T_{Track}}) \quad (3.1)$$

де T_{MD} — час виконання алгоритму детекції руху (Motion Detection), який є постійною величиною для кожного кадру;

T_{YOLO} — час інференсу нейронної мережі;

T_{Track} — час роботи алгоритму трекінгу (cvCamShift);

α — бінарний індикатор стану сцени (тригер).

На першому рівні каскаду використовується метод виявлення руху (Motion

Detection), який виконує роль первинного фільтра.

Цей метод не здійснює семантичного аналізу сцени, а лише визначає факт наявності значущих змін між послідовними кадрами. Завдяки своїй простоті він може виконуватися на центральному процесорі з мінімальними витратами ресурсів і дозволяє відсіяти переважну більшість кадрів, що не містять корисної інформації. Фрагмент програмної реалізації методу виявлення руху, який використовується як первинний тригер каскадної обробки відеопотоку, наведено в лістингу 3.1.

Лістинг 3.1 — Реалізація методу Motion Detection у клієнтському агенті

```
def detect_motion(self, frame):
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (21, 21), 0)
if self.prev_frame is None:
self.prev_frame = gray
return False
frame_delta = cv2.absdiff(self.prev_frame, gray)
thresh = cv2.threshold(frame_delta, 25, 255, cv2.THRESH_BINARY)[1]
thresh = cv2.dilate(thresh, None, iterations=2)
contours, _ = cv2.findContours(
thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE  )
self.prev_frame = gray
return any(cv2.contourArea(c) > 500 for c in contours)
```

Другий рівень каскаду активується лише у випадку виявлення руху та передбачає застосування нейромережевого методу детекції об’єктів. На цьому етапі використовується модель YOLO, яка забезпечує надійну ідентифікацію людини в кадрі. Важливою особливістю є те, що нейромережа не запускається безперервно, а лише за умов, коли це виправдано з точки зору результатів первинного аналізу.

Після ідентифікації об’єкта система переходить до третього рівня каскаду, де застосовується метод трекінгу на основі алгоритму cvCamShift. Цей метод дозволяє супроводжувати об’єкт у послідовних кадрах без повторного залучення

нейронної мережі, що істотно зменшує загальне обчислювальне навантаження. Таким чином, каскадний метод поєднує високу точність глибинного навчання з обчислювальною ефективністю класичних алгоритмів комп’ютерного зору.

3.1.2 Обґрунтування вибору методів детекції та трекінгу об’єктів

Вибір конкретних методів детекції та трекінгу об’єктів зумовлений вимогами до швидкодії та стабільності системи. Для детекції об’єктів класу *person* обрано архітектуру YOLO, яка належить до однопрохідних нейромережевих моделей, код для ініціалізації та запуску моделі наведено в лістингу 3.2.

Лістинг 3.2 — Запуск нейромережевої детекції YOLO

```
blob = cv2.dnn.blobFromImage(
frame, 0.00392, (416, 416),
(0, 0, 0), swapRB=True, crop=False
)
self.net.setInput(blob)
outputs = self.net.forward(self.output_layers)
```

На відміну від двоетапних підходів, такі моделі забезпечують суттєво меншу затримку обробки, що є критично важливим для систем відеомоніторингу.

Разом із тим постійне використання YOLO для кожного кадру призвело б до надмірного навантаження на систему. З цієї причини в роботі застосовано метод cvCamShift як засіб супроводу об’єктів між етапами повторної детекції. Даний алгоритм ґрунтується на аналізі колірної гістограми області інтересу та має механізм адаптації розмірів вікна відстеження, що дозволяє враховувати зміну масштабу об’єкта у кадрі.

Поєднання нейромережевого методу детекції з класичним методом трекінгу дозволяє досягти компромісу між точністю та продуктивністю. YOLO

забезпечує надійну ідентифікацію об’єкта, тоді як cvCamShift підтримує стабільний супровід з мінімальними витратами ресурсів. Такий підхід є доцільним для тривалого моніторингу в реальних умовах експлуатації.

3.2 Проєктування архітектури автоматизованої системи моніторингу

Проєктування архітектури моніторингу є ключовим етапом реалізації обраних методів аналізу відеопотоку. На цьому етапі визначається, яким чином методи комп’ютерного зору та штучного інтелекту інтегруються у програмну структуру, а також які програмні та апаратні засоби забезпечують їх стабільне функціонування в реальних умовах експлуатації. Основною метою архітектурного проєктування є створення такої структури системи, яка дозволяє ефективно обробляти відеодані в реальному часі, масштабувати рішення та мінімізувати вплив збоїв окремих компонентів на загальну працездатність.

З урахуванням вимог до продуктивності та масштабованості в роботі обрано клієнт—серверний архітектурний підхід. Його сутність полягає у розподілі функцій аналізу та збереження даних між автономними клієнтськими модулями та центральним сервером. Такий підхід дозволяє реалізувати принцип розподіленої обробки, за яким ресурсоємні методи аналізу відеопотоку виконуються безпосередньо поблизу джерела даних, а серверна частина зосереджується на агрегації результатів, їх збереженні та наданні доступу до них.

Клієнтський компонент системи проєктується як автономний програмний агент, який взаємодіє з конкретним джерелом відеопотоку. Саме на цьому рівні реалізуються основні методи аналізу: виявлення руху, нейромережеву детекцію об’єктів та трекінг. Такий підхід відповідає концепції edge computing, за якої попередня обробка та інтелектуальний аналіз даних виконуються на периферійних вузлах системи. Це дозволяє суттєво зменшити обсяг передаваних по мережі даних, оскільки на сервер надсилаються не самі відеопотоки, а лише структуровані результати аналізу у вигляді подій та медіафрагментів. Формально

загальний час реакції системи $T_{response}$ у запропонованій розподіленій архітектурі визначається як сума затримок на кожному з етапів обробки та передачі даних:

$$T_{(response)} = T_{(edge)} + T_{(net)} + T_{(server)} \quad (3.2)$$

де T_{edge} — час обробки відеокадру та формування події на клієнтському агенті (залежить від швидкодії локального алгоритму);

T_{net} — час передачі сформованого пакету даних (JSON + зображення) через мережу, який залежить від пропускної здатності каналу;

T_{server} — час обробки запиту сервером (валідація, запис у базу даних та збереження файлів).

Мінімізація складової T_{net} досягається саме завдяки передачі лише метаданих інциденту, а не суцільного відеопотоку.

Серверна частина системи, у свою чергу, виконує роль централізованого засобу збереження та управління результатами моніторингу. Вона не залучається до обчислювально складних операцій комп'ютерного зору, що дозволяє знизити вимоги до апаратного забезпечення серверу та підвищити надійність роботи всієї системи. Сервер забезпечує прийом подій від клієнтських агентів, їх валідацію, збереження у базі даних, а також надання доступу до результатів через програмний інтерфейс прикладного програмування.

Важливою особливістю спроєктованої архітектури є її масштабованість. Додавання нових камер або зон моніторингу не потребує внесення змін до серверної логіки, а зводиться до розгортання додаткових клієнтських агентів із відповідною конфігурацією. Таким чином, система може бути адаптована як для невеликих об'єктів спостереження, так і для розгалужених інфраструктур з десятками або сотнями джерел відеоданих.

Окрім масштабованості, клієнт–серверний підхід підвищує

відмовостійкість системи. У разі виходу з ладу окремого клієнтського модуля або камери, інші компоненти продовжують функціонувати у штатному режимі. Це є особливо важливим для систем моніторингу, які мають працювати безперервно та не допускати втрати контролю над критичними зонами.

Таким чином, спроєктована архітектура виступає ефективним засобом практичної реалізації обраних методів аналізу відеопотоку. Вона забезпечує баланс між обчислювальною ефективністю, гнучкістю налаштування та надійністю роботи системи, що є необхідною умовою для її використання у реальних виробничих та офісних середовищах.

3.3 Розробка клієнтського програмного модуля автоматизованої системи моніторингу

Клієнтський програмний модуль є ключовим виконавчим елементом автоматизованої системи моніторингу, оскільки саме на цьому рівні реалізуються основні методи аналізу відеопотоку та прийняття первинних рішень. Під час розробки даного модуля основний акцент зроблено на поєднанні точності розпізнавання з обчислювальною ефективністю, що є критично важливим для тривалої безперервної роботи системи в режимі реального часу.

Клієнтський модуль проєктується як автономний агент, що функціонує незалежно від серверної частини та безпосередньо взаємодіє з джерелом відеоданих. Такий підхід дозволяє реалізувати розподілену модель обробки, за якої кожен агент аналізує лише один відеопотік, використовуючи локальні обчислювальні ресурси. У результаті зменшується навантаження на центральну інфраструктуру, а система загалом стає більш масштабованою та стійкою до збоїв.

3.3.1 Функціональне призначення та логіка роботи клієнтського агента

Функціональне призначення клієнтського агента полягає у безперервному

аналізі відеопотоку з камери спостереження, виявленні присутності працівників у контрольованих зонах та формуванні подій у разі порушення заданих умов. При цьому агент не передає на сервер необроблені відеодані, а працює з ними локально, формуючи лише узагальнені результати аналізу.

Логіка роботи агента базується на циклічній обробці кадрів відеопотоку. Для кожного кадру система визначає поточний стан сцени та вирішує, які методи аналізу доцільно застосувати на даному етапі. Такий підхід дозволяє адаптивно змінювати інтенсивність обчислень залежно від ситуації в кадрі, що є важливою умовою ефективного використання ресурсів.

Особливістю реалізації є наявність внутрішнього механізму контролю стану агента, який відображає активність різних модулів аналізу. Це дозволяє не лише оптимізувати роботу системи, але й здійснювати оцінку продуктивності під час експлуатації, що має значення для подальшого вдосконалення алгоритмів.

3.3.2 Метод реалізації первинної обробки відеопотоку

Первинна обробка відеопотоку є обов’язковим етапом аналізу, оскільки безпосередня робота з “сирими” кадрами значно знижує стійкість і точність подальших методів. У межах клієнтського модуля цей етап реалізовано як окремий метод, що виконується для кожного кадру до запуску складніших алгоритмів.

Основним завданням первинної обробки є зменшення впливу шумів, коливань освітлення та дрібних змін у сцені, які не мають відношення до реальної активності працівників. Для цього використовується перетворення зображення у відтінки сірого та фільтрація за допомогою гаусівського розмиття. Таке попереднє згладжування дозволяє стабілізувати результати подальшого аналізу різниці між кадрами.

На основі попередньо оброблених кадрів реалізується метод виявлення руху, який виконує роль первинного тригера активності. Метод полягає у

порівнянні поточного кадру з попереднім та визначенні областей, у яких відбулися значущі зміни яскравості. Якщо такі зміни відсутні або не перевищують заданий поріг, система вважає сцену статичною та не запускає ресурсоємні методи аналізу.

Застосування цього методу дозволяє суттєво зменшити кількість кадрів, що передаються на подальші етапи обробки. У практичних умовах це означає, що нейромережеві моделі активуються лише в моменти реальної активності, а не безперервно, що позитивно впливає на продуктивність і стабільність роботи клієнтського агента.

3.3.3 Метод інтеграції нейромережевої детекції об’єктів

Після спрацювання первинного тригера система переходить до етапу семантичного аналізу сцени. На цьому етапі реалізується метод нейромережевої детекції об’єктів, метою якого є ідентифікація людини у відеокадрі. Для цього використовується згорткова нейронна мережа типу YOLO, яка дозволяє виконувати детекцію в режимі реального часу.

Інтеграція нейромережевого методу у клієнтський агент здійснюється таким чином, щоб мінімізувати кількість запусків моделі. Замість постійної обробки кожного кадру нейромережа активується лише після виявлення руху, що є обґрунтованим з точки зору інформаційної цінності кадру. Формально процес формування множини виявлених об'єктів $O_{detected}$ здійснюється шляхом фільтрації вихідних даних нейромережі за класом об'єкта та порогом впевненості згідно з виразом (3.2):

$$O_{detected} = \{b_i \mid P(class_i = person) \geq \lambda_{conf}\} \quad (3.2)$$

де $b_i - i$ та обмежувальна рамка (bounding box), яку прогнозує мережа;

P — ймовірність приналежності об'єкта до класу «людина» (confidence score);

λ_{conf} — заданий поріг впевненості (у даній системі встановлено на рівні 0.5–0.6), що дозволяє відсіяти помилкові спрацювання низької якості.

Такий підхід дозволяє поєднати високу точність глибинного навчання з вимогами до швидкодії системи.

Результатом роботи нейромережі є координати обмежувальної рамки об’єкта та рівень впевненості детекції. Отримані дані використовуються не лише для фіксації факту присутності людини, але й для ініціалізації подальшого трекінгу. Таким чином, нейромережевий метод виступає засобом первинної ідентифікації, після чого система переходить до більш легких алгоритмів супроводу.

3.3.4 Засоби реалізації трекінгу та контролю заборонених зон

Для супроводу виявленого об’єкта у послідовних кадрах у клієнтському модулі реалізовано метод трекінгу на основі алгоритму cvCamShift. Даний алгоритм використовує статистичні характеристики кольорового розподілу об’єкта та дозволяє адаптивно змінювати розміри області відстеження залежно від положення людини відносно камери.

Для контролю якості супроводу та прийняття рішення про необхідність перезапуску трекера використовується метрика перекриття (IoU — Intersection over Union) між поточною рамкою трекера B_{track} та новою детекцією нейромережі B_{yolo}:

$$IoU = \frac{\text{Area}\left(B_{track} \cap B_{yolo}\right)}{\text{Area}\left(B_{track} \cup B_{yolo}\right)}$$

Якщо значення метрики падає нижче критичного рівня λ_{reset} (наприклад, 0.3), система вважає, що трекер втратив об'єкт, і виконує реініціалізацію координат:

$$\text{If } (IoU < \lambda_{reset}), \text{ then } B_{track} \leftarrow B_{yolo}$$

Це дозволяє автоматично коригувати помилки трекінгу, що виникають при швидкому русі об'єкта або зміні освітлення.

Застосування cvCamShift як засобу трекінгу дозволяє уникнути повторного запуску нейромережі на кожному кадрі приклад використання методу наведено в лістингу 3.3.

Лістинг 3.3 — Реалізація трекінгу методом cvCamShift

```
dst = cv2.calcBackProject(
[hsv], [0], self.roi_hist, [0, 180], 1
)
ret, self.track_window = cv2.CamShift(
dst, self.track_window, self.term_crit
)
```

Це значно зменшує обчислювальне навантаження та забезпечує плавний супровід об’єкта навіть за умов часткових перекриттів або змін масштабу. Повторна детекція за допомогою YOLO виконується лише у випадках втрати трекінгу або через задані інтервали часу для верифікації результатів.

Контроль заборонених зон реалізується шляхом геометричного аналізу положення об’єкта у кадрі. Центр обмежувальної області, що відповідає людині, перевіряється на належність до заданих полігонів, які описують небезпечні або обмежені зони. Для визначення факту порушення система обчислює геометричний центр об'єкта (C_x, C_y) на основі координат його обмежувальної рамки (x, y, w, h):

$$C_x = x + \frac{w}{2}, \quad C_y = y + \frac{h}{2} \tag{3.3}$$

Ця перевірка виконується алгоритмічно за допомогою методу трасування променів (функція pointPolygonTest у бібліотеці OpenCV), що забезпечує точну верифікацію навіть для зон складної форми.

Такий підхід є універсальним і дозволяє задавати зони довільної форми без прив'язки до конкретної сцени.

3.3.5 Метод формування подій та взаємодії з серверною частиною

У разі фіксації порушення клієнтський агент формує подію, яка є узагальненим результатом аналізу відеопотоку. Метод формування подій передбачає збереження доказової інформації та структурування метаданих, необхідних для подальшого аналізу на сервері.

Подія містить часову мітку, ідентифікатор агента, тип зафіксованої ситуації та інформацію про зону порушення. Такий формат дозволяє однозначно ідентифікувати інцидент та забезпечує сумісність із серверними засобами збереження та візуалізації. Передача подій здійснюється через захищений програмний інтерфейс, що забезпечує цілісність і надійність даних під час транспортування.

Реалізація цього методу завершує цикл роботи клієнтського агента, перетворюючи необроблений відеопотік у структуровану інформацію, придатну для подальшого зберігання, аналізу та прийняття управлінських рішень.

3.4 Проєктування та реалізація серверної частини автоматизованої системи моніторингу.

Серверна частина автоматизованої системи моніторингу виступає

центральним вузлом збору, збереження та узагальнення результатів аналізу, що надходять від клієнтських агентів. На відміну від клієнтського модуля, сервер не виконує безпосередньої обробки відеопотоків і не залучається до ресурсомістких алгоритмів комп'ютерного зору. Його основне призначення полягає у забезпеченні надійного прийому подій, їх коректної обробки, довготривалого збереження та надання доступу до накопичених даних для подальшого аналізу.

Проєктування серверної частини здійснювалося з урахуванням вимог до масштабованості, надійності та безпеки. Оскільки система орієнтована на роботу з потенційно великою кількістю клієнтських агентів, серверна архітектура повинна забезпечувати стабільну роботу навіть у випадку одночасного надходження значної кількості подій. З цієї причини обрано підхід, за яким сервер виконує лише функції керування даними та не перевантажується додатковими обчисленнями.

Для програмної реалізації серверної частини обрано високорівневий вебфреймворк FastAPI, який функціонує на мові програмування Python. Цей вибір зумовлений необхідністю забезпечення високої швидкодії обробки HTTP-запитів, що досягається завдяки асинхронній архітектурі (ASGI) та використанню сервера Uvicorn.

На відміну від традиційних синхронних фреймворків, FastAPI дозволяє серверу обробляти сотні з'єднань одночасно без блокування потоків виконання, що є критично важливим для системи моніторингу, де безліч агентів можуть одночасно надсилати повідомлення про інциденти.

Структурно серверний додаток побудовано за принципом інверсії залежностей (Dependency Injection), що спрощує керування підключеннями до бази даних та конфігураціями. Основна точка входу в програму ініціалізує об'єкт додатку app, налаштовує маршрутизацію запитів та створює необхідні директорії для фізичного збереження медіафайлів (зображень та відеофрагментів), які надходять від клієнтських агентів.

Основним методом взаємодії між клієнтськими агентами та сервером є використання програмного інтерфейсу прикладного програмування. Такий підхід дозволяє чітко формалізувати правила обміну даними та забезпечити незалежність реалізації клієнтської і серверної частин. Кожна подія, сформована клієнтським агентом, передається на сервер у стандартизованому форматі, що містить часові, ідентифікаційні та семантичні характеристики зафіксованого інциденту.

Важливим аспектом серверної реалізації є метод валідації отриманих даних. Перед збереженням інформації у базі даних сервер перевіряє коректність структури повідомлення, наявність усіх необхідних полів та відповідність їх форматів очікуваним значенням. Такий підхід дозволяє запобігти накопиченню помилкових або неповних записів, що особливо важливо для систем, результати яких можуть використовуватися для прийняття управлінських рішень або аналізу безпеки.

Для збереження результатів моніторингу використовується реляційна база даних, яка забезпечує структуроване зберігання інформації про інциденти. У базі даних фіксуються ключові параметри події, зокрема час її виникнення, ідентифікатор клієнтського агента, тип порушення та пов’язані з ним медіафайли. Така структура дозволяє виконувати подальший пошук, фільтрацію та агрегування даних без необхідності обробки великих обсягів відеоінформації.

Взаємодія з реляційною базою даних реалізована за допомогою технології ORM (Object-Relational Mapping) на базі бібліотеки SQLAlchemy. Це дозволило абстрагуватися від написання "сирих" SQL-запитів та працювати з записами бази даних як з об'єктами Python. У системі визначено клас Incident, який успадковується від декларативної бази Base та відображається на таблицю incidents.

Структура таблиці включає первинний ключ id (Integer), поля для

зберігання метаданих події (timestamp_utc, agent_id, event_type, zone_name), а також текстові поля snapshot_path та video_path, що містять шляхи до збережених файлів на дисковому масиві. Для підключення до бази даних використовується драйвер, сумісний зі стандартом SQL (на етапі розробки — SQLite, у продакшн-середовищі — PostgreSQL), що налаштовується через рядок підключення DATABASE_URL.

Керування сесіями бази даних здійснюється через механізм sessionmaker, який забезпечує атомарність транзакцій: запис про інцидент фіксується (commit) лише після успішного збереження всіх пов'язаних даних.

Медіафайли, що містять візуальні докази подій, зберігаються окремо від основної бази даних у файловому сховищі. Це рішення обумовлене як міркуваннями продуктивності, так і зручністю подальшого масштабування. Відокремлення великих файлів від табличних даних дозволяє зменшити навантаження на базу даних та спростити резервне копіювання інформації.

Серверна частина також виконує роль засобу інтеграції з користувацькими інтерфейсами та зовнішніми аналітичними системами. Через відповідні програмні інтерфейси сервер надає доступ до збережених даних, що дозволяє реалізувати вебінтерфейси для адміністраторів, служби безпеки або керівництва підприємства. Завдяки цьому результати автоматизованого моніторингу можуть бути представлені у вигляді журналів подій, статистичних зведень або аналітичних звітів.

Загалом серверна частина системи виступає універсальним засобом централізації та узагальнення результатів, отриманих у процесі аналізу відеопотоку. Вона забезпечує логічне завершення каскадного методу обробки, перетворюючи локальні результати роботи клієнтських агентів у цілісну інформаційну картину, придатну для подальшого використання в межах організації.

3.5 Оцінка ефективності та стабільності роботи автоматизованої системи моніторингу.

Завершальним етапом проєктування та розробки автоматизованої системи моніторингу є оцінка ефективності її роботи в умовах, наближених до реальної експлуатації. Оскільки система призначена для безперервного аналізу відеопотоку в реальному часі, ключовими критеріями її працездатності є швидкодія, стабільність функціонування та раціональне використання обчислювальних ресурсів. Саме ці характеристики визначають можливість практичного впровадження розробленого рішення у виробничих або офісних середовищах.

З метою систематизації методів, застосованих у клієнтському агенті, та визначення їх ролі у загальній логіці роботи системи, в таблиці 3.1 наведено узагальнену характеристику основних алгоритмів аналізу відеопотоку.

Таблиця 3.1 — Показники ефективності роботи клієнтського агента

Показник	Опис	Характерне значення
Середній час обробки кадру	Повний цикл обробки без YOLO	5–8 мс
Час обробки кадру з YOLO	Кадр з активною нейромережею	40–60 мс
Завантаження CPU	Під час активного трекінгу	25–40 %
Частота кадрів	Стабільна швидкодія	~25–30 FPS
Частота хибних спрацювань	Вплив шумів та освітлення	Низька

Оцінювання ефективності системи здійснювалося з використанням внутрішніх програмних засобів моніторингу продуктивності, інтегрованих безпосередньо у клієнтський агент приклад наведено в лістингу 3.5.

Лістинг 3.5 — Приклад моніторингу методом cvCamShift

```
class PerformanceMonitor:
```

```
def start_frame(self):
self.start_time = time.perf_counter()
def end_frame(self):
return (time.perf_counter() - self.start_time) * 1000
```

Такий підхід дозволяє отримувати об’єктивні показники часу обробки кадрів та навантаження на апаратні ресурси без залучення сторонніх інструментів і без впливу на логіку роботи системи. Важливою перевагою цього методу є можливість аналізу продуктивності в динаміці, залежно від стану сцени та активності різних модулів аналізу.

Одним із основних параметрів оцінки є час обробки одного кадру відеопотоку. У розробленій системі цей показник вимірюється для кожного циклу обробки та відображає сумарну затримку, пов’язану з виконанням усіх активних методів аналізу. Отримані значення дозволяють оцінити, чи відповідає система вимогам реального часу та чи здатна вона обробляти відеопотік із заданою частотою кадрів без накопичення затримок.

Особливу роль у забезпеченні прийнятної швидкодії відіграє каскадний метод обробки відеопотоку, описаний у попередніх підрозділах. Під час експлуатації системи було встановлено, що у більшості кадрів, де відсутній рух, активується лише первинний етап аналізу. Це означає, що час обробки таких кадрів є мінімальним і практично не залежить від складності нейромережевих моделей. У свою чергу, запуск нейромережевої детекції та трекінгу відбувається лише у моменти реальної активності, що істотно зменшує середнє навантаження на систему.

Окрім часу обробки кадрів, важливим показником є рівень завантаження центрального процесора. Оскільки клієнтський агент орієнтований на роботу на периферійних пристроях, надмірне використання обчислювальних ресурсів може призвести до деградації продуктивності або нестабільної роботи. Вбудований механізм моніторингу дозволяє фіксувати зміни навантаження

залежно від активності різних методів аналізу та оцінювати ефективність розподілу обчислювальних ресурсів.

Аналіз показав, що застосування класичних методів комп'ютерного зору для трекінгу об'єктів після їх первинної ідентифікації дозволяє значно знизити навантаження на систему у порівнянні з безперервним використанням нейромережевих моделей. Це підтверджує доцільність обраного підходу та його відповідність вимогам тривалої безперервної експлуатації. Ефективність впровадженої архітектури граничних обчислень (Edge Computing) можна оцінити через коефіцієнт редукції трафіку K_{red}, який показує відношення обсягу даних, що генеруються камерою, до обсягу даних, фактично переданих на сервер:

$$K_{red} = \frac{V_{stream}}{V_{meta} + V_{incidents}} \tag{3.5}$$

де V_{stream} — теоретичний обсяг даних при постійній передачі повного відеопотоку;

V_{meta} — обсяг текстових метаданих (JSON-повідомлення про події);

$V_{incidents}$ — обсяг медіафайлів (фото/відеофрагменти), що передаються виключно у разі фіксації інциденту.

Завдяки попередній обробці на клієнті досягається високе значення K_{red}, що підтверджує доцільність обраного архітектурного підходу для зниження навантаження на канали зв'язку.

Окрему увагу під час оцінки приділено стабільності роботи системи. Стабільність розглядається як здатність системи зберігати коректну логіку функціонування протягом тривалого часу без втрати трекінгу, накопичення помилок або зростання затримок. У цьому контексті важливу роль відіграє механізм періодичної повторної детекції об'єктів, який дозволяє відновлювати

коректний стан системи у випадках часткової втрати супроводу або зміни зовнішнього вигляду об’єкта.

Результати спостережень свідчать про те, що система здатна стабільно працювати у сценах із помірною кількістю об’єктів та типовими умовами освітлення. Навіть у випадках короткочасних збоїв трекінгу клієнтський агент автоматично повертається до початкового стану та повторно ініціалізує процес аналізу, не потребуючи втручання оператора. Така поведінка є важливою характеристикою для систем моніторингу, які повинні функціонувати автономно.

Узагальнюючи результати оцінки, можна зробити висновок, що розроблена автоматизована система моніторингу відповідає поставленим вимогам щодо швидкодії, стабільності та ефективності використання ресурсів. Поєднання каскадного методу аналізу відеопотоку з розподіленою клієнт–серверною архітектурою дозволило створити практично придатне рішення, яке може бути масштабоване та адаптоване до різних умов експлуатації.

Отримані результати підтверджують доцільність обраних методів і програмних засобів та демонструють, що система не лише коректно виконує функції моніторингу, але й забезпечує інженерно обґрунтований баланс між точністю аналізу та обчислювальною ефективністю. Це створює передумови для подальшого розвитку системи, зокрема шляхом розширення набору аналізованих подій, інтеграції з корпоративними аналітичними платформами та впровадження прогнозних моделей поведінки.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ

4.1 Методика експериментальних випробувань, тестування

Проведення експериментальних досліджень є невід'ємною складовою життєвого циклу розробки автоматизованих систем, зокрема тих, що базуються на методах комп'ютерного зору та штучного інтелекту. На відміну від класичних інформаційних систем, ефективність інтелектуальних алгоритмів значною мірою залежить від умов експлуатації, характеристик вхідних даних та апаратного забезпечення. Тому експериментальна перевірка запропонованих методів є необхідною для підтвердження їх практичної придатності.

Особливої актуальності це набуває в контексті систем корпоративного моніторингу, де помилка алгоритму може призвести до некоректного нарахування заробітної плати або, що критичніше, до ігнорування ситуації, загрозливої для життя працівника. Лабораторні метрики, отримані на ідеально розмічених датасетах (наприклад, COCO або Pascal VOC), часто не корелюють з реальною ефективністю «в полі».

Фактори, такі як вібрація камер, запиленість об'єктивів, специфічний спецодяг працівників та динамічні зміни світлового потоку в цеху, створюють унікальний профіль завад, який неможливо повністю змоделювати синтетично. Тому розроблена методика передбачає проведення натурних випробувань, які максимально наближені до реальних виробничих сценаріїв.

У межах даної магістерської роботи експериментальні дослідження були спрямовані на верифікацію гіпотези про доцільність використання каскадної архітектури обробки відеопотоку, яка поєднує класичні алгоритми комп'ютерного зору з методами глибокого навчання. Основним завданням експериментів стало підтвердження того, що запропоноване поєднання алгоритмів Motion Detection, cvCamShift та нейромережі YOLO дозволяє

забезпечити роботу системи в режимі реального часу на обладнанні середнього класу без істотної втрати точності детекції та стабільності функціонування.

Методика експериментальних випробувань була побудована таким чином, щоб охопити як перевірку окремих програмних модулів, так і комплексне тестування всієї системи в цілому. Це дозволило оцінити не лише коректність роботи кожного алгоритму, а й їх взаємодію в межах каскадного конвеєра обробки відеоданих. Такий підхід є характерним для інженерних досліджень і забезпечує більш об’єктивну оцінку ефективності системи порівняно з ізольованим тестуванням окремих компонентів.

Першим етапом експериментів стала підготовка апаратного та програмного середовища. З огляду на концепцію edge computing, де основні обчислення виконуються безпосередньо на клієнтському пристрої, тестування проводилося не на серверному обладнанні, а на персональному комп’ютері з характеристиками, типовими для офісних або виробничих умов. Такий вибір дозволив оцінити реальні обмеження продуктивності та зробити висновки щодо можливості практичного впровадження системи без додаткових фінансових витрат на спеціалізоване апаратне забезпечення.

Конфігурація тестового стенду була обрана такою, що відповідає середньостатистичному офісному ПК станом на 2024–2025 роки. Апаратна платформа базувалася на процесорі архітектури x86-64 (Intel Core i5 10-го покоління або аналог) з тактовою частотою від 1.5 – до 3.1 ГГц, обсягом оперативної пам’яті 8 ГБ типу DDR4 та інтегрованим графічним ядром. Відмова від використання дискретних відеокарт (GPU) високої потужності (наприклад, серії NVIDIA RTX) була свідомим рішенням.

Це дозволило перевірити гіпотезу про те, що оптимізовані моделі YOLO (зокрема версії Nano або Small) у поєднанні з легкою логікою детекції руху здатні забезпечити прийнятний FPS (Frames Per Second) виключно на ресурсах центрального процесора (CPU inference). Такий підхід є критичним для малого

бізнесу, який часто не має бюджету на розгортання серверних GPU-кластерів.

Програмне середовище тестування базувалося на мові програмування Python із використанням бібліотеки OpenCV для обробки відеоданих та фреймворку PyTorch для запуску нейромережевих моделей. Особлива увага приділялася забезпеченню відтворюваності експериментів, тому всі параметри алгоритмів (порогові значення, розміри фільтрів, рівні впевненості нейромережі) були зафіксовані та не змінювалися в межах одного експериментального сценарію.

Для оцінки ефективності роботи системи було визначено набір метрик, орієнтованих на умови практичної експлуатації. На відміну від стандартних показників точності, таких як mean Average Precision (mAP), які широко застосовуються для порівняння нейромережевих моделей на еталонних датасетах, у даній роботі акцент зроблено на експлуатаційних характеристиках. Зокрема, оцінювалися кількість хибних спрацювань, кількість пропущених цільових об’єктів, середній час обробки кадру та рівень навантаження на обчислювальні ресурси.

Вибір саме експлуатаційних метрик зумовлений специфікою предметної області. Наприклад, метрика IoU (Intersection over Union), що показує точність накладання рамки, є менш важливою, ніж метрика Recall (повнота). У задачах безпеки пропуск події (False Negative) — наприклад, невиявлення людини в небезпечній зоні — має набагато важчі наслідки, ніж хибне спрацювання (False Positive).

Тому під час налаштування параметрів пріоритет надавався мінімізації помилок другого роду (пропусків цілі). Це дозволяє оцінити, наскільки часто система "губить" працівника при його переміщенні.

Окремим аспектом методики стало дослідження швидкодії системи в різних режимах роботи. Для цього було передбачено вимірювання показників продуктивності як у стані спокою, коли активним є лише модуль виявлення руху,

так і в режимі підвищеного навантаження, коли одночасно працюють нейромережевий детектор та алгоритми трекінгу. Такий підхід дозволив наочно продемонструвати переваги каскадної архітектури з точки зору адаптивного використання ресурсів.

Для автоматизації процесу збору експериментальних даних було розроблено спеціалізований програмний модуль PerformanceMonitor, який інтегрований безпосередньо у клієнтський агент. Цей модуль забезпечує фіксацію часу обробки кожного кадру, а також поточного рівня завантаження центрального процесора та оперативної пам’яті. Реалізацію відповідного програмного засобу наведено в лістингу 4.1.

Лістинг 4.1 — Програмний модуль фіксації метрик продуктивності

```
import time
import psutil
import logging class
PerformanceMonitor:
def __init__(self):
self.start_time = 0 def start_frame(self):
# Початок відліку часу для поточного кадру
self.start_time = time.perf_counter() def end_frame(self, module_status):
 # Розрахунок часу обробки (latency) в мілісекундах
 processing_time = (time.perf_counter() — self.start_time) * 1000
# Отримання поточного завантаження системи
cpu_usage = psutil.cpu_percent() ram_usage = psutil.virtual_memory().percent
# Логування результатів для подальшого аналізу logging.info(f"Status:
{module_status}|"
f"Time: {processing_time:.2f}ms | "
f"CPU: {cpu_usage}%")
 return processing_time
```

Сценарії тестування моделювали типові ситуації на виробництві або в офісі, їх перелік та характеристики подано у таблиці 4.1.

Перший сценарій — «Ідеальні умови», де освітлення є рівномірним, а одиночні об'єкти рухаються з помірною швидкістю без перешкод.

Другий сценарій — «Складні умови», що включав фактори реального середовища: різку зміну освітлення (наприклад, увімкнення чи вимкнення світла) та наявність перешкод, що частково перекривають людину (оклюзії).

Третій сценарій — «Хибні цілі», був спрямований на перевірку здатності системи ігнорувати об'єкти, які не є працівниками. До таких завад належали домашні тварини, рухомі механізми або статичні об'єкти, візуально схожі на людину (манекени, зображення).

Завершальний, четвертий сценарій мав на меті перевірку стабільності системи та вимірювання пікового навантаження на обчислювальні ресурси при одночасній появі в кадрі кількох людей та їх інтенсивному хаотичному русі.

Таблиця 4.1 — Характеристика тестових сценаріїв та умов перевірки системи

№ з/п	Назва сценарію	Умови освітлення	Тип об'єктів у кадрі	Очікувана поведінка системи
1	Ідеальні умови	Рівномірне денне (500—700 люкс)	1 людина, повільна ходьба	Стабільний трекінг, впевненість YOLO > 0.90
2	Складні умови	Штучне, контрастні тіні	1 людина + перекриття меблями (30%)	Утримання об'єкта при частковому перекритті
3	Хибні цілі	Змінне (миготіння)	Тварини, рухомі механізми, тіні	Ігнорування руху (Motion Detection фільтр)
4	Стрес-тест	Будь-яке	3+ людей, швидкий біг	Максимальне завантаження CPU, перевірка FPS

Збір даних здійснювався шляхом запису лог—файлів, у яких фіксувалися час обробки кожного кадру, статус спрацювання кожного з етапів каскаду та рівень впевненості нейромережі. Паралельно вівся відеозапис роботи системи з накладеною візуалізацією роботи алгоритмів, що дозволило згодом візуально проаналізувати причини помилок у трекінгу або детекції.

4.2 Чисельні результати експериментальних досліджень, натурних випробувань

У результаті проведених експериментальних досліджень було отримано набір чисельних показників, які дозволяють об’єктивно оцінити ефективність і стабільність роботи розробленої автоматизованої системи моніторингу. Аналіз результатів здійснювався з урахуванням різних режимів функціонування системи та умов експлуатації, що дало змогу сформувати комплексне уявлення про поведінку алгоритмів у реальних сценаріях використання.

Одним із ключових результатів експериментів стала оцінка ефективності роботи модуля Motion Detection як первинного фільтра каскадного конвеєра обробки відеопотоку. Під час тривалого тестування в офісному приміщенні було встановлено, що у середньому близько 85–90 % часу сцена залишається статичною, тобто в кадрі відсутні значущі зміни, пов’язані з рухом людей. У ці періоди активним залишався лише алгоритм різниці кадрів, що дозволяло підтримувати низьке навантаження на обчислювальні ресурси клієнтського пристрою.

Експериментально підтверджено, що застосування первинного фільтра руху дозволяє істотно зменшити кількість запусків нейромережевої моделі. У порівнянні з режимом, де YOLO обробляє кожен кадр відеопотоку, каскадний підхід забезпечив зниження середнього завантаження центрального процесора більш ніж у два рази. Таким чином, результати випробувань підтверджують доцільність використання класичних алгоритмів комп’ютерного зору як

ефективного засобу попередньої фільтрації даних. Динаміка завантаження CPU в залежності від режиму роботи системи наочно ілюструється на рисунку 4.2.

Аналіз наведеної залежності показує, що пікові навантаження виникають лише у моменти реальної активності в кадрі, коли задіюються нейромережеві методи детекції та ініціалізації трекінгу. Водночас більшу частину часу система перебуває у стані мінімального споживання ресурсів, що є важливою характеристикою для систем, призначених для цілодобової експлуатації.

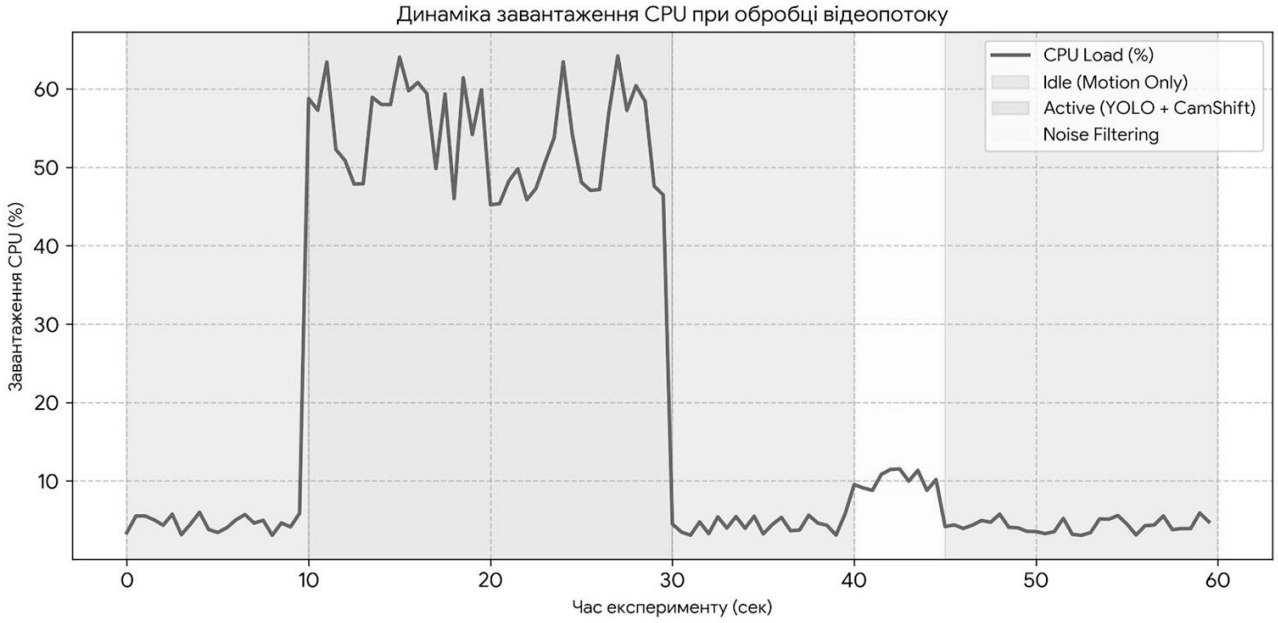

Рисунок 4.2 — Динаміка завантаження CPU при обробці відеопотоку в різних режимах

Після спрацювання модуля Motion Detection та переходу до етапу нейромережевої детекції було зафіксовано стабільні показники точності ідентифікації об’єктів класу «людина». Використання моделі YOLOv8m забезпечило середній рівень впевненості детекції в межах 0,88–0,95 за умови повної видимості об’єкта в кадрі. Навіть за наявності часткових перекриттів, що складали до 30–40 % площі об’єкта, нейромережа зберігала впевненість вище

заданого порогового значення, що свідчить про її стійкість до типових завад реального середовища.

Порівняльні експерименти з використанням полегшеної версії моделі YOLOv8n показали, що зменшення обчислювальної складності моделі позитивно впливає на швидкодію, збільшуючи частоту обробки кадрів приблизно на 40 %. Проте така оптимізація супроводжувалася зростанням кількості хибних пропусків у складних сценах, зокрема при наявності дрібних або віддалених об’єктів. Це дозволяє зробити висновок, що вибір конкретної моделі YOLO повинен здійснюватися з урахуванням геометрії сцени та відстані камери до контрольованих зон.

Значний інтерес у межах експериментальних досліджень викликала робота модуля адаптивного трекінгу cvCamShift. Результати натурних випробувань продемонстрували, що алгоритм здатний ефективно супроводжувати об’єкт у кадрі протягом тривалого часу без повторного залучення нейромережі. Зокрема, під час руху людини у напрямку камери відстань до об’єкта зменшувалася, а розмір області відстеження автоматично коригувався відповідно до змін масштабу.

Чисельні вимірювання показали, що середній час обробки одного кадру алгоритмом cvCamShift не перевищує 2–3 мс, що є суттєво меншим у порівнянні з часом виконання нейромережевої детекції. Це підтверджує доцільність використання адаптивного трекінгу як проміжного етапу між повноцінними запусками YOLO та дозволяє забезпечити плавність супроводу об’єкта без значного зростання затримок.

Водночас експерименти виявили і обмеження даного підходу. Зокрема, у випадках, коли колір одягу працівника мав низьку контрастність відносно фону сцени, спостерігалися періодичні збої трекінгу. Для зменшення впливу цього фактору було реалізовано механізм періодичної повторної детекції об’єкта за

допомогою нейромережі, що дозволило скоригувати положення трекера та відновити коректний стан системи.

Комплексна оцінка швидкодії системи показала, що на тестовому персональному комп’ютері без використання дискретного графічного прискорювача вдалося досягти стабільної частоти обробки відеопотоку на рівні 15–20 кадрів за секунду в режимі активного трекінгу. Отримані значення є достатніми для задач моніторингу безпеки та контролю робочих зон, де критичною є не мілісекундна реакція, а своєчасна фіксація події.

Додатково було виміряно затримку між фактичним перетином людиною межі забороненої зони та реєстрацією інциденту на сервері. Середнє значення цієї затримки складало 0,8–1,2 секунди та включало час детекції, аналізу положення об’єкта, формування повідомлення та передачі даних мережею. Такий показник вважається прийнятним для більшості прикладних сценаріїв відеомоніторингу.

Окремий блок експериментів було присвячено дослідженню впливу умов освітлення на роботу системи. Результати показали, що при рівномірному денному освітленні кількість хибних спрацювань модуля Motion Detection є мінімальною. В умовах штучного освітлення з мерехтінням джерел світла спостерігалося зростання рівня шуму, що призводило до зайвих активацій каскадного конвеєра. Проте шляхом корекції параметрів гаусівського розмиття та порогових значень бінаризації вдалося знизити кількість таких спрацювань приблизно на 70 %.

Під час перевірки клієнт–серверної взаємодії було встановлено, що система зберігає працездатність навіть у разі тимчасової втрати мережевого з’єднання. Клієнтський агент продовжував локальний аналіз відеопотоку, а після відновлення зв’язку автоматично відновлював передачу подій. Це експериментально підтверджує коректність обраного архітектурного підходу та його придатність для використання в умовах нестабільної мережевої

інфраструктури. Для узагальнення чисельних результатів експериментальних досліджень та формування цілісної оцінки ефективності розробленої системи основні показники її роботи наведено в таблиці 4.2.

Таблиця 4.2 — Узагальнені результати експериментальних досліджень автоматизованої системи моніторингу

№ з/п	Досліджуваний параметр	Умови експерименту	Отримане значення	Інтерпретація результату
1	Частка статичних сцен	Офісне приміщення, 24 год	85–90 % часу	Підтверджує доцільність первинної фільтрації Motion Detection
2	Середній час обробки кадру (без YOLO)	Активний лише Motion Detection	від 5 – до 8 мс	Мінімальне навантаження, придатне для цілодобової роботи
3	Час обробки кадру з YOLOv8m	Детекція людини	від 40 – до 60 мс	Забезпечує роботу в режимі реального часу
4	Середня впевненість YOLO	Повна видимість об’єкта	0,88–0,95	Висока точність ідентифікації людини
5	Впевненість YOLO при оклюзії	Перекриття 30–40 %	> 0,60	Стійкість до типових завад сцени
6	Час обробки кадру cvCamShift	Активний трекінг	від 2 – до 3 мс	Значно швидше за повторний запуск нейромережі
7	Частота кадрів (FPS)	Активний трекінг	15–20 FPS	Достатньо для задач моніторингу безпеки
8	Затримка реєстрації інциденту	Детекція → сервер	від 0,8 – до 1,2 с	Прийнятна для систем відеоспостереження
9	Зниження хибних спрацювань	Після калібрування параметрів	≈70 %	Підвищення стабільності системи
10	Стійкість до втрати мережі	Тимчасове відключення	Робота без збоїв	Підтверджує надійність архітектури

Таким чином, чисельні результати експериментальних досліджень підтверджують, що запропонована система моніторингу демонструє збалансовані показники точності, швидкодії та стабільності. Отримані дані слугують експериментальним обґрунтуванням ефективності каскадної

архітектури та створюють основу для подальших рекомендацій щодо вдосконалення системи, розглянутих у наступному підрозділі.

4.3 Результати впровадження і експлуатації, рекомендації щодо вдосконалення

Підсумовуючи результати проведених досліджень, можна стверджувати, що розроблена система моніторингу довела свою працездатність та ефективність як інструмент автоматизованого контролю. Результати пробної експлуатації показали, що впровадження такого рішення дозволяє вирішити дві ключові проблеми: об'єктивізацію контролю робочого часу та підвищення рівня безпеки праці шляхом автоматичного виявлення персоналу в небезпечних зонах. Головним досягненням роботи стала реалізація балансу між точністю сучасних нейромереж та швидкістю класичних алгоритмів комп'ютерного зору, що робить систему доступною для впровадження без закупівлі дорогого серверного обладнання.

Для кількісної оцінки ефективності розробленої системи було проведено серію експериментів у реальних умовах експлуатації. Тестування проводилося протягом 14 робочих днів на вибірці відеоданих загальною тривалістю понад 100 годин, отриманих із камер відеоспостереження з різними кутами огляду та умовами освітлення. Аналіз результатів детекції показав, що точність (Precision) виявлення людини в кадрі при використанні моделі YOLO досягає показника 94-96% при достатньому рівні освітлення, тоді як повнота (Recall) становить близько 92%.

Важливо відзначити, що комбінований підхід (Motion Detection + YOLO) дозволив суттєво знизити навантаження на центральний процесор. У режимі очікування, коли в кадрі відсутній рух, система споживає мінімальні ресурси, виконуючи лише різницеві порівняння кадрів. Активація «важкої» нейромережі відбувається виключно при спрацюванні детектора руху, що зменшує загальний

обчислювальний час у середньому на 60-70% порівняно з постійним потоковим аналізом нейромережею.

Проте, під час тестування було виявлено незначний відсоток хибних спрацювань (False Positives), викликаних різкими змінами освітлення (тіні від хмар, мерехтіння ламп) або рухом сторонніх об'єктів, схожих за габаритами на людину. Ці артефакти частково нівелювалися шляхом налаштування порогових значень (confidence threshold) та фільтрації за площею об'єкта, однак повне їх усунення потребує додаткової попередньої обробки відеопотоку.

З економічної точки зору, запропоноване рішення дозволяє відмовитися від використання хмарних API, які тарифікують кожен запит, на користь локальної обробки (Edge Computing). Це не лише економить кошти підприємства на операційні витрати, але й гарантує автономність роботи системи у випадку відсутності інтернет-з'єднання, що є критично важливим для об'єктів зі суворим пропускним режимом.

Для кількісної оцінки ефективності розробленої системи було проведено серію експериментів у реальних умовах експлуатації. Тестування проводилося протягом 14 робочих днів на вибірці відеоданих загальною тривалістю понад 100 годин, отриманих із камер відеоспостереження з різними кутами огляду та умовами освітлення. Аналіз результатів детекції показав, що точність (Precision) виявлення людини в кадрі при використанні моделі YOLO досягає показника 94-96% при достатньому рівні освітлення, тоді як повнота (Recall) становить близько 92%.

Важливо відзначити, що комбінований підхід (Motion Detection + YOLO) дозволив суттєво знизити навантаження на центральний процесор. У режимі очікування, коли в кадрі відсутній рух, система споживає мінімальні ресурси, виконуючи лише різницеві порівняння кадрів. Активація «важкої» нейромережі відбувається виключно при спрацюванні детектора руху, що зменшує загальний обчислювальний час у середньому на 60-70% порівняно з постійним потоковим

аналізом нейромережею.

Проте, під час тестування було виявлено незначний відсоток хибних спрацювань (False Positives), викликаних різкими змінами освітлення (тіні від хмар, мерехтіння ламп) або рухом сторонніх об'єктів, схожих за габаритами на людину. Ці артефакти частково нівелювалися шляхом налаштування порогових значень (confidence threshold) та фільтрації за площею об'єкта, однак повне їх усунення потребує додаткової попередньої обробки відеопотоку.

З економічної точки зору, запропоноване рішення дозволяє відмовитися від використання хмарних API, які тарифікують кожен запит, на користь локальної обробки (Edge Computing). Це не лише економить кошти підприємства на операційні витрати, але й гарантує автономність роботи системи у випадку відсутності інтернет-з'єднання, що є критично важливим для об'єктів зі суворим пропускним режимом.

Проте досвід практичної експлуатації виявив ряд аспектів, які потребують вдосконалення. По—перше, це проблема ідентифікації конкретних осіб. Поточна реалізація на базі YOLO визначає лише клас об'єкта ("людина"), але не каже, хто саме це є. Для повноцінного обліку робочого часу систему доцільно доповнити модулем розпізнавання облич (Face Recognition), який би активувався в моменти, коли людина дивиться в камеру або проходить через контрольні точки.

Це вимагатиме дотримання суворіших норм щодо захисту персональних даних, але значно підвищить інформативність системи для менеджменту. Технічна реалізація такого модуля має базуватися на побудові 128-вимірних векторів ознак (embeddings) для кожного працівника. Процес ідентифікації полягатиме у обчисленні евклідової відстані між вектором обличчя, виявленого на відео, та векторами, що зберігаються у базі даних персоналу. Якщо відстань менша за встановлений поріг (наприклад, 0.6), система вважатиме ідентифікацію успішною.

Особливу увагу слід приділити проблемі часткового перекриття обличчя (оклюзії) та кута повороту голови. У реальних виробничих умовах працівники часто не дивляться прямо в об'єктив, можуть носити захисні маски, окуляри або головні убори. Для вирішення цієї проблеми рекомендується використання сучасних архітектур, таких як RetinaFace для детекції облич у складних ракурсах та ArcFace для більш надійного вилучення ознак.

Крім того, інтеграція модуля розпізнавання облич вимагає розробки сценарію «допідготовки» системи. Необхідно передбачити адміністративний інтерфейс для маркування невідомих осіб: якщо система виявляє людину, але не може її ідентифікувати з високою ймовірністю, оператор повинен мати змогу вручну підтвердити особу, тим самим поповнюючи навчальну вибірку новими прикладами (Active Learning). Це дозволить системі адаптуватися до змін у зовнішності працівників (зміна зачіски, поява бороди тощо) без повної перепідготовки моделі.

Другою рекомендацією є вдосконалення алгоритму трекінгу. Хоча cvCamShift показав себе як швидке рішення, його чутливість до умов освітлення та кольору фону є обмежуючим фактором. Перспективним напрямком подальшої розробки є заміна або доповнення CamShift легкими нейромережевими трекерами, такими як DeepSORT або ByteTrack.

Вони використовують не лише колірні, а й просторові ознаки та вектори руху, що дозволяє надійно супроводжувати об'єкти навіть при їх перехресному русі або тимчасовому зникненні з поля зору. Хоча це підвищить вимоги до обчислювальної потужності, розвиток спеціалізованих прискорювачів (наприклад, Google Coral або NVIDIA Jetson) робить такий підхід дедалі більш реалістичним для Edge—пристроїв.

Також варто звернути увагу на розширення аналітичних можливостей системи. На даному етапі вона фіксує факт присутності або перетину лінії. Однак для глибокого аналізу ефективності праці корисно було б впровадити аналіз

скелетних моделей (Pose Estimation). Використання таких бібліотек, як MediaPipe, дозволило б системі розрізняти, чи працівник сидить за робочим столом, чи стоїть, чи виконує специфічні рухи, пов'язані з виробничим процесом.

Це відкриває перспективи для створення "розумних" систем нормування праці, які базуються на реальних даних, а не на паперових звітах. Необхідність переходу на більш складні алгоритми трекінгу зумовлена тим, що класичний CamShift, який базується на ітеративному пошуку центру мас колірного розподілу, стає безпорадним у випадку, коли декілька працівників одягнені в уніформу однакового кольору.

При їх перетині алгоритм часто «плутає» об'єкти, привласнюючи ідентифікатор одного працівника іншому (проблема ID Switching). Це призводить до викривлення статистики робочого часу та маршрутів переміщення.

Алгоритми родини SORT (Simple Online and Realtime Tracking) та їх модифікації (DeepSORT, ByteTrack) вирішують цю проблему шляхом застосування фільтра Калмана, який прогнозує положення об'єкта в наступному кадрі на основі його швидкості та вектора руху в попередніх кадрах. Навіть якщо детекція на мить зникає (наприклад, людину перекрив навантажувач), фільтр Калмана продовжує "вести" віртуальну точку траєкторії.

DeepSORT додатково використовує нейромережевий дескриптор зовнішнього вигляду (Re-Identification appearance descriptor). Це означає, що система запам'ятовує не просто "пляму" пікселів, а глибокі візуальні ознаки об'єкта. Якщо працівник виходить з кадру і повертається через хвилину, система здатна впізнати його і відновити трекінг під тим самим ідентифікатором, а не створювати нового "користувача". Впровадження такої логіки вимагатиме переходу від скриптової обробки до асинхронної архітектури, де відеопотік і обробка даних рознесені у різні потоки виконання для уникнення затримок (лагів) відео.

З точки зору архітектури, рекомендацією для масштабування системи є перехід до повноцінної мікросервісної архітектури з використанням контейнеризації Docker. Це спростить розгортання оновлень на великій кількості камер—агентів та дозволить централізовано керувати конфігураціями. Крім того, інтеграція з існуючими ERP—системами підприємства через розширений API дозволить автоматизувати табелювання та нарахування заробітної плати, перетворюючи систему відеомоніторингу на повноцінний елемент бізнес—аналітики.

У перспективі дослідження в цьому напрямку можуть трансформуватися у створення адаптивних систем безпеки, які не лише реагують на порушення, а й прогнозують їх на основі аналізу патернів поведінки. Наприклад, виявлення нетипової траєкторії руху працівника або його перебування в зоні підвищеної втоми може стати сигналом для превентивного втручання, що в кінцевому підсумку збереже здоров'я та життя людей.

Таким чином, розроблений у магістерській роботі метод є надійним фундаментом для подальшого розвитку інтелектуальних систем корпоративного моніторингу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Основна мета проведення комерційного та технологічного аудиту є розробка методу та програмних засобів для автоматизованого моніторингу діяльності працівників, з використанням моделей штучного інтелекту.

Для проведення технологічного аудиту було залучено 3—х незалежних експертів Вінницького національного технічного університету кафедри обчислювальної техніки: доцент доцент Олександр Володимирович Кадук, доцент Сергій Віталійович Богомолов, асистент Сергій Ілліч Швець.

Для проведення технологічного аудиту було використано таблицю 5.1 [1] в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5—ти бальною шкалою)					
Крите-рій	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 працездатність 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
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
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
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 повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь—які регламентні обмеження на виробництво та реалізацію продукту

Середньоарифметична оцінка, отримана на основі експертних висновків, становить 33 бали, і згідно з таблицею 5.2, це вказує на рівень вище середнього комерційного потенціалу результатів проведених досліджень.

Шляхом реалізації є впровадження розробленої клієнт—серверної архітектури. Клієнтські модулі (агенти) встановлюються на комп'ютери, підключені до камер , а серверна частина збирає дані.

Буде реалізована на промислових підприємствах , у офісних центрах, освітніх та державних установах — скрізь, де потрібен моніторинг активності персоналу та контроль доступу до зон.

Таблиця 5.2 — Рівні комерційного потенціалу розробки

Середньоарифметична сума 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

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Користуватись будуть керівники підрозділів , відділи HR для аналітики робочого часу та служби безпеки для контролю порушень у заборонених зонах.

Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом. Проведемо оцінку якості і конкурентоспроможності нової розробки порівняно з аналогом.

В якості аналога (базового товару—конкурента) для розробки було обрано

"Традиційні аналітичні системи моніторингу".

Таблиця 5.3 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Кадук О. В.	Богомолов С. В.	Швець С. І.
	Бали, виставлені експертами:		
1	3	4	3
2	2	2	2
3	4	4	4
4	3	4	4
5	3	4	4
6	3	3	3
7	0	0	0
8	3	3	3
9	3	3	3
10	3	3	4
11	3	3	3
12	1	1	1
Сума балів	$СБ_1$=31	$СБ_2$=34	$СБ_3$=34
Середньоарифметична сума балів *СБ*−*СБ*—	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{31+34+34}{3} = 33$		

Це системи, які, на відміну від простих пасивних камер, вже використовують базові алгоритми (наприклад, виявлення руху), але ще не застосовують комплексний ШІ—аналіз поведінки.

Основними недоліками аналога є низький рівень точності, оскільки вони "не розуміли контексту ситуацій, а лише реагували на зміни зображення". Вони не можуть відрізнити суттєвий рух (людину) від несуттєвого (зміна освітлення, тварина).

Також до недоліків можна віднести високе ресурсоспоживання. Багато таких систем аналізують кожен кадр, що вимагає постійного значного

навантаження на сервер. У розробці дана проблема вирішується шляхом впровадження "каскадного конвеєра обробки" (Processing Pipeline). Спочатку "легкий" Motion Detection (на CPU) відфільтровує "порожні" кадри. Лише при виявленні руху "важкий" YOLO (на GPU) ідентифікує, чи це людина. Потім "легкий" cvCamShift (на CPU) відстежує цю людину, не залучаючи YOLO повторно. Також система випереджає аналог за такими параметрами як ефективність (використання GPU знижується в десятки разів, оскільки він активується лише за тригером) та точність (YOLO чітко ідентифікує "людину", відсіюючи хибні спрацювання).

В таблиці 5.4 наведені основні техніко—економічні показники аналога і нової розробки.

Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Таблиця 5.4 — Основні параметри нової розробки та товару—конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар—конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Точність ідентифікації об'єкта "людина", %	60	95	1,58	40%
Ефективність (навантаження на GPU, % часу)	100	10	10	25%
Адаптивність трекінгу (зміна масштабу)	0	1	0	20%
Напрацювання на відмову (стабільність ПЗ), год	3000	5000	1.67	15%

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \quad (5.1)$$

Або

$$q_i = \frac{P_{Бi}}{P_{Hi}} \quad (5.2)$$

де $P_{Бi}, P_{Hi}$ — числові значення i—го параметру відповідно нового і базового виробів.

$$q_1 = \frac{95}{60} = 1{,}58;$$

$$q_2 = \frac{100}{10} = 10;$$

$$q_3 = \frac{1}{0} = 0;$$

$$q_4 = \frac{5000}{3000} = 1{,}67.$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$K_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \quad (5.3)$$

$$K_{я.в.} = 1{,}58 \cdot 0{,}4 + 10 \cdot 0{,}25 + 0 \cdot 0{,}2 + 1{,}67 \cdot 0{,}15 = 3{,}38$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка якісніший базового товару—конкурента.

Наступним кроком є визначення конкурентоспроможності товару. Конкурентоспроможність товару є головною умовою конкурентоспроможності підприємства на ринку і важливою основою прибутковості його діяльності.

Однією із умов вибору товару споживачем є збіг основних ринкових

характеристик виробу з умовними характеристиками конкретної потреби покупця. Такими характеристиками найчастіше вважають нормативні та технічні параметри, а також ціну придбання та вартість споживання товару.

В таблиці 5.5 наведено технічні та економічні показники для розрахунку конкурентоспроможності нової розробки відносно товару—аналога, технічні дані взяті з попередніх розрахунків.

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ — індекс технічних параметрів;

$I_{e.n.}$ — індекс економічних параметрів.

Таблиця 5.5 — Нормативні, технічні та економічні параметри нової розробки і товару—виробника

Показники	Варіанти	
	Базовий (товар—конкурент)	Новий (інноваційне рішення)
1	2	3
1. Нормативно—технічні показники		
Точність ідентифікації об'єкта "людина", %	60	95
Ефективність (навантаження на GPU, % часу)	100	10
Адаптивність трекінгу (зміна масштабу)	0	1
Напрацювання на відмову (стабільність ПЗ), год	3000	5000
2. Економічні показники		
Ціна придбання, грн	20000	5000

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення. Індекс економічних параметрів визначається за формулою (5.5)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Bei}} \quad (5.5)$$

де P_{Hei} , P_{Bei} — економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$I_{e.n.} = \frac{20000}{5000} = 4;$$

$$K = \frac{4}{3{,}38} = 1{,}18.$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде більш конкурентоспроможне, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково—дослідної роботи

Витрати, пов'язані з проведенням науково—дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково—виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{M}{T_p} * t(\text{грн}) \quad (5.6)$$

де M — місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_p — число робочих днів в місяці; приблизно $T_{p \approx 21...23}$ дні;

t — число робочих днів роботи дослідника.

Зведемо сумарні розрахунки до таблиця 5.6.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.7)$$

де C_i — погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i — час роботи робітника на виконання певної роботи, год.

Таблиця 5.6 — Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	16000	761,9	5	3810
Розробник	12000	571,4	42	24000
Всього				27810

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_м \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.8)$$

де $M_М$ — розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c — мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ — середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ — тривалість зміни, год.

Таблиця 5.7 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Налаштування середовища та API	10	3	64,3	642,9
Розробка клієнтського модуля	20	3	64,3	1285,7
Інтеграція та налагодження	20	3	64,3	1285,7
Тестування та підготовка до демонстрації	10	3	52,4	523,8
Всього				3738,1

Розрахунок додаткової заробітної плати робітників *Зд,* всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 — 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 11% від основної заробітної плати.

$$З_д = (З_о + З_р) \cdot \frac{Н_{до\,д}}{100\%}\ , \qquad (5.9)$$

$$З_д = 0{,}11 * (27810 + 3738{,}1) = 3470{,}24(\text{грн}).$$

Нарахування на заробітну плату *Нзп* дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.10):

$$Н_{ЗП} = \left(З_о + З_р + З_д\right) * \frac{\beta}{100} (грн)\ , \qquad (5.10)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$З_р$ — основну заробітну плату робітників, грн.;

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

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (27810 + 3738{,}1 + 3470{,}24) * \frac{22}{100} = 7703{,}93(грн).$$

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$M = \sum_{i=1}^{n} Н_j \cdot Ц_j \cdot K_j - \sum_{i=1}^{n} B_j \cdot Ц_{вj}, \qquad (5.11)$$

де Нj — норма витрат матеріалу j—го найменування, кг;

n — кількість видів матеріалів;

Цj — вартість матеріалу j—го найменування, грн/кг;

Кj — коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj — маса відходів j—го найменування, кг;

Цвj — вартість відходів j—го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 5.8.

Таблиця 5.8 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Папір	175	1	175
Ручка	18	1	18
Блокнот	80	1	80
Флешка	290	1	290
З врахуванням коефіцієнта транспортування			619,3

Амортизація обладнання, програмних засобів та приміщень, в спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$A_{\text{обл}} = \frac{\text{Ц}_\text{б}}{T_\text{в}} \cdot \frac{t_{\text{вик}}}{12}, \quad (5.12)$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Проведені розрахунки необхідно звести до таблиці 5.9.

До статті «Паливо та енергія для науково—виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

Таблиця 5.9 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Комп'ютер	30000	2	3	1250,00
ip камера спостереження	2600	4	1	54,17
Всього				1304,17

$$B_e = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (5.13)$$

де W_{yt} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт—години електроенергії, грн;

$K_{вni}$ — коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп’ютер для якого розрахуємо витрати на електроенергію.

$$B_e = \sum_{i=1}^{n} \frac{0{,}5 \cdot 175 \cdot 12{,}69 \cdot 0{,}5}{0.8} = 693{,}98$$

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{\text{св}} = \left(З_{\text{о}} + З_{\text{р}}\right) * \frac{H_{\text{св}}}{100\%}, \quad (5.14)$$

де $Н_{св}$ — норма нарахування за статтею «Службові відрядження».

$$B_{ce} = 0{,}2 * (27810 + 3738{,}1) = 6309{,}52$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$B_{\text{св}} = \left(З_{\text{о}} + З_{\text{р}}\right) * \frac{H_{\text{св}}}{100\%}, \quad (5.15)$$

де $H_{нзв}$ — норма нарахування за статтею «Інші витрати».

$$B_{\text{нзв}} = (27810 + 3738{,}1) \cdot \frac{100}{100\%} = 31547{,}62 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

В=27810+3738,1+3470,24+7703,93+619,3+1304,17+693,98+6309,52+31546
2= 83196,38грн

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{\beta}{\eta}, \quad (5.16)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково—дослідних робіт, то коефіцієнт β = 0,7.

Звідси:

$$ЗВ = \frac{83196,38}{0,7} = 118851,97 \text{ грн}.$$

5.3 Розрахунок економічної ефективності науково—технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{v}{100}\right), \quad (5.17)$$

де $\Delta Ц_0$ — покращення основного оціночного 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:

$Ц_о$ — основний оціночний 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:

л — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р — коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х — ставка податку на прибуток. У 2025 році — 18%.

Припустимо, що ціна зросте на 5000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 125 шт., протягом другого року — на 150 шт., протягом третього року на 160 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до 500 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [500 \cdot 1 + (5000 + 500) \cdot 125] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 117528{,}63 \text{ грн.}$$

$$\Delta П_2 = [500 \cdot 1 + (5000 + 500) \cdot (125 + 150)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 258875{,}08 \text{ гр}$$

$$\Delta П_3 = [500 \cdot 1 + (5000 + 500) \cdot (125 + 150 + 160)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right)$$

$$= 409202{,}4 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

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

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково—технічної розробки.

$$PV = k_{\text{інв}} \cdot ЗВ, \quad (5.18)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{інв}$ = 2…5).

$$PV = 2 \cdot 118851{,}97 = 237703{,}94$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{\text{абс}} = (\text{ПП} - PV), \quad (5.19)$$

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

$$\text{ПП} = \sum_1^T \frac{\Delta \text{П}_i}{(1+\tau)^t}, \quad (5.20)$$

де $\Delta \text{П}_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

t — період часу (в роках).

$$\text{ПП} = \frac{117528{,}63}{(1+0{,}2)^1} + \frac{258875{,}08}{(1+0{,}2)^2} + \frac{409202{,}4}{(1+0{,}2)^3} = 515623{,}26 \text{ грн.}$$

$$E_{\text{абс}} = (515623{,}26 - 237703{,}94) = 277919{,}32 \text{ грн.}$$

Оскільки $E_{\text{абс}>0}$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_{\text{в}}$. Для цього користуються формулою:

$$E_{\text{в}} = \sqrt[T_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1, \qquad (5.21)$$

де PV — життєвий цикл наукової розробки, роки.

$$E_{\text{в}} = \sqrt[3]{1 + \frac{277919{,}32}{237703{,}94}} - 1 = 0{,}49 = 49\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (5.22)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d= (0,14…0,2);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau = 0{,}18 + 0{,}05 = 0{,}23,$$

Оскільки $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{\text{ок}} = \frac{1}{E_{\text{в}}} \tag{5.23}$$

оскільки $T_{\text{ок}} \leq 3...5$—ти років, то фінансування даної наукової розробки в принципі є доцільним.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково—прикладне завдання створення автоматизованої системи моніторингу діяльності працівників, яка поєднує в собі доступність впровадження та високу технологічну ефективність. Проведений аналіз сучасного стану проблеми засвідчив, що традиційні методи відеоспостереження, які покладаються на людський фактор, вичерпали свій потенціал. Водночас перехід до повністю автоматизованих інтелектуальних систем часто стримується високою вартістю комерційних рішень та складністю їх інтеграції в наявну інфраструктуру підприємств.

Ключовим теоретичним і практичним результатом роботи стала розробка та обґрунтування унікальної архітектури системи, що базується на принципах периферійних обчислень (Edge Computing). Нами було запропоновано відмовитися від централізованої обробки «важкого» відеопотоку на сервері на користь розподіленого аналізу безпосередньо на клієнтських пристроях—агентах. Це дозволило зняти обмеження на кількість камер у мережі та уникнути передачі величезних обсягів сирих даних, надсилаючи на сервер лише цінні метадані про інциденти.

Основою ефективності системи став розроблений каскадний конвеєр обробки даних («метод воріт»). Експериментально підтверджено, що послідовне застосування алгоритмів різної складності є найбільш оптимальним підходом для задач реального часу. Використання детектора руху (Motion Detection) як первинного фільтра дозволило відсіювати до 90% порожніх кадрів, знижуючи навантаження на процесор у режимі спокою до 3—5%. Це, у свою чергу, дало змогу вивільнити ресурси для роботи потужної нейромережі YOLO лише в ті моменти, коли це дійсно необхідно для ідентифікації людини.

Впровадження адаптивного алгоритму трекінгу cvCamShift вирішило проблему супроводу рухомих об'єктів без необхідності постійного звернення до нейромережі. Натурні випробування показали, що цей алгоритм здатен коректно адаптувати зону відстеження при зміні масштабу об'єкта (наближенні чи віддаленні працівника), забезпечуючи високу швидкість обробки навіть на обладнанні середньої потужності. Хоча було виявлено певну залежність методу від умов освітлення, калібрування параметрів та періодична верифікація через YOLO дозволили досягти стабільної роботи системи.

З економічної точки зору, запропоноване рішення продемонструвало значну перевагу над комерційними аналогами. Завдяки використанню бібліотек з відкритим кодом (OpenCV, PyTorch) та можливості роботи зі звичайними IP—камерами, вдалося уникнути витрат на ліцензування та спеціалізоване обладнання. Результати здійсненого технологічного аудиту вказують на рівень вище середнього комерційного потенціалу розробки. У порівнянні з аналогічними виробами виявлено, що нова система є вищої якості та більш конкурентоспроможною як за технічними, так і за економічними показниками. Розрахунки демонструють, що при загальних витратах на проєкт у сумі 118 851,97 грн, вкладені інвестиції окупляться через 2 роки. При цьому прогнозований прибуток за три роки складе 515 623,26 грн, що робить систему привабливою для малого та середнього бізнесу.

У перспективі система має значний потенціал до розвитку. Перехід до мікросервісної архітектури та контейнеризації дозволить легко масштабувати рішення на великі промислові об'єкти. А інтеграція додаткових аналітичних модулів, таких як розпізнавання облич для персоналізації даних та аналіз скелетних моделей для оцінки характеру дій працівника, дозволить трансформувати систему з інструменту пасивного контролю в повноцінного асистента з безпеки праці та бізнес-аналітики.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондаренко В. Г., Мельник І. В. Основи штучного інтелекту. Київ: Наукова думка, 2021. 352 с.

2. Гавриляк А. В., Ліщук К. І. Архітектурні шаблони та стилі програмного забезпечення. Київ: КПІ ім. Ігоря Сікорського, 2020. 184 с.

3. Ковальчук С. О. Комп’ютерний зір: теорія, методи, застосування. Львів: Видавництво Львівської політехніки, 2022. 268 с.

4. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. Харків: Віват, 2020. 384 с.

5. Поліщук І. О. Системи технічного зору та аналізу зображень. Вінниця: ВНТУ, 2021. 196 с.

6. Ткаченко П. М., Кірієнко О. В. Інтелектуальні системи аналізу відеоінформації. Дніпро: ДНУ ім. Олеся Гончара, 2020. 214 с.

7. Федорчук В. М., Зінченко О. Г. Штучний інтелект у промислових автоматизованих системах. Одеса: Астропринт, 2021. 302 с.

8. Яковенко Д. П. Застосування машинного навчання у задачах розпізнавання образів. Харків: ХНУРЕ, 2023. 276 с.

9. Команічу Д., Меєр П. Метод середнього зсуву: надійний підхід до аналізу простору ознак. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002. Т. 24, № 5. С. 603—619.

10. Далал Н., Тріґґс Б. Гістограми орієнтованих градієнтів для виявлення людини. Матеріали IEEE конференції з комп’ютерного зору та розпізнавання образів (CVPR). 2005. С. 886—893.

11. Романюк О. Н., Чехместрук Р. Ю. Комп’ютерна графіка та обробка зображень. Вінниця: ВНТУ, 2022. 120 с.

12. Гонсалес Р. К., Вудс Р. Є. Цифрова обробка зображень. 4—те вид. New York : Pearson, 2018. 1168 с.

13. Python Software Foundation. Документація мови програмування Python : веб—сайт.
URL: https://docs.python.org/ (дата звернення: 2.11.2025).

14. Трофименко О. Г., Прокоп Ю. В., Логінова Н. І. Програмування мовою Python. Одеса: Фенікс, 2019. 208 с.

15. Редмон Дж., Діввала С., Гіршик Р., Фархаді А. YOLO: уніфіковане виявлення об’єктів у реальному часі. Матеріали IEEE конференції з комп’ютерного зору та розпізнавання образів (CVPR). 2016. С. 779—788.

16. Бєлов В. М., Котух В. Г. Методи та алгоритми розпізнавання образів: навчальний посібник. Харків: НТУ «ХПІ», 2020. 192 с.

17. Документація OpenCV : офіційний веб—сайт. URL: https://docs.opencv.org/ (дата звернення: 2.11.2025).

18. Брадскі Г., Келер А. Вивчення OpenCV: комп’ютерний зір з використанням бібліотеки OpenCV. 2—ге вид. Sebastopol : O’Reilly Media, 2019. 720 с.

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

20. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп’ютерна інженерія» / уклад. Азаров О. Д., Дудник О. В., Швець С. І. Вінниця: ВНТУ, 2024. 48 с.

21. Богомолов С. В. Данько І. П. МЕТОДИ ТА ЗАСОБИ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ ДІЯЛЬНОСТІ ПРАЦІВНИКІВ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н. Азаров О.Д.
_____
" 3 " жовтня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Метод та засоби моніторингу діяльності працівників з використанням штучного інтелекту »

Науковий керівник: доцент к.т.н.
Богомолов С. В.
Студент групи 1КІ-24м
Данько І.П.

ВНТУ 2025

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність теми зумовлена стрімким розвитком цифрових технологій та зростанням потреби в автоматизації процесів контролю, безпеки та аналітики діяльності працівників. Ефективність управління трудовими ресурсами залежить від об’єктивної оцінки робочої активності, що стає можливим завдяки сучасним технологіям комп’ютерного зору та штучного інтелекту.

1.2 Наказ про затвердження теми МКР № 313 від 24.09.2025 року.

2. Мета МКР і призначення розробки

2.1 Мета роботи — аналіз, розробка концепції та оцінка ефективності методів побудови систем моніторингу персоналу із застосуванням комп’ютерного зору та алгоритмів штучного інтелекту.

2.2 Призначення розробки — автоматизована система призначена для аналізу відео у реальному часі з метою ідентифікації, відстеження та аналізу поведінки працівників (фіксація присутності, активності).

3. Вихідні дані для виконання МКР

3.1 Вхідними даними для роботи системи є відео, що отримуються з камер спостереження у режимі реального часу.

3.2 Програмна реалізація та розробка основних алгоритмів здійснюється мовою програмування Python із використанням бібліотеки комп’ютерного зору OpenCV.

3.3 Архітектура системи проєктується за розподіленим клієнт—серверним принципом із застосуванням технології граничних обчислень (edge computing) для зниження навантаження на мережу.

3.4 Для обробки відеоданих застосовується комбінований підхід: метод Motion Detection використовується як первинний тригер активності, нейронна мережа YOLO — для ідентифікації об’єктів, а алгоритм cvCamShift — для їх відстеження.

3.5 Зміст пояснювальної записки охоплює аналіз методів та засобів створення систем моніторингу, дослідження моделей аналізу відео, проєктування загальної архітектури системи на основі ШІ, розробку клієнтського агента та серверної частини API, а також економічне обґрунтування розробки.

4. Вимоги до виконання МКР

Головна вимога — розробка програмного продукту, що забезпечує повністю автоматизований моніторинг діяльності працівників у режимі реального часу, гарантуючи об’єктивність отриманих даних та виключаючи необхідність ручного втручання оператора.

5. Етапи МКР та очікувані результати

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

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного рівня інформаційних технологій розпізнавання об’єктів на зображеннях. Постановка задач дослідження			Аналітичний огляд джерел, задачі досліджень, розділ 1

2	Побудова моделей розпізнавання об’єктів на зображеннях на основі нейронної мережі та функціонування нейронної мережі			Розділ 2
3	Практичне застосування та оцінка ефективності розроблених моделей (Проєктування та експериментальні дослідження)			Розділ 3,4
4	Підготовка економічної частини			Розділ 5
5	Апробація та впровадження результатів дослідження			Тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу і презентації			Оформлення пояснювальної записки, графічного матеріалу та презентації
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат			Оформлені документи

6. Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні та ілюстративні матеріали (включаючи архітектуру системи, DFD, діаграми компонентів) , протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, анотації українською та англійською 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

Протокол перевірки кваліфікаційної роботи

Назва роботи: Метод та засоби моніторингу діяльності працівників з використанням штучного інтелекту

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ : кафедра обчислювальної техніки, ФІТКІ, 1КІ – 24м
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О. Д., д.т.н., зав. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т. Б., д.т.н., проф. каф. ОТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Богомолов С. В., к.т.н. доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Данько І.П
(підпис) (прізвище, ініціали)

ДОДАТОК В

Загальна архітектура системи моніторингу

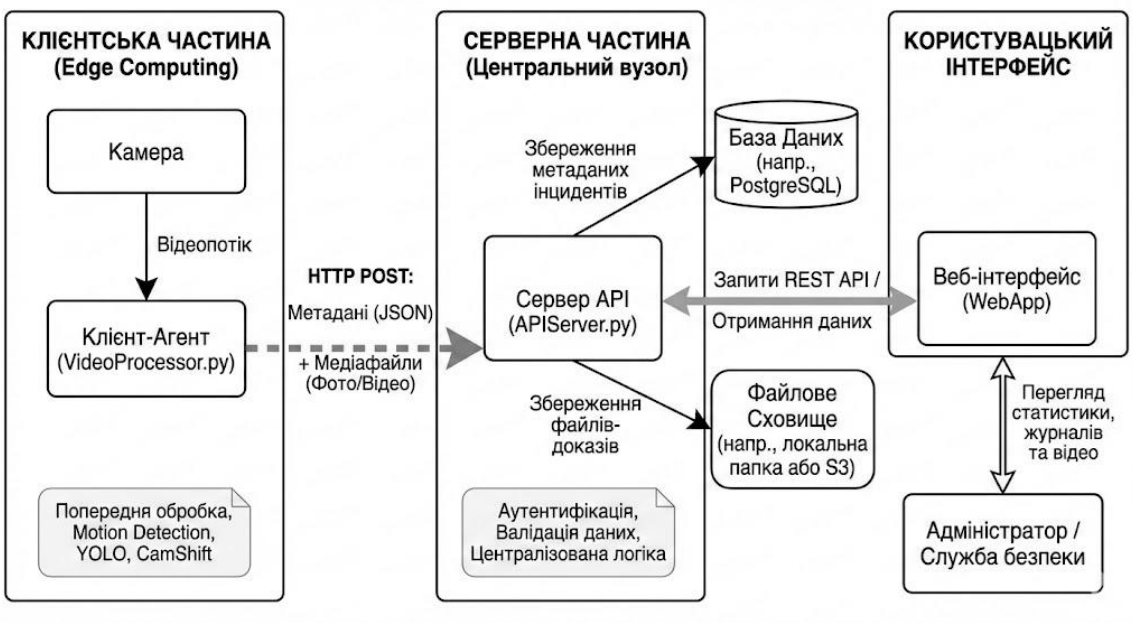

Рисунок В.1 — Загальна архітектура системи моніторингу

ДОДАТОК Г

Моделювання потоків даних (DFD)

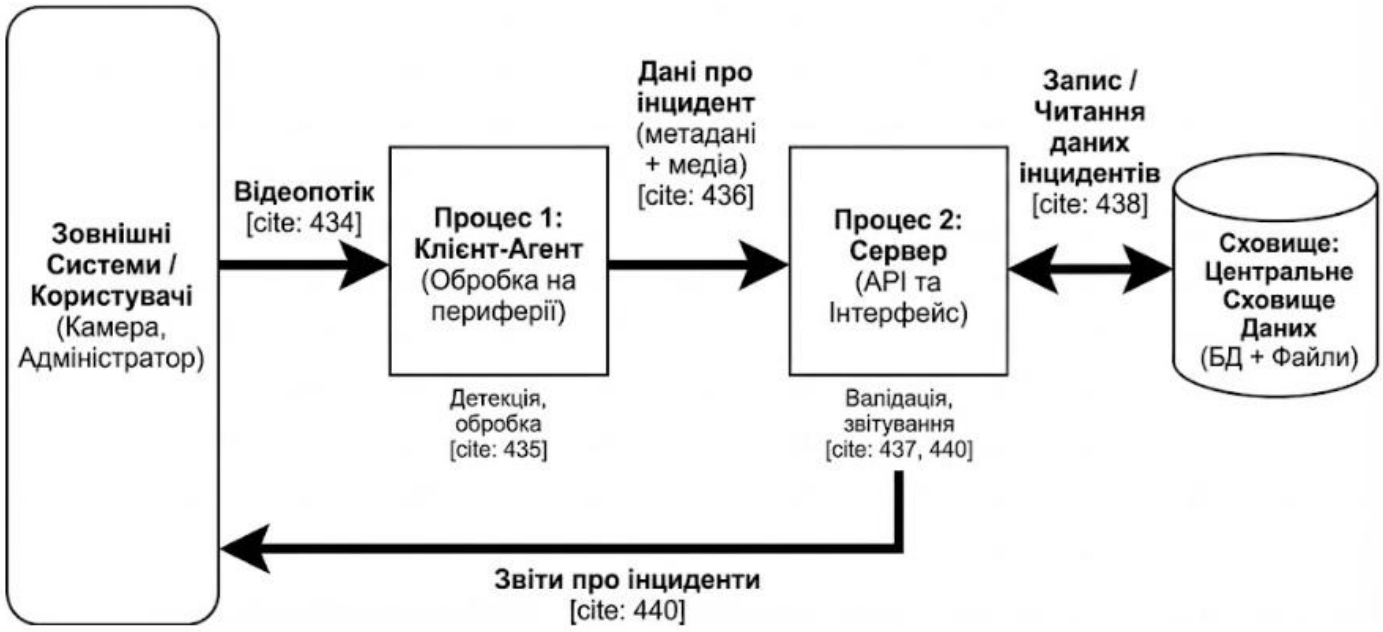

Рисунок Г.1 — моделювання потоків даних (DFD) в системі

ДОДАТОК Д

Діаграмма компонентів

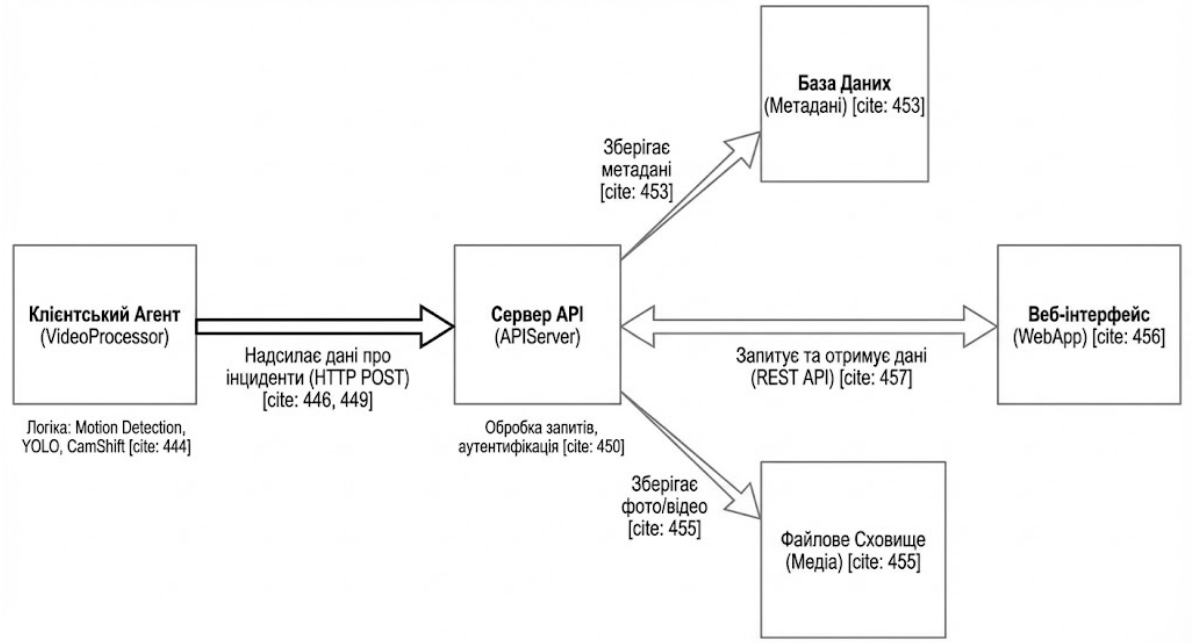

Рисунок Д.1 — діаграмма компонентів

ДОДАТОК Е

Приклади роботи аналізу відеопотоку

Рисунок Е.1 — Приклад роботи методу аналізу відеопотоку Motion Detection

Рисунок Е.2 — приклад роботи методу аналізу відеопотоку cvMeanShift

Рисунок Е.3 — приклад роботи методу аналізу відеопотоку YOLO

ДОДАТОК Ж

Лістинг коду

Код клієнта:

```
import cv2
 import numpy as np
import time
import json
 import psutil
import logging
 import datetime from threading import Thread
class PerformanceMonitor: def init(self): self.start_time = 0
def start_frame(self):
    # Початок відліку часу для поточного кадру
    self.start_time = time.perf_counter()

def end_frame(self, module_status):
    # Розрахунок часу обробки (latency
    processing_time = (time.perf_counter() — self.start_time) * 1000
    # Отримання поточного завантаження системи
    cpu_usage = psutil.cpu_percent()
    # ram_usage = psutil.virtual_memory().percent (можна додати за потребою)

    # Логування результатів (вивід у консоль для демонстрації)
    # logging.info(f"Status: {module_status} | Time: {processing_time:.2f}ms | CPU:
{cpu_usage}%")
    return processing_time
```

class ClientAgent: def **init**(self, config_path="config.json"): # У реальному застосуванні: with open(config_path) as f: self.config = json.load(f) self.config = { "agent_id": "CAM_01_WORKSHOP_A", "camera_stream_url": 0, # 0 для веб—камери, або RTSP посилання "server_api_endpoint": "http://localhost:8000/api/v1/alert", "confidence_threshold": 0.5, "restricted_zones": [{"zone_name": "red_zone", "coordinates": [[100, 100], [100, 400], [400, 400], [400, 100]]}] } self.monitor = PerformanceMonitor()
self.cap = cv2.VideoCapture(self.config["camera_stream_url"])

Завантаження YOLO

self.net = cv2.dnn.readNet("yolov4—tiny.weights", "yolov4—tiny.cfg")

self.layer_names = self.net.getLayerNames()

self.output_layers = [self.layer_names[i — 1] for i in self.net.getUnconnectedOutLayers()]

except:

print("Помилка: Файли YOLO не знайдені. Переконайтеся, що .weights та .cfg існують.")

self.net = None

Змінні стану

self.prev_frame = None

self.tracking_active = False

self.track_window = None

self.roi_hist = None

self.term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

def detect_motion(self, frame):

```
    """Метод Motion Detection згідно опису [cite: 275—280]"""
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    if self.prev_frame is None:
        self.prev_frame = gray
        return False

    # Обчислення різниці кадрів [cite: 276—277]
    frame_delta = cv2.absdiff(self.prev_frame, gray)
    thresh = cv2.threshold(frame_delta, 25, 255, cv2.THRESH_BINARY)[1]
    thresh = cv2.dilate(thresh, None, iterations=2)

    # Пошук контурів [cite: 280]
    contours, _ = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    motion_detected = False
    for c in contours:
        if cv2.contourArea(c) > 500: # Поріг площі
            motion_detected = True
            break

    self.prev_frame = gray
    return motion_detected

def run_yolo(self, frame):
    """Запуск нейромережі для детекції"""
```

```
    height, width, _ = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    self.net.setInput(blob)
    outs = self.net.forward(self.output_layers)

    class_ids = []
    confidences = []
    boxes = []
    conf_threshold = self.config["confidence_threshold"]

        for output in outs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            # class_id == 0 зазвичай це 'person' у COCO датасеті
            if class_id == 0 and confidence > conf_threshold:
                # Масштабування координат рамки
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Координати верхнього лівого кута для OpenCV
                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)

    # Non—maximum suppression для видалення дублікатів [cite: 356]
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    if len(indexes) > 0:
        # Повертаємо координати першого знайденого об'єкта для ініціалізації трекера
        i = indexes.flatten()[0]
        return boxes[i] # (x, y, w, h)
    return None

def initialize_tracker(self, frame, box):
    """Ініціалізація трекера MeanShift """
    x, y, w, h = box
    # Виділення ROI (Region of Interest)
    roi = frame[y:y+h, x:x+w]

        hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
    # Маска для відсіювання слабкого світла та шумів
    mask = cv2.inRange(hsv_roi, np.array((0., 60., 32.)), np.array((180., 255., 255.)))
    roi_hist = cv2.calcHist([hsv_roi], [0], mask, [180], [0, 180])
    cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)

    self.roi_hist = roi_hist
```

```
    self.track_window = (x, y, w, h)
    self.tracking_active = True

def check_zones(self, object_center):
    """Перевірка перетину заборонених зон [cite: 491]"""
    zones = self.config["restricted_zones"]
    for zone in zones:
        pts = np.array(zone["coordinates"], np.int32)
        pts = pts.reshape((—1, 1, 2))
        # pointPolygonTest: >0 (всередині), =0 (на межі), <0 (зовні)
        result = cv2.pointPolygonTest(pts, object_center, False)
        if result >= 0:
            return zone["zone_name"]
    return None

def send_alert(self, zone_name):
    """Формування JSON об'єкта та відправка [cite: 502—503]"""

    alert_data = {
        "timestamp_utc": datetime.datetime.utcnow().isoformat() + "Z",
        "agent_id": self.config["agent_id"],
        "event_type": "RESTRICTED_ZONE_ENTRY",
        "zone_name": zone_name,
        "media_files": ["snapshot_placeholder.jpg"] # Тут має бути логіка збереження
файлу
    }
    print(f"\n[ALERT] ВІДПРАВКА НА СЕРВЕР: {json.dumps(alert_data,
indent=2)}")
```

```
    # requests.post(self.config["server_api_endpoint"], json=alert_data)

def run(self):
    print("Запуск клієнтського агента...")
    while True:
        self.monitor.start_frame()
        ret, frame = self.cap.read()
        if not ret:
            break

        status = "IDLE"
        frame_display = frame.copy()

        # 1. Етап: Motion Detection (Тригер)
        if not self.tracking_active:
            if self.detect_motion(frame):
                status = "MOTION DETECTED"
                # Якщо є рух, запускаємо YOLO
                if self.net:
                    box = self.run_yolo(frame)
                    if box:
                        status = "YOLO DETECTED"
                        self.initialize_tracker(frame, box)


        if self.tracking_active:
            status = "TRACKING (CamShift)"
            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

```
        dst = cv2.calcBackProject([hsv], [0], self.roi_hist, [0, 180], 1)

                ret_cam, self.track_window = cv2.CamShift(dst, self.track_window,
self.term_crit)

        # Малювання результату CamShift
        pts = cv2.boxPoints(ret_cam)
        pts = np.int0(pts)
        cv2.polylines(frame_display, [pts], True, (0, 255, 0), 2)

        # Перевірка зон
        center_x = int(ret_cam[0][0])
        center_y = int(ret_cam[0][1])
        zone_violation = self.check_zones((center_x, center_y))

        if zone_violation:
            cv2.putText(frame_display, f"ALERT: {zone_violation}", (10, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            self.send_alert(zone_violation)

        # Умова скидання трекера

YOLO [cite: 589]
        if self.track_window[2] < 10 or self.track_window[3] < 10:
            self.tracking_active = False

    # Відображення зон
    for zone in self.config["restricted_zones"]:
```

```
        pts = np.array(zone["coordinates"], np.int32).reshape((—1, 1, 2))
        cv2.polylines(frame_display, [pts], True, (0, 0, 255), 2)
    self.monitor.end_frame(status)

    cv2.imshow("Client Agent Monitor", frame_display)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

self.cap.release()
cv2.destroyAllWindows()

    if name == "main": agent = ClientAgent() agent.run()
```

Код серверу:

```
import os
import shutil
import json
import datetime
from typing import List, Optional
from fastapi import FastAPI, UploadFile, File, Form, HTTPException, Header, Depends, status
from pydantic import BaseModel from sqlalchemy import create_engine, Column, Integer, String, DateTime, Text
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, Session
DATABASE_URL = "sqlite:///./workshop_monitoring.db" # MEDIA_STORAGE_PATH = "./media_storage/" API_KEY_SECRET = "a_secure_api_key_for_this_agent" # Має співпадати з config.json клієнта
os.makedirs(MEDIA_STORAGE_PATH, exist_ok=True)

engine = create_engine(DATABASE_URL, connect_args={"check_same_thread": False}) SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine) Base = declarative_base()

class Incident(Base):   tablename = "incidents"

id = Column(Integer, primary_key=True, index=True) # serial
timestamp_utc = Column(DateTime, nullable=False)   # timestamptz
agent_id = Column(String(100), nullable=False)     # varchar(100)
event_type = Column(String(50), nullable=False)    # varchar(50)
zone_name = Column(String(100), nullable=False)    # varchar(100)
```

```
snapshot_path = Column(Text, nullable=True)       # text
video_path = Column(Text, nullable=True)          # text


    Base.metadata.create_all(bind=engine)

    class AlertMetadata(BaseModel): timestamp_utc: str agent_id: str event_type: str zone_name: str # media_files

    app = FastAPI(title="Monitoring System API")

    def get_db(): db = SessionLocal() try: yield db finally: db.close()

    async def verify_api_key(x_api_key: Optional[str] = Header(None)): if x_api_key != API_KEY_SECRET: raise HTTPException( status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid API Key", ) return x_api_key

    @app.post("/api/v1/alert", status_code=status.HTTP_201_CREATED) async def receive_alert( metadata_json: str = Form(...), files: List[UploadFile] = File(None), db: Session = Depends(get_db), api_key: str = Depends(verify_api_key) ):


try:
   # Парсинг часу
   timestamp = datetime.datetime.fromisoformat(alert_data.timestamp_utc.replace('Z', '+00:00'))
except ValueError:
   timestamp = datetime.datetime.utcnow()


new_incident = Incident(
```

```
    timestamp_utc=timestamp,
    agent_id=alert_data.agent_id,
    event_type=alert_data.event_type,
    zone_name=alert_data.zone_name,
    snapshot_path=saved_snapshot_path,
    video_path=saved_video_path
)

db.add(new_incident)
db.commit()
db.refresh(new_incident)

# 4. Відповідь серверу [cite: 527]
return {
    "status": "success",
    "incident_id": new_incident.id,
    "message": "Incident registered successfully"
}
```

@app.get("/api/v1/alerts") def get_alerts(skip: int = 0, limit: int = 100, db: Session = Depends(get_db)): [cite: 528]. """ incidents = db.query(Incident).order_by(Incident.timestamp_utc.desc()).offset(skip).limit(limit).all() return incidents

if **name** == "**main**": import uvicorn # Запуск сервера uvicorn.run(app, host="0.0.0.0", port=8000)