Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ВИЗНАЧЕННЯ КІЛЬКОСТІ ВІДВІДУВАЧІВ НА ОСНОВІ ТЕХНОЛОГІЇ ТРЕКІНГУ З ІНТЕГРАЦІЄЮ В СЕРВІС ВІДЕОСПОСТЕРЕЖЕННЯ ISPY
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1
Ісаєв Є.А.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ісаєву Євгенію Антоновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для визначення кількості відвідувачів на основі технології трекінгу з інтеграцією в сервіс відеоспостереження iSpy

затверджена наказом університету від 20 травня 2024 року № 464 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, NumPy, фреймворк Ultralytics, мова програмування Python, Java, фреймворк Spring Boot, база даних MySQL, сервіс Celonis, застосунок для камер відеоспостереження iSpy.

4. Перелік питань, що потрібно опрацювати в роботі

1. Вивчити можливості існуючих застосунків для відеоспостереження.
2. Ознайомитися з методами детекції об'єктів та трекінгу.
3. Проаналізувати бібліотеки та фрейворки для вирішення задач детекції та трекінгу.
4. Вивчити особливості додавання нової функціональності для сервісу відеоспостереження iSpy.
5. Розробити алгоритм визначення кількості відвідувачів на основі технології трекінгу.
6. Розробити плагін, реалізувати інтеграцію з iSpy.
7. Провести тестування розробленого плагіну в реальных умовах, де в експериментах визначити моделі детекції та трекінгу, які якнайкраще відповідають задачі.
8. Налаштувати параметри алгоритму підрахунку людей.
9. Проаналізувати зібранні дані, що зберігаються під час входу та виходу людей із приміщення (відео, зафіксовані події в БД), сформувати звіти.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, порівняння систем відеоспостереження, постановка задачі, інтеграція із сервісом iSpy, різниця між детекцією та трекінгом об'єктів, еволюція методів трекінгу, Ultralytics YOLOv8, розробка архітектури та алгоритму плагіну, експерименти із моделями YOLOv8, експерименти із трекерами від Ultralytics, практичне застосування розробленого плагіну.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-10.04.24	
3	Аналіз літератури з досліджуваної проблеми	10.04.24-16.04.24	
4	Аналіз технічних засобів	16.04.24-28.04.24	
5	Розробка плагіну	28.04.24-08.05.24	
6	Програмна реалізація	08.05.24-21.05.24	
7	Оформлення пояснювальної записки	21.05.24-24.05.24	
8	Перевірка на плагіат	28.05.24	
9	Рецензування	29.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	03.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Яковлева О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 79 с., 2 табл., 52 рис., 37 джерел.

ЗАСТОСУНКИ ДЛЯ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ, ПЛАГІН, ДЕТЕКЦІЯ, ТРЕКІНГ, КІЛЬКІСТЬ ЛЮДЕЙ, ULTRALYTICS.

Об’єктом роботи є питання моніторингу наповненості приміщень людьми на основі відеоспостереження з використання трекінгових технологій.

Метою роботи є розробка застосунку для визначення кількості відвідувачів, на основі технології трекінгу, з інтеграцією в сервіс відеоспостереження iSpy.

В роботі проведено огляд методів та аналіз фреймворків для реалізації задач детекції та трекінгу об’єктів; розроблено алгоритм для визначення входу та виходу людини із приміщення; спроектовано архітектуру плагіна. У результаті роботи здійснена програмна реалізація застосунку для визначення кількості відвідувачів, який використовує технології трекінгу, та інтегрується із сервісом iSpy. Даний застосунок може застосовуватися для визначення наповненості приміщень людьми в різних сферах, наприклад, для торгівельних залів, офісів, студентських аудиторій та інші.

APPLICATIONS FOR CCTV CAMERAS, PLUGIN, DETECTION, TRACKING, NUMBER OF PEOPLE, ULTRALYTICS.

The object of work is the issue of monitoring the occupancy of premises by people on the basis of video surveillance using tracking technologies.

The purpose of the work is to develop an application for determining the number of visitors based on tracking technology with integration into the iSpy video surveillance service.

The paper provides an overview of methods and analysis of frameworks for implementing object detection and tracking tasks; an algorithm for determining the entry and exit of a person from the room is developed; the plug-in architecture is designed. As a result of the work, the software implementation of the application for determining the number of visitors, which uses tracking technologies and integrates with the iSpy service, was carried out. This application can be used to determine the occupancy of premises in various areas, such as trading floors, offices, student classrooms, etc. As a result of the work, the software implementation of the application for determining the number of visitors, which uses tracking technologies and integrates with the iSpy service, was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

GPL – General Public License

HSL – Hure, Saturation, and Lightness

SVM – Support vector machine

NLP – Natural language processing

MDNet – Multi-Domain Net

MOTA – Multiple Object Tracking Accuracy

HOTA – Higher Order Tracking Accuracy

ID – Identification

IDE – Integrated Development Environment

ВСТУП

У сучасному світі, використання камер відеоспостереження стало дуже важливим кроком у забезпеченні безпеки, контролю, та можливості аналізу середовища, для ефективного управління ним. Із швидким розвитком технологій, можливості виконання зазначених завдань стало набагато ефективнішими, дешевшими, та навіть технологічнішими, ніж це було десять років тому.

Раніше людство не мало можливості використовувати таку велику кількість камер, через їх вартість. Також, якість відео, яке створювалося за допомогою цих камер, було значно гірше, ніж сьогодні. Не існувало дешевих хмарних сервісів, які забезпечували б надійність системи відеоспостереження. Також, через вартість та обмежену потужність обладнання для обробки інформації, технології у сфері штучного інтелекту, зокрема комп'ютерного зору, не досягали таких інноваційних відкриттів, як сьогодні [1].

На сьогоднішній день системи камер відеоспостережень мають великий потенціал у сфері комп'ютерного зору, який можливо використати, для забезпечення більш ефективного, та якісного виконання задач безпеки, контролю, та аналітики. Системи камер відеоспостережень створюють достатню якість відео, яка необхідна для використання у технологіях комп'ютерного зору.

За останні п'ять років, технологія комп'ютерного зору отримала значний розвиток. На сьогоднішній день комп'ютерний зір здатен виконувати різноманітні задачі, зокрема: розпізнавання об'єктів, сегментація зображень, відстеження об'єктів, визначення позиції та орієнтації, розпізнавання жестів та емоцій, генерація зображень [2]. Це лише деякі приклади задач, які виконує штучний інтелект у сфері комп'ютерного зору. За допомогою навчання, та використання нейронних мереж, штучний

інтелект продовжує розвиватися, стаючи більш ефективнім у вирішенні складних задач у сфері обробки зображень.

Нажаль, системи камер відеоспостереження не використовують сучасні технології комп'ютерного зору, зокрема технологію відстеження об'єктів, лише в одному додатку було знайдено можливість відстеження об'єктів, але нажаль, технологія відстеження об'єкту була застаріла.

Технологія відстеження об'єктів – це процес автоматичного визначення та відстеження об'єкту та його положення на відео. Ця технологія має власні алгоритми, які використовує для відстеження об'єктів. На ранніх етапах створення цієї технології, основним алгоритмом відстеження об'єктів був оптичний метод, який визначав зміну яркості зображення. На сьогоднішній день, технологія відстеження об'єктів використовує комбінований метод, який включає в себе комбінацію різних алгоритмів для забезпечення поставленої задачі.

Застосунок iSpy [3] – це найпопулярніший застосунок для систем камер відеоспостережень, він має велику кількість можливостей, це – хмарні технології зберігання даних, розпізнавання руху об'єктів та налаштування розпізнавання, зручний інтерфейс застосунку, відкритий вихідний код, що сприяє формуванню активної спільноти користувачів, які обмінюються досвідом, створюють нові плагіни для застосунку. Даний застосунок має плагін для забезпечення можливості відстеження об'єктів, але нажаль він не використовує сучасні технології відстеження об'єктів.

Дана робота присвячена вирішенню питання розроблення застосунку для визначення кількості відвідувачів на основі технології трекінгу з інтеграцією в сервіс відеоспостереження iSpy.

1 АНАЛІЗ ІСНУЮЧИХ ЗАДАЧ ТА ЇХ РІШЕНЬ НА ОСНОВІ АНАЛІЗУ ВІДЕО З КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ

1.1 Задачі, які вирішуються за допомогою відеоспостереження

На сьогоднішній день, застосування камер відеоспостереження є досить поширеним явищем. Відеоспостереження вирішує багато задач, завдяки яким власники приватних підприємств, або будь-якої іншої будівлі використовують камери відеоспостереження [4]. Серед таких задач, є:

– відстеження та моніторинг. Камери відеоспостереження забезпечують можливість моніторингу певних площ, об’єктів та місць, з метою відстеження наданих системі об’єктів;

– забезпечення безпеки. Камери відеоспостереження забезпечують нагляд за певними територіями, які потребують більш ретельного нагляду, забезпечуючи безпеку, запобігаючи злочинній діяльності, вчиненні крадіжок, або інших незаконних дій;

– підвищення продуктивності працівників. Працівники схильні підвищувати свою продуктивність, якщо за ними стежать камери відеоспостереження;

– контроль дотримання правил і норм поведінки. Камери, які встановлені на дорогах забезпечують дотримання правил поведінки водіїв та пішоходів на дорогах. Камери, які встановлені у місцях громадського призначення забезпечують дотримання правил і норм поведінки відвідувачів і персоналу закладу;

– виявлення випадків аварій. Відеоспостереження забезпечує виявлення автомобільних аварій, падіння людини, аварій на підприємствах, або інших випадків надзвичайних ситуацій. Також, камери можуть надсилати відповідну допомогу при виявленні надзвичайної ситуації;

– розслідування злочинів. Наявність записів з відеокамер можуть бути

важливим доказом у розслідуванні злочинів;

– моніторинг довкілля. Камери відеоспостереження можуть використовуватися для моніторингу стану довкілля, виявляючи пожежу, надлишкові викиди підприємств, виявлення змін у кількості природних ресурсів;

– організація ефективного робочого простору. Камери, встановлені у робочих зонах підприємств, можуть бути використаними для підрахунку ефективності використання робочого простору, виявлення скупчення людей, та запобігання можливих ризиків, пов’язаних з неефективним використанням робочого простору;

– моніторинг входів і виходів. Камери відеоспостереження фіксують людей, які зайшли та вийшли із будівлі, дозволяючи використовувати цю інформацію у майбутньому. Наприклад, під час надзвичайної ситуації, за допомогою камер відеоспостереження, рятувальним службам буде надано можливість швидко визначити кількість осіб, які залишилися у приміщенні. А також, ідентифікувати їх, використовуючи додаткові технології, які використовуються у системах відеоспостереження.

1.2 Аналіз готових рішень для відеоспостереження

На сьогоднішній день існує велика кількість програмних застосунків, призначених для роботи з камерами відеоспостереження. Вони мають власні переваги та недоліки. В даній роботі необхідно проаналізувати популярні застосунки, котрі мають достатньо велику аудиторію користувачів, мають стабільну версію, надійне зберігання відеозаписів, не потребують додаткових коштів для встановлення, або використання.

Системи для камер відеоспостереження класифікуються за наявністю безкоштовної версії застосунку, наявністю штучного інтелекту для детекції,

наявністю хмарного збереження даних, наявністю відкритого вихідного коду. Нижче (табл. 1.1) наведено порівняльна таблиця для різних систем відеоспостереження.

Таблиця 1.1 – порівняння систем відеоспостереження

Назва застосунку	**Штучний інтелект**	**Хмарне збереження даних**	**Open Source**	**Безкоштовна версія, яка включає в себе штучний інтелект**
Xeoma	Так	Так	Ні	Ні
EyeLine Video Surveillance	Так	Так	Частково	Так (обмежений функціонал)
iSpy	Так	Так	Так	Ні
CamHi	Ні	Так	Частково	Ні
CloudEyes	Так	Так	Частково	Ні
AtHome Video Streamer	Так	Так	Ні	Ні

Xeoma [5] – це популярний застосунок для надання можливості відеозаписів з різних типів камер відеоспостережень. Працює на всіх популярних операційних системах. Має простий інтерфейс (рис. 1.1), гнучкі налаштування, технічну підтримку з боку розробників застосунку. Даний застосунок має великі аналітичні можливості, на основі штучного інтелекту, такі як:

- розпізнавання номерних знаків, обличь, емоцій, віку, статі;
- розпізнавання класів об’єктів і звуків;
- виявлення медичних масок, падіння людини, виявлення натовпу, підрахунок відвідувачів, теплова карта, тощо.

Рисунок 1.1 – Розпізнавання об’єкту за допомогою Xeoma [6]

Серед недоліків застосунку Xeoma, варто виділити відсутність безкоштовної версії, в якій можливість застосовувати штучний інтелект була увімкнена. Також, не варто розраховувати на моделі, які запроваджує компанія Xeoma, дана компанія не запровадила метрики, за якими користувачі могли б порівняти моделі із аналогами. Це свідчить, що майбутні клієнти повинні довіряти знятим відеороликам, та відгукам користувачів, які опубліковані на офіційному сайті Xeoma. На жаль, застосунок Xeoma не має безкоштовної версії.

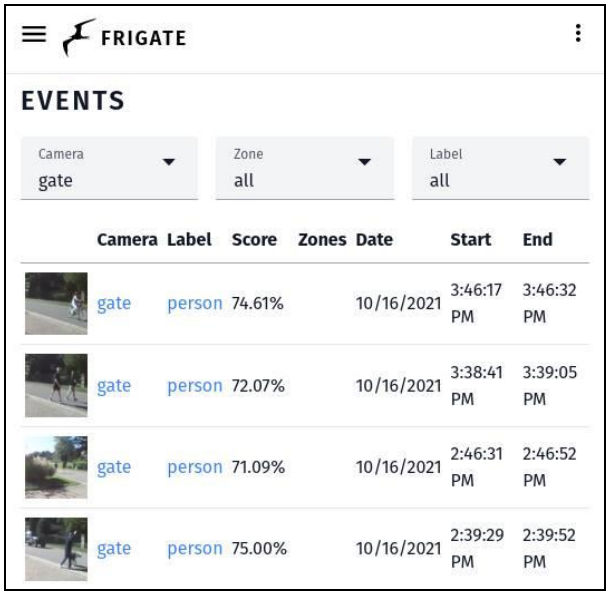


Рисунок 1.2 – Розпізнавання об’єктів за допомогою Frigate [7]

Інший варіант застосунку для камер відеоспостереження – це Frigate [7]. Даний продукт розроблений для систем Android, IOS та Windows. Frigate, підтримує різні типи відеокамер, також, він може інтегруватися з різними сервісами, наприклад – інтеграція з сервісом домашнього асистенту.

Даний застосунок (рис. 1.2) інтегрується безпосередньо в браузер, забезпечує низьку затримку камери, а також відкриває датчики та перемикачі в реальному часі, щоб увімкнути автоматизацію та сповіщення на ваш розсуд. Frigate відстежує об’єкти в режимі реального часу і може визначити точний момент, коли людина починає підніматися по сходах або коли автомобіль заїжджає на під’їзну доріжку.

Даний застосунок може вести 100 виявлень кадрів за секунду, не пропускаючи жодного кадру. Даний застосунок використовує Google Coral TPU для забезпечення функції виявлення, саме через це він є досить швидким та легким, аби його змогло підтримувати навіть слабкі пристрої. Завдяки локальній обробці даних Frigate, платити за відправку відеозаписів з камери в хмару для аналізу не потрібно.

Даний застосунок має обмежений функціонал, який тільки зосереджений на відправці повідомлень користувачам у випадку виявлення певної події, яку користувач задав заздалегідь. Ця проблема може бути вирішена за допомогою інтеграції сторонньої програми, яка написана з метою збільшення функціоналу застосунку Frigate. Через те, що даний застосунок має відкритий вихідний код, існує можливість інтеграції та написання програмного коду всередині Frigate, або навіть окремого застосунку, який розширить функціонал даного застосунку.

Серед значних недоліків даного застосунку, варто відмітити платну версію підписки, яка активує можливості штучного інтелекту. Даний застосунок не надає можливості перегляду записаних відео за допомогою користувацького інтерфейсу. Також, даний продукт знаходиться на стадії розробки, що свідчить про велику вірогідність неточностей у процесі роботи застосунку. Нажаль, застосунок Frigate не підходить для виконання задачі

даної роботи.

iSpy – це один із найпопулярніших застосунків для камер відеоспостережень. Даний застосунок має ліцензію GPL (General Public License), що свідчить про те, що користувачі можуть завантажувати, використовувати, вивчати та змінювати даний застосунок безкоштовно. Варто зазначити, що незважаючи на те, що базова версія даного сервісу є безкоштовна, все одно необхідно платити за додаткові функції та послуги у рамках підписки на даний сервіс.

Платна підписка на сервіс iSpy включає в себе додаткові можливості, пов’язані із використанням штучного інтелекту, його гнучкої конфігурації, технічну підтримку зі сторони експертів сервісу iSpy, підтримку хмарного зберігання, SMS-сповіщення, та інший розширений функціонал, який вимагає оплачену підписку на сервіс iSpy.

Серед вагомих переваг сервісу iSpy варто відмітити підтримку даним сервісом різних типів камер, включаючи вебкамери, IP-камери, відео регістраторів та мікрофонів, камер ноутбуку, тощо. Підтримку збереження даних у хмарних сервісах, або на локальному сховищі. Застосунок iSpy має власний детектор руху, який надає можливість детекції руху перед камерою. Після того, як детектор виявив рух перед камерою, iSpy запустить процедуру сповіщення користувача. Процедуру сповіщення користувача можливо налаштувати під власні потреби, та змінити певні кроки. За замовчуванням процедура сповіщення буде автоматично запускати запис відео за останню хвилину, та відіграє звуковий сигнал, аби звернути увагу до себе, після запису відео, воно буде архівовано на локальному сховищі пристрою, або у хмарному сховищі, якщо оплачена підписка на сервіс. Також даний застосунок має зручний користувацький інтерфейс, за допомогою якого можливо легко розібратися із застосунком. На рисунку 1.3 представлено сторінку Timeline, за допомогою якої існує можливість передивитися усі існуючі записи відео за будь-який день.

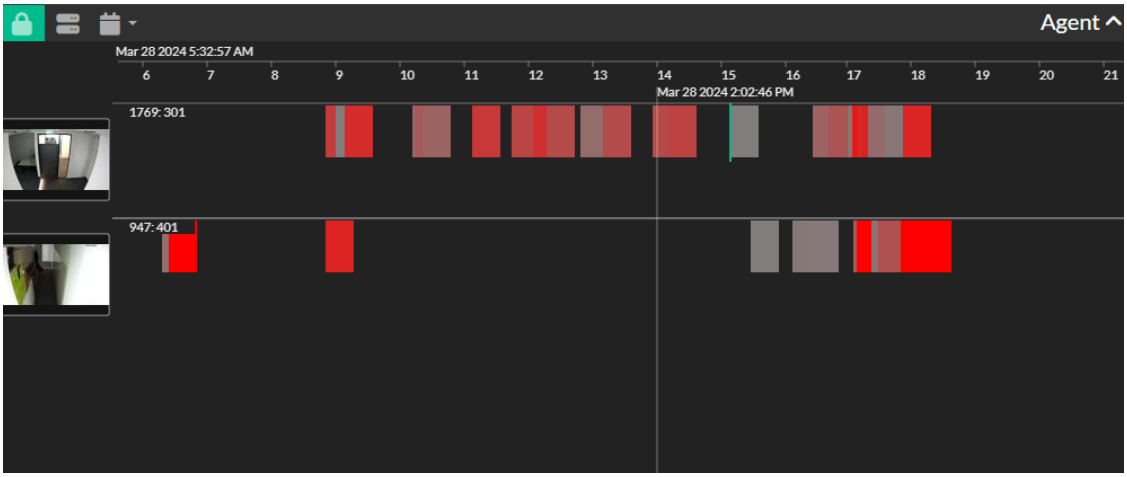


Рисунок 1.3 – Timeline у сервісі iSpy

Застосунок iSpy представляє собою сервіс, який дуже швидко і легко встановлюється на будь-яку систему. Після завершення процедури встановлення даний сервіс потребує авторизуватися, або зареєструватися для того, аби продовжити діяльність сервісу. Після успішної авторизації у даному сервісі, він буде доступний за посиланням localhost:8090. Після входу у даний сервіс необхідно буде додати камери, через кнопку «New Device», яка розташована ліворуч зверху. Після цього необхідно буде обрати один із доступних варіантів пристроїв, приклад наведено на рисунку 1.4.

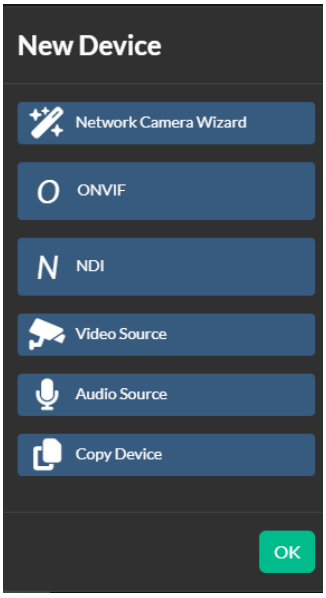


Рисунок 1.4 – Додавання нового пристрою у iSpy

Після успішного додавання нового пристрою у сервіс iSpy, він буде відображатися на основній сторінці, на рисунку 1.5 наведено приклад.

Рисунок 1.5 – Головна сторінка iSpy

Застосунок iSpy має різні вбудовані детектори руху. Кожен із детекторів руху має різні характеристики і підходить для різних цілей. Нижче наведено перелік видів детекторів руху iSpy:

– custom Frame (користувацький кадр) – застосунок iSpy збереже один кадр у власній пам’яті та буде порівнювати його з наступними кадрами;

– background Modelling (моделювання фону) – стосунок iSpy збереже один кадр у власній пам’яті та з часом буде адаптувати його, аби він був змінений до поточного кадру. Це необхідно робити, якщо існує сцена, де постійно щось рухається. Із часом, модель iSpy це вивчить, та буде ігнорувати це;

– two Frames (два кадри) – порівняння останнього кадру із поточним, найпопулярніший тип детекторів руху, який використовується більшістю користувачів;

– none (відсутній) – відсутність виявлення руху, використовується для простого запису відео, без детекції руху.

Параметри детекції руху у застосунку iSpy також можливо змінювати за допомогою зміни параметру діапазону спрацювання. Він визначає максимальні та мінімальні рівні спрацювання детектора руху. Як видно

на (рис. 1.6), камера надає можливість прямого спостереження за обсягом руху, виявленого в момент часу, разом з точними координатами точок спрацювання. Функція контролю дозволяє встановлювати у відсотках мінімальне та максимальне значення для спрацювання. Максимальне значення параметру діапазону спрацювання надає можливості ігнорувати глобальні зміни у сцені, наприклад зміна яскравості у сцені. Даний параметр необхідно налаштовувати, його налаштування залежить від багатьох параметрів, таких як: камера, модель детекції, сцена і сценарії запису. Налаштування даного параметру потребує багато часу, аби правильно підібрати параметри під певний сценарій застосування. Нажаль, даний параметр доступний лише у платній версії застосунку, та використовує застарілі технології детекції, які базуються лише на змінах яскравості зображення.

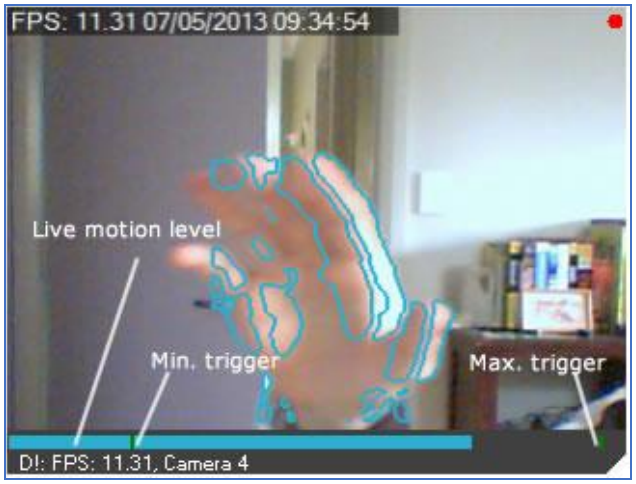


Рисунок 1.6 – Тестування параметру діапазону спрацювання [8]

Застосунок iSpy має ще один параметр, який впливає на детекцію руху – це параметр фільтрації HSL (відтінок, насиченість та яскравість зображення). Параметр HSL дозволяє детектору руху ігнорувати визначені діапазони яскравості та кольорів при виконанні задачі пошуку руху. Даний параметр може бути корисним при налаштуванні детектора руху під певний сценарій, наприклад: виявлення диму, або вогню у приміщенні, детекція

об’єктів певного кольору, ігнорування об’єктів, які постійно рухаються (хмари, або дерева). Ліворуч, на рисунку нижче (рис. 1.7) зображено колірне коло, яке визначає який діапазон кольорів детектор буде ігнорувати. Також, там можливо визначити максимальна та мінімальну насиченість та яскравість зображення. Та праворуч, на рисунку нижче (рис. 1.7) відображається попереднє тестування даного параметру. Також, слід зазначити, що параметр HSL сильно впливає на обробку зображення, тому варто звернути увагу на потужність пристрою, на якому запущено сервіс iSpy. Нажаль, даний параметр також доступний лише за підпискою на сервіс iSpy, а також, він не підтримується у нових версіях цього застосунку. Даний параметр занадто сильно впливає на швидкість обробки зображення, що значно сповільнює роботу детектора, нажаль, параметр HSL не підходить для застосування у даній роботі.

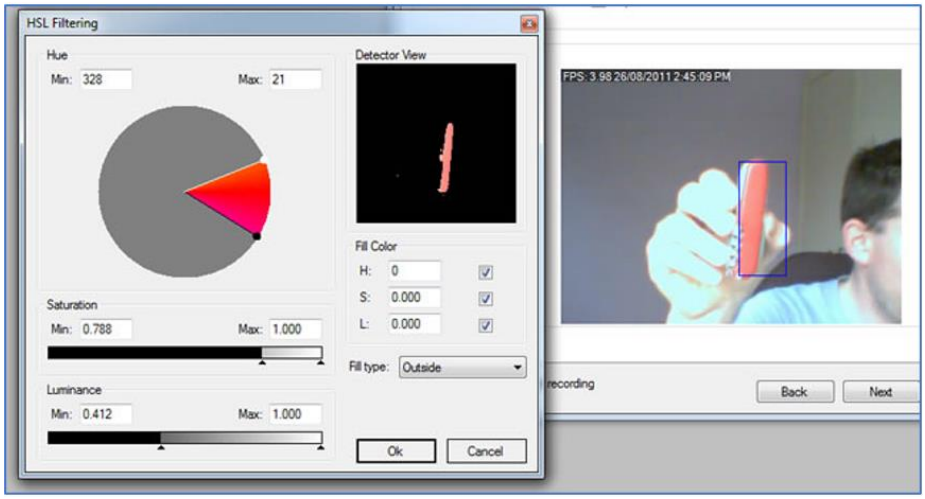


Рисунок 1.7 – Тестування змін у параметрі HSL у застосунку iSpy [8]

Ще однією корисною функцією для відстеження руху в застосунку iSpy є виявлення зон, приклад якої зображено нижче (рис. 1.8). За допомогою цієї функції можна встановити певні області у полі зору камери для відстеження, ігноруючи інші області. Користувач може додати будь-яку кількість прямокутних зон виявлення, і iSpy буде стежити лише за рухом в межах цих зон. Для додавання нової зони необхідно натиснути на напівпрозорий

прямокутник і перетягнути його. Щоб очистити всі зони, слід натиснути кнопку «Очистити зони». Якщо ж жодні зони не визначені, iSpy буде відстежувати рух по всьому полі зору камери.

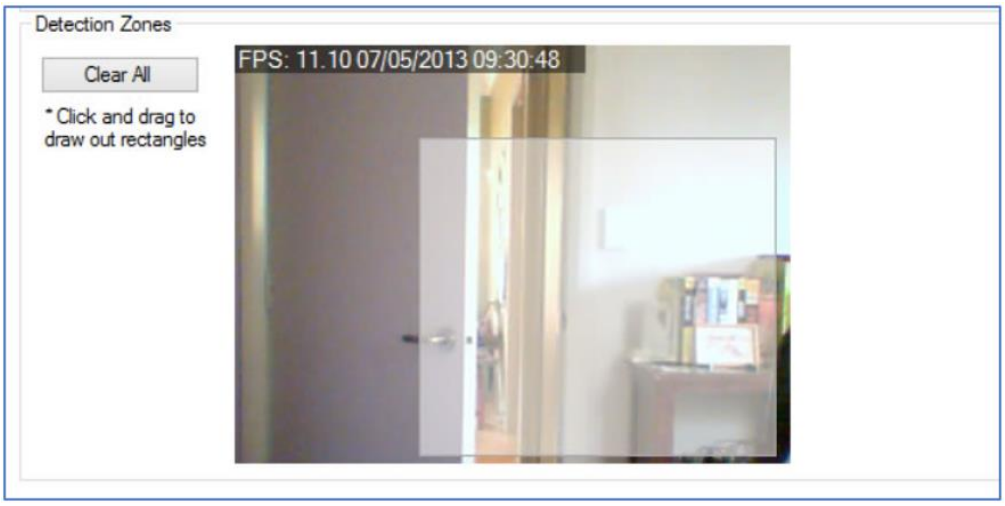


Рисунок 1.8 – Тестування зон виявлення у застосунку iSpy [8]

Нажаль, налаштування зон виявлення прив’язується до внутрішнього детектора руху застосунку iSpy, та входить до платної версії підписки до застосунку. Налаштування зон виявлення не підходить до задачі даної роботи, бо присутня лише у платній версії застосунку та використовуватися не буде.

Застосунок iSpy має безліч переваг, але нажаль, ключовими недоліками даного застосунку є відсутність конфігурації детекції об’єктів у безкоштовній версії застосунку, використання застарілих алгоритмів детекції об’єктів, які вже не використовуються у сьогоденні. Але не дивлячись ні на що, даний застосунок має найбільшу перевагу – це GPL ліцензія, разом із відкритим вихідним кодом, що дозволяє розробникам покращувати даний застосунок, надаючи написанні користувацькі плагіни для сервісу iSpy.

Сервіс iSpy має велику спільноту користувачів, яка надає підтримку, обмінюючись досвідом, та запроваджуючи нові користувацькі плагіни для даного сервісу. Дана робота присвячена використанню трекінгу об’єктів, у

застосунках відеоспостереження, у ході дослідження користувацьких плагінів для застосунку iSpy, було знайдено один користувацький плагін для трекінгу об’єктів [9]. Але нажаль, він був написаний у 2019 році, та використовує застарілі технології детекції та трекінгу об’єктів, які вже не використовуються у сьогоденні.

Завдяки тому, що iSpy має відкритий вихідний код, даний застосунок надає можливість написати власний плагін, який буде використовувати сучасні методи детекції та трекінгу об’єктів, які дозволять даному сервісу виконувати задачі набагато ефективніше.

1.3 Досягнення Computer Vision

1.3.1 Прогрес в Computer Vision

Сфера комп’ютерного зору має багату історію, яка налічує кілька десятиліть, а її розвиток до сьогоднішнього передового стану був зумовлений ключовими технологічними досягненнями та науковими проривами. Нижче наведений огляд того, як комп’ютерний зір розвивався від своїх ранніх витоків до сучасного стану прогресу.

Ранні системи комп’ютерного зору стикалися зі значними проблемами через обмеженість обчислювальних ресурсів, низьку роздільну здатність зображень і складність інтерпретації візуальних сцен. Переважали ручні підходи на основі функцій, але вони мали обмеження в обробці варіабельності та масштабу.

Сфера комп’ютерного зору виникла в 1960-х роках завдяки новаторській роботі таких дослідників, як Ларрі Робертс, який розробив першу систему комп’ютерного зору, здатну інтерпретувати прості візуальні сцени. Перші роботи були зосереджені на таких завданнях, як виявлення границь, розпізнавання образів і розуміння зображень.

Протягом 1970-х і 1980-х років дослідники розробляли фундаментальні алгоритми для обробки та аналізу зображень, включаючи методи виділення ознак, зіставлення з шаблоном і оптичного розпізнавання символів (OCR). Значні досягнення включали розроблення перетворення Хафа для виявлення ліній і методу Лукаса-Канаде для оцінки оптичного потоку.

У 1990-х і 2000-х роках відбувся зсув у бік інтеграції методів машинного навчання в комп'ютерний зір. У дану епоху з'явилися статистичні методи, нейронні мережі та машини опорних векторів (SVM) для таких завдань, як розпізнавання об'єктів, розпізнавання облич та класифікація зображень.

Прорив у глибокому навчанні в 2010-х роках зробив революцію в комп'ютерному зорі. Завдяки великим наборам даних і прогресу в обчисленнях на графічних процесорах CNN досягли безпрецедентної точності в таких завданнях, як класифікація зображень, виявлення об'єктів і семантична сегментація [10]. Такі моделі, як AlexNet, VGGNet, ResNet та інші, оптимізовані для конкретних завдань – розширили межі продуктивності, наприклад: Faster R-CNN для виявлення об'єктів, U-Net для сегментації медичних зображень. Наявність величезних маркованих наборів даних, таких, як: ImageNet, COCO і поява методу навчання з перенесенням дозволила налаштувати попередньо навчені моделі для конкретних застосувань, полегшивши доступ до найсучасніших можливостей комп'ютерного зору.

На сьогоднішній день, комп'ютерний зір все більше інтегрується з обробкою природної мови (NLP) та обробкою аудіо для мультимодального розуміння. Дослідження зосереджені на підвищенні надійності, точності та зручності сприйняття інформації. Ведуться роботи з розробки систем комп'ютерного зору, здатних безперервно навчатися і адаптуватися до мінливого середовища з часом, імітуючи сприйняття і адаптацію, подібні до людських.

1.3.2 Задачі детекції та трекінгу об’єктів

Computer Vision має дві основні сфери застосування – це детекція та трекінг об’єктів. Ці два поняття, на перший погляд, дуже сході між собою, але вони кардинально відрізняються між собою.

Детекція об’єктів [11] – це автоматизований процес визначення та виявлення присутності об’єктів на зображенні. Алгоритми детекції проводять аналіз пікселів зображення, та виявляють області на зображенні, які підходять по певним критеріям, наприклад: форма, текстура, колір, та контекст.

Трекінг об’єктів – це автоматизований процес визначення та виявлення присутності об’єктів на відео протягом часу. Після того, як об’єкт було визначено, трекер буде слідкувати за ним на наступних кадрах, визначаючи на наступних кадрах, його нове положення на відео.

Отже, детекція і трекінг досить сильно пов’язані між собою їхньою природою, але вони виконують різні функції і мають різні призначення. Детекція об’єктів виконує задачі, пов’язанні із виявленням об’єктів, локалізацією об’єктів, класифікацією об’єктів, сегментацією об’єктів, коли трекінг об’єктів виконує наступні задачі: відстеження руху, оновлення положення, прогнозування шляху.

1.3.3 Камери відеоспостереження

Камери відеоспостереження необхідні для моніторингу, та створення запису відео в певних зонах інтересу. Існують велика кількість різних камер відеоспостереження, серед яких варто виділити [12]:

– бездротові камери – це камери, відеосигнал яких передається по радіохвилям, що забезпечує гнучке встановлення, та відсутня необхідність

прокладання дротів. Варто зазначити, що даний тип камер можуть мати обмеження у дальності та надійності;

– аналогові камери – це класичний приклад камер, відеосигнал яких передається через аналогові кабелі. Зазвичай, такий тип камер має доступну ціну, однак вони мають обмежене розширення та функціональність;

– PTZ камери – це тип камер, які мають змогу змінювати кут нахилу, змінювати своє положення, обертаючись навколо своєї осі, збільшувати масштаб зображення;

– IP-камери – це камери, які використовують мережу Інтернет для передачі відеосигналу. Зазвичай, даний тип камер є дорожчим, серед інших типів камер. IP-камери мають більш високу якість відео, більш великий кут огляду, а також мають унікальні опції, наприклад: нічне бачення, або двостороннє аудіо;

– HD-SDI – даний тип камер має змогу передавати відео високої якості, без необхідності використання мережі Інтернет;

– приховані камери – камери, що сховані у предметах, та необхідні для прихованого спостереження за об’єктом;

– пожежобезпечні камери, або камери IP66, та вище – це тип камер, які зроблені із спеціальних стійких матеріалів, завдяки яким, вони більш стійкі до пожеж.

1.4 Постановка задачі

Таким чином, дослідивши різні застосунки для використання камер відеоспостереження, стало зрозуміло, що жоден застосунок, який є безкоштовним, не має в своєму функціоналі сучасну детекцію та трекінг об’єктів. А застосунки для камер відеоспостереження, які все ж таки мають дані технології є платними та знаходяться у стадії розробки, що свідчить про

те, що використовувати застосунки, які ще не працюють стабільно не варто через те, що вони можуть згенерувати багато нових проблем.

Також, стало зрозуміло, що жоден із застосунків для використання камер відеоспостереження не мав можливості зберігати кількість людей, які зайшли, або вийшли із визначеної зони інтересу у базу даних, яку згодом, можливо було б використовувати для надання інформації про кількість людей за певний проміжок часу відповідальним особам. Ця задача є дуже корисною, наприклад, для служб порятунку, оскільки у разі пожежі вони могли б отримати детальну інформацію про кількість людей у будівлі та ідентифікувати їх за допомогою записів, зроблених за допомогою застосунків для камер відеоспостереження. Також, ця задача є корисною для аналітиків, які, за допомогою даної технології, використовували дану інформацію для запобігання створення натовпів у центрах масового обслуговування, коректної організації робочого простору у офісних будівлях, аби не навантажувати певну область приміщення, а розподілити навантаження за всією площею.

Отже, метою роботи є розробка плагіну для застосунку, призначеного для моніторингу камер відеоспостереження iSpy, який буде підраховувати кількість осіб, що увійшли, то вийшли із будівлі. Даний плагін буде надавати інформацію про кількість осіб, що знаходилися всередині будівлі за певний проміжок часу.

Для досягнення мети необхідно вирішити такі завдання:

– вивчити можливості існуючих застосунків для відеоспостереження;

– ознайомитися з методами детекції об’єктів та трекінгу;

– проаналізувати бібліотеки та фрейворки для вирішення задач детекції та трекінгу;

– вивчити особливості додавання нової функціональності для сервісу відеоспостереження iSpy;

– розробити алгоритм визначення кількості відвідувачів на основі технології трекінгу;

– спроектувати архітектуру плагіну для підрахунку людей у приміщенні та формування звітів;

– розробити плагін, реалізувати інтеграцію з iSpy;

– провести тестування розробленого плагіну в реальных умовах, де в експериментах визначити моделі детекції та трекінгу, які якнайкраще відповідають задачі;

– налаштувати параметри алгоритму підрахунку людей;

– проаналізувати зібранні дані, що зберігаються під час входу та виходу людей із приміщення (відео, зафіксовані події в БД), сформувати звіти.

2 ПРОЕКТУВАННЯ ПЛАГІНА ДЛЯ ВИЗНАЧЕННЯ КІЛЬКОСТІ ЛЮДЕЙ НА ОСНОВІ ВІДЕОСПОСТЕРЕЖЕННЯ

2.1 Методи трекінгу

2.1.1 Загальні характеристики трекерів

Відстеження об’єктів – це важлива технологія з безліччю застосувань, від управління складськими операціями до систем нагляду у дронах. Основний принцип полягає в пошуку та виявленні об’єктів, що може бути складним через їх різноманітність і змінність в зовнішньому вигляді.

Відстеження об’єктів – це фундаментальна задача комп’ютерного зору, яка передбачає безперервний моніторинг положення та траєкторії руху об’єктів у відеопослідовності.

Відстеження об’єктів відрізняється від виявлення об’єктів. Відстеження об’єктів і виявлення об’єктів – це дві різні концепції в комп’ютерному зорі. Під час виявлення об’єктів алгоритм спробує детермінувати об’єкти на конкретному кадрі або зображенні. У відстеженні об’єктів система спробує не лише виявити об’єкт, а й спрогнозувати його майбутнє положення шляхом аналізу його траєкторії на послідовних кадрах чи зображеннях.

Алгоритми відстеження об’єктів можна розділити на різні типи залежно від завдання та типу вхідних даних, на яких вони навчаються. Алгоритми відстеження можуть бути класифіковані за кількістю об’єктів, що вони відслідковують, та поділяються на:

– відстеження одного об’єкта [13] – передбачає відслідковування лише одного конкретного об’єкта протягом відео, або послідовності зображень. Початкові координати об’єкта та рамки встановлюються у першому кадрі чи зображенні, і після цього об’єкт розпізнається та відстежується у наступних кадрах чи зображеннях. Алгоритми відстеження одиночних об’єктів повинні бути здатні відстежувати будь-який об’єкт, який

вони виявлять, навіть якщо на ньому не було навчено жодної класифікаційної моделі. Нижче, на рисунку 2.1, представлено приклад відстеження одного об’єкта;

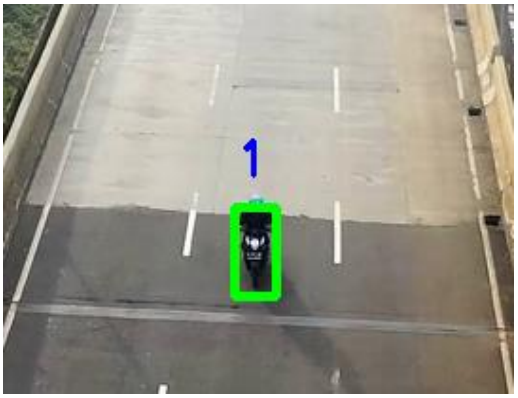


Рисунок 2.1 – Застосування Single Object Tracker

– відстеження декількох об’єктів [14] – передбачає відслідковування більш ніж одного об’єкта. Алгоритми відстеження спочатку визначають кількість об’єктів у кожному кадрі, а потім відстежують ідентичність кожного об’єкта від одного кадру до іншого. Нижче представлено застосування даного алгоритму (рис. 2.2).

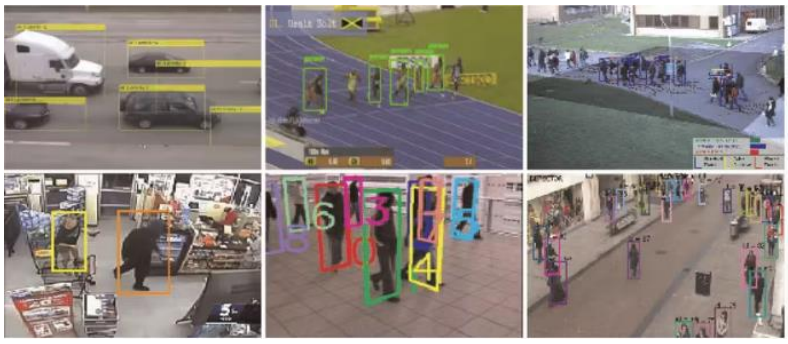


Рисунок 2.2 – Застосування алгоритму відстеження декількох об’єктів [15]

Історія створення трекерів налічує багато етапів та важливих моментів, що сприяли їхньому розвитку. Початок цієї історії можна віднести до пізнішого XX століття, коли виникла потреба в точному визначенні руху об’єктів на відеозаписах.

На ранніх етапах розвитку алгоритмів трекерів, вони використовувалися виключно на промисловості та наукових дослідженнях. Данні трекери базувалися на використанні примітивних алгоритмів виявлення та відстеження об’єктів, які рухалися на зображеннях. У ході виконання даної роботи, було власноруч створено приклад примітивного трекеру. Даний трекер накладав чорно-білу маску на певну область у відео, та визначав кількість чорних пікселів у даній області. Якщо кількість чорних пікселів була більша за задане значення, то даний алгоритм вважав, що знайшов рухомий об’єкт на кадрі, та починав відслідковувати даний об’єкт, визначаючи його наступні координати, та ідентифікуючи об’єкт. На рисунку нижче (рис. 2.3) представлено даний алгоритм. Нажаль, даний алгоритм не є ефективним через те, що він використовує примітивну технологію трекінгу. Зазвичай, трекер губить знайдений об’єкт, та трекер знову знаходить даний об’єкт, та призначає йому новий ідентифікатор. Також, зображення має знаходитися на статичній камері, яка не буде змінювати своє положення. Практично неможливо налагодити максимальну кількість чорних пікселів, після яких об’єкт розпочне відслідковуватись через те, що об’єкти можуть бути різних розмірів, та зі зближенням, вони також змінюють свій розмір.

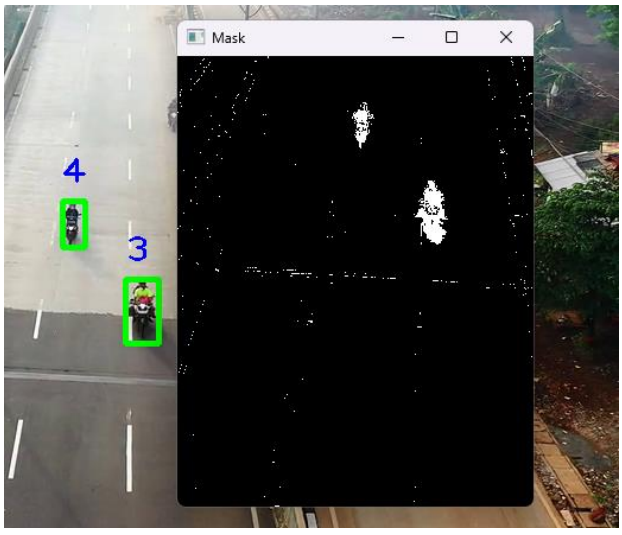


Рисунок 2.3 – Застосування примітивного алгоритма трекінгу

З розвитком комп’ютерної технології та штучного інтелекту у 1990-2000-х роках, трекери почали використовувати складніші алгоритми, що базувалися на методах машинного навчання та комп’ютерного зору [16]. Це дозволило покращити точність відстеження та зробило трекери більш автоматизованими.

У 2010-х роках з’явилися трекери, які використовували глибоке навчання, такі як згорткові нейронні мережі, щоб вдосконалити процес відстеження об’єктів [17]. Новостворені методи дозволили трекерам автоматично виявляти об’єкти на відео та навіть відстежувати їх у складних умовах, таких як зміна освітлення чи часткова оклюзія об’єктів.

На сьогоднішній день, трекери використовують комбінований метод відстеження об’єктів, вони використовують алгоритми глибокого навчання, алгоритми комп’ютерного зору разом [18], надаючи більш ефективну роботу трекерів. Кожного дня, трекери все більш розвиваються і у майбутньому будуть мати набагато більший розвиток, ніж сьогодні.

2.1.2 Класичний алгоритм відстеження об’єктів.

Технологія відстеження об’єктів має певний алгоритм, за яким вона працює:

Крок 1. Ініціалізація цілей – визначення кількості цілей та їх ідентифікація. Об’єкт, який потрібно відстежувати, виділяється шляхом малювання обмежувальної рамки навколо нього. У випадку послідовності зображень це зазвичай відбувається на першому зображенні, а в відео на першому кадрі. Згодом, алгоритм відстеження має передбачити положення об’єкта на наступних кадрах, одночасно ідентифікуючи його. Цей процес можна виконати вручну, або автоматично. Користувачі виконують ручну ініціалізацію, щоб позначити положення об’єкта за допомогою

обмежувальних рамок або еліпсів. З іншого боку, детектори об’єктів зазвичай використовуються для автоматичної ініціалізації;

Крок 2. Моделювання зовнішнього вигляду об’єкта важливе для врахування різноманітних умов, таких як освітлення, кути огляду та швидкість руху, які можуть впливати на вигляд об’єкта на зображеннях. Це дозволяє алгоритмам відстеження адаптуватися до змін та спотворень, які можуть виникати під час руху об’єкта, що допомагає уникнути дезінформації та втрати сліду об’єкта. Цей крок завжди складається із двох компонентів: візуального представлення – побудови надійних описів об’єктів, з використанням візуальних характеристик, та статистичного моделювання – побудови ефективних математичних моделей для ідентифікації об’єктів, з використанням методів статистичного навчання;

Крок 3. Прогнозування руху – це оцінка руху об’єкта, який ідентифікувався, з метою визначення точності положення об’єкта, яку може передбачити модель. Це задача оцінки динамічного стану, і вона зазвичай вирішується за допомогою предикторів, таких як: методи лінійної регресії, фільтри Калмана. На рисунку 2.4 представлено фільтр Калмана;

Рисунок 2.4 – Алгоритм фільтра Калмана [19]

Крок 4. Встановлення місцезнаходження цілі – надає приблизне місцезнаходження об’єкта, та ймовірності його розташування. Після цього

використовується візуальна модель для точного визначення місцезнаходження цілі. Це може бути здійснено через жадібний пошук або максимальну апостеріорну оцінку, які засновані на оцінці руху.

2.1.3 Проблеми під час відстеження об’єктів

Відстеження об’єктів під час простих умов є досить простою задачею для сучасних трекерів. Але нажаль, у реальності, відстеження об’єктів дуже часто відбувається за складних умов, наприклад: часта зміна яскравості зображення, перекриття об’єкта, або зміна фону зображення. Нижче наведено найпоширеніші проблеми, під час відстеження об’єктів [20]:

– перевантаження фону – коли фон густо заселений, виділення ознак, виявлення або навіть відстеження цільового об’єкта ускладняється, оскільки фон вносить більше інформації або шуму, роблячи мережу менш сприйнятливою до важливих ознак;

– зміна освітленості – це ситуація, коли освітлення об’єкта інтересу різко змінюється, коли він рухається, що робить його виявлення та оцінку складнішою;

– оклюзія – це ситуація, коли рамки цільового об’єкта часто закриваються, коли різні об’єкти та тіла входять або виходять з кадру. Це заважає алгоритму ідентифікувати та відстежувати об’єкт, оскільки це заважає фону або передньому плану. Це часто відбувається, коли контури кількох об’єктів надто близькі один до одного, що заважає алгоритмам ідентифікувати новий об’єкт, що відстежується;

– зміна форми об’єкта – об’єкт інтересу може обертатися, зменшуватися в розмірі, деформуватися в різних зображеннях і кадрах. Це може бути результатом кількох змін, таких як зміна точки зору. Крім того, ця

проблема часто викликає труднощі для алгоритму в інтуїтивному відстеженні об’єкта;

– швидкий рух – це вплив на здатність відстежувати об’єкт його швидкість руху;

– мала роздільна здатність – залежно від роздільної здатності, кількість пікселів у рамці навчального набору даних може бути замалою для послідовного відстеження об’єктів;

– зміна масштабу зображення – це ситуація, коли обмежувальні рамки першого кадру та поточного кадру виходять за межі заданого діапазону, та здатність алгоритму відстежувати цільовий об’єкт може бути порушена.

Отже, відстеження об’єктів має велику кількість труднощів, під час застосування алгоритмів в умовах реального світу. Саме через ці проблеми, було винайдено алгоритми глибокого навчання, які застосовують нейронні мережі, що значно покращують їх ефективність, роблячи алгоритми більш гнучкими, та надаючи можливість застосовувати дані алгоритми в умовах реального світу.

2.1.4 Алгоритми глибокого навчання для відстеження об’єктів

Традиційні або класичні алгоритми машинного навчання, такі як алгоритм k-найближчих сусідів, або машина опорних векторів, використовувалися в деяких методологіях. Ці підходи ефективні для прогнозування цільового об’єкта, але вони вимагають вилучення важливої та дискримінаційної інформації професіоналами. Натомість, алгоритми глибокого навчання, витягують ці важливі ознаки та уявлення самостійно. Нижче представлено найпопулярніші алгоритми глибокого навчання, які використовуються для відстеження об’єктів.

Алгоритм DeepSORT. Оригінальний алгоритм SORT надає дуже добрі результати з точки зору точності відстеження та акуратності. Але SORT повертає треки з великою кількістю перемикань ідентифікаторів і дає збій у випадку оклюзії. Це пов'язано із використанням матриці асоціацій. DeepSORT використовує кращу метрику асоціацій, яка поєднує дескриптори руху та зовнішнього вигляду. DeepSORT можна визначити як алгоритм відстеження, який відстежує об'єкти не тільки на основі швидкості та руху об'єкта, але й на основі зовнішнього вигляду об'єкта (рис. 2.5).

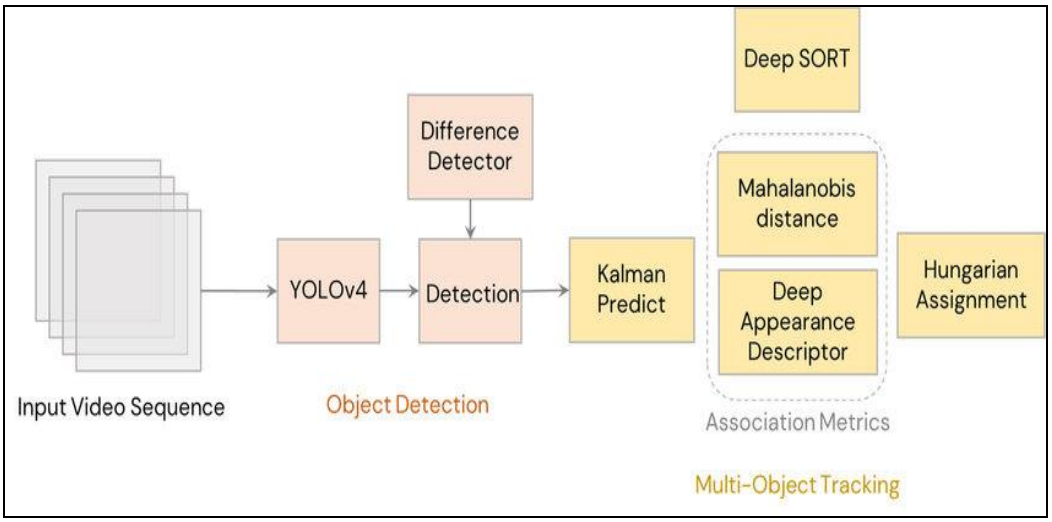


Рисунок 2.5 – Архітектура Deep Sort [21]

Для цієї мети, перед початком відстеження, в автономному режимі тренується добре дискримінуюча ознака, що вбудовується, безпосередньо перед початком відстеження. Мережа навчається на великому наборі даних повторної ідентифікації об'єктів, що робить її придатною для відстеження контексту. Для навчання метричної моделі глибокої асоціації в DeepSORT використовується підхід навчання на основі косинусної метрики. Косинусна відстань враховує інформацію про зовнішній вигляд, що особливо корисно для відновлення ідентичності після довготривалої оклюзії, коли рух є менш дискримінативним. Це означає, що косинусна відстань є метрикою, яка допомагає моделі відновлювати ідентичність у випадку довготривалої

оклюзії, коли оцінка руху також не спрацьовує. Використання цих простих речей може зробити трекер ще більш потужним і точним. На рисунку нижче представлено архітектуру алгоритму DeepSort.

MDNet (Multi-Domain Net) – це різновид алгоритму відстеження об’єктів на основі CNN (згорткова нейронна мережа), який використовує для навчання великі набори даних. Використовуючи анотовані відео, що належать до різних доменів, він вивчає спільні представлення цілей. Його метою є вивчення різноманітних варіацій і взаємозв’язків у просторі. MDNet складається з двох компонентів. Підготовка – це тренування алгоритму на кількох анотованих відео, щоб досліджувати репрезентацію та просторові характеристики. У цьому випадку для вивчення багатодоменного представлення потрібна мережа глибокого навчання. Візуальне відстеження в режимі онлайн – це процес, де шари, специфічні для домену, видаляються, створюючи спільні шари в мережі. Вони складаються з вивчених репрезентацій. Під час виведення, додається шар бінарної класифікації, який потім додатково навчається, або допрацьовується в режимі онлайн. На рисунку нижче представлена архітектура MDNet.

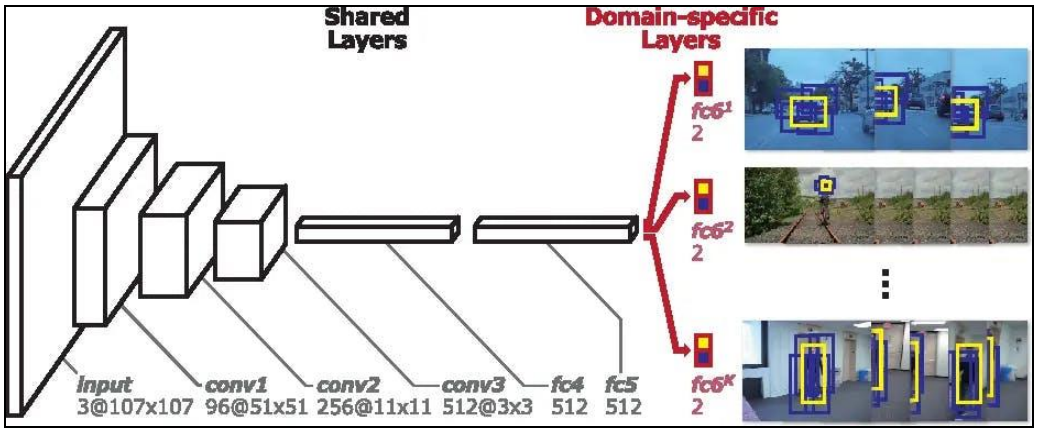


Рисунок 2.6 – Архітектура багатодоменної мережі, що показує спільні рівні та K відгалужень доменних рівнів [22]

– GOTURN – цей алгоритм відстеження використовує базову мережу прямого поширення, яка не потребує онлайн-навчання, що дозволяє йому

працювати зі швидкістю 100 кадрів на секунду під час тестування. Алгоритм навчається як на позначених відео, так і на великій колекції зображень, запобігаючи надмірному пристосуванню. Алгоритм відстеження навчається відстежувати звичайні об’єкти в реальному часі, коли вони рухаються по простору. Архітектура мережі GOTURN представлена на рисунку 2.7. На вхід мережі подається область пошуку та ціль поточного кадру. Навчаючись порівнювати ці області, мережа може визначити цільовий об’єкт на поточному зображенні.

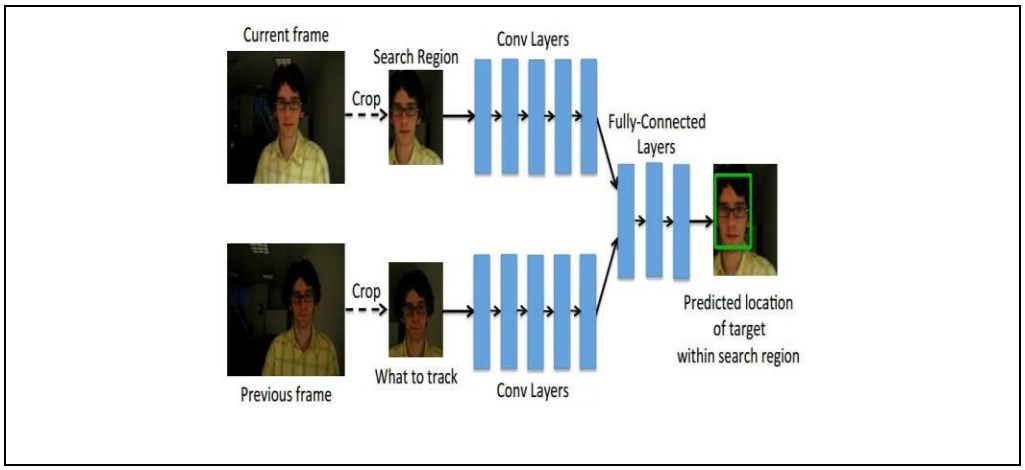


Рисунок 2.7 Архітектура мережі GOTURN [23]

SiamMask – використовує сіамські мережі, та метою даного алгоритму є покращення техніки офлайн-навчання повністю згорткової сіамської нейронної мережі. Сіамські мережі – це глибокі нейронні мережі згорткового типу, які використовують два вхідні шляхи для створення компактного просторового представлення ознак. Вони отримують на вхід обрізане зображення та велике зображення для подальшого пошуку. Сіамська мережа має один вихід, який порівнює подібність двох вхідних зображень і визначає, чи містять вони однакові об’єкти. Завдяки застосуванню бінарної сегментації для збільшення втрат, цей підхід стає дуже ефективним для алгоритмів відстеження об’єктів. Після навчання, SiamMask може працювати в режимі онлайн і потребує лише ініціалізації обмежувальних рамок. Він може

адаптуватися до різних масок сегментації об’єктів та швидко обробляти обмежувальні рамки зі швидкістю 35 кадрів на секунду.

2.2 Методи детекції об’єктів

Відстеження об’єктів на кожному кадрі зображення застосовує методи детекції об’єктів задля ідентифікації об’єктів. Нижче наведено основні методи детекції об’єктів:

– метод хмарних точок – основна ідея методу полягає в тому, що об’єкти на зображенні можна розглядати як групи точок, які знаходяться близько одна до одної і утворюють певні структури, або хмарні утворення. Потенційні об’єкти можна виділити та ідентифікувати шляхом аналізу геометричних характеристик цих хмарних точок, таких як: розмір, форма та положення на зображенні. Цей метод добре працює в ситуаціях, коли можна виділити об’єкти за їхніми візуальними характеристиками, не вимагаючи складних алгоритмів або навчання моделей;

– метод порогової обробки – основна ідея цього методу полягає в тому, що необхідно встановити поріг інтенсивності. За цього порогу всі пікселі вважаються об’єктами, а інші пікселі вважаються фоном;

– метод фільтрації ознак – основна ідея методу фільтрації ознак полягає в тому, що кожна особливість об’єкта можна розглядати як фільтр, який допомагає виділити об’єкт на зображенні. Наприклад, можна виділити об’єкти за їхніми візерунками та поверхневими деталями за допомогою аналізу текстури та використання колірних фільтрів. У поєднанні з кількома алгоритмами та фільтрами можна досягти точної та надійної детекції об’єктів;

– метод, який використовує глибоке навчання. Глибоке навчання дозволяє створювати моделі, здатні класифікувати об’єкти з великою

точністю, завдяки складним нейронним мережам, здатним навчатися на великих обсягах даних. Такий метод є одним із найбільш сучасних і ефективних у роботі з ідентифікацією об’єктів;

– метод каскадних ідентифікаторів – це унікальний метод пошуку об’єктів, заснований на послідовному використанні різних класифікаторів і фільтрів. Його основна перевага полягає в тому, що він може швидко й ефективно відкидати частини зображення, які не містять об’єктів, що нас цікавлять, зосереджуючись лише на ділянках, які можуть бути важливими.

2.3 Особливості реалізації рішень для системи відеоспостереження iSpy

Система відеоспостереження iSpy має відкритий вихідний код, який дозволяє розробникам легко інтегруватися у застосунок, змінюючи вихідний код для власних потреб, з метою покращення даного застосунку. Система відеоспостереження iSpy має ліцензію GPL (General Public License), завдяки якій користувачі мають право на завантаження, та зміну застосунку iSpy задля власних потреб [24].

Застосунок iSpy також має Agent DVR API, завдяки якому можливо переглянути усі можливі HTTP запити до даної системи. Інтерфейс програмування додатків, також відомий як API, є складним переліком інструкцій, протоколів і інструментів, призначених для взаємодії компонентів програми. Розуміння функцій і можливостей API дозволяє розробникам спілкуватися та обмінюватися даними. У процесі розробки програмного забезпечення API полегшують доступ до готових компонентів без необхідності повторної реалізації. Крім того, API полегшують обмін даними та функціями між різними програмними системами та платформами.

Застосунок iSpy має документацію власного API. На рисунку 2.7 представлено параметри методів API.

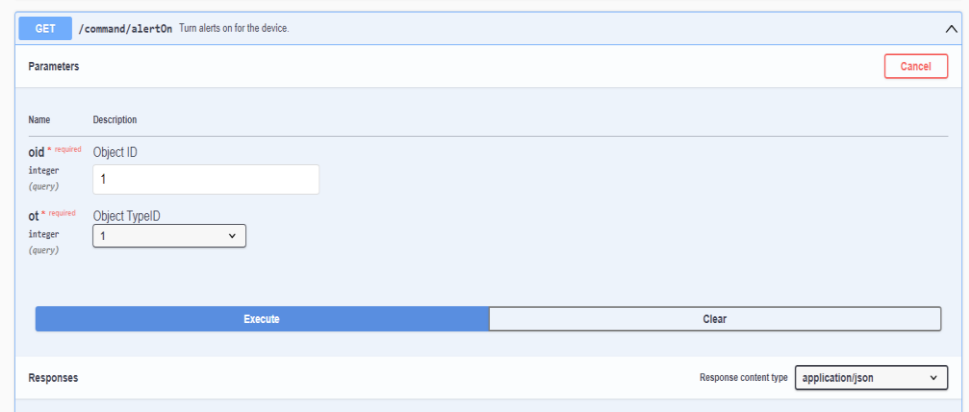


Рисунок 2.7 – Документація команди iSpy API

iSpy API було написано за допомогою застосунку Swagger, завдяки якому користувачі мають можливість зручно переглянути команди API, та навіть спробувати запустити будь яку команду, та отримати результат дії. На рисунку 2.8 представлено результат виконання команди.

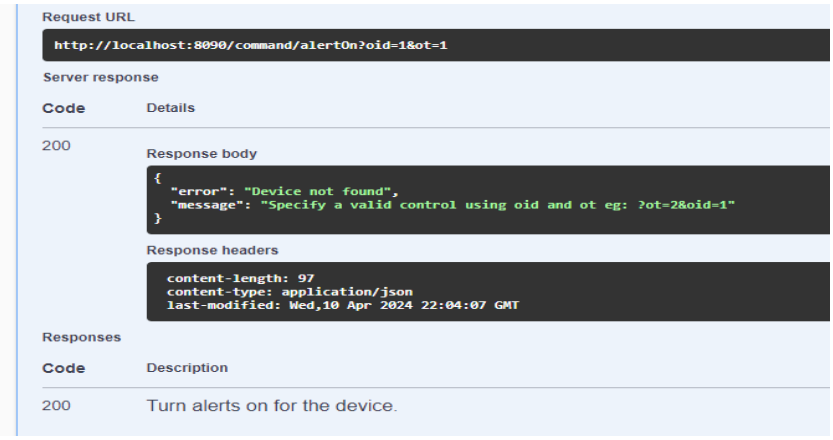


Рисунок 2.8 – Результат виконання команди iSpy API

2.4 Фреймворк Ultralytics

2.4.1 Загальна інформація про фреймворк

Ultralytics [25] – це фреймворк з відкритим вихідним кодом, написаний на Python. Його мета полягає в тому, щоб створювати, навчати та

застосовувати моделі глибокого навчання для завдань комп’ютерного зору та обробки зображень.

Функціональні можливості Ultralytics:

– детекція об’єктів – Ultralytics має змогу знаходити та класифікувати об’єкти на зображеннях, або відео;

– відслідковування об’єктів – трекінг дії об’єкта по кадрам відео;

– розпізнавання облич – фреймворк має змогу розпізнавати обличчя людей на відео та зображеннях;

– генерація зображень – фреймворк має змогу генерувати зображення на основі заданих зображень, чи параметрів;

– сегментація зображень – фреймворк може сегментувати зображення, відокремлюючи окремі сфери інтересів;

Отже, фреймворк Ultralytics є потужним фреймворком для роботи із комп’ютерним зором, який має добре навчені технології обробки зображень, котрі можуть застосовуватися у багатьох сферах життя.

2.4.2 Моделі YOLOv8

YOLOv8 пропонує широкий спектр моделей, кожна з яких спеціалізована для вирішення певних завдань комп’ютерного зору. Ці моделі призначені для виявлення об’єктів і виконання різноманітних завдань, включаючи сегментацію об’єктів, виявлення поз/ключових точок, виявлення орієнтованих об’єктів і класифікацію.

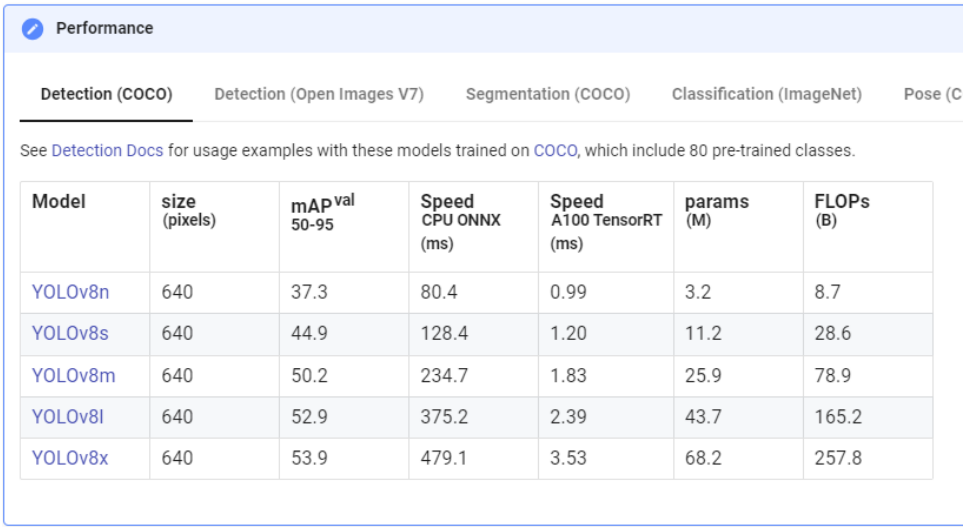
Кожен варіант серії YOLOv8 оптимізований для виконання відповідного завдання, що гарантує високу продуктивність і точність. Крім того, ці моделі сумісні з різними режимами роботи, такими як висновок, валідація, навчання та експорт, що полегшує їх використання на різних етапах розгортання та розробки.

Як було написано раніше, моделі YOLOv8 застосовуються для п’яти типів завдань: детекція, сегмантація, виявлення поз, детекція обернених об’єктів, та класифікація об’єктів на зображенні. Також, вони мають різні розміри: n (nano), s (small), m (medium), l (large) та x (extra). Дані розміри ідуть за порядком зростання, та потребують більше ресурсів від комп’ютеру, на якому запущено дану модель. Також, збільшення розміру моделі призводить до збільшення точності роботи детекції об’єкта на зображенні.

Для виконання мети даної роботи було обрано моделі для виконання задачі детекції об’єктів. Даний тип моделей ідеально підходить для виконання задачі даної роботи, вони мають більше сто класів, за якими можуть класифікувати об’єкти. Необхідно було обмежити моделі YOLOv8, аби вони мали змогу класифікувати тільки об’єкти класу «людина» на зображенні.

Кожна модель має певні метрики, за якими можливо було визначити якість моделі. На сьогоднішній день, широко використовується метрика mAP, або mean Average Precision – це метрика, яка розраховує середню точність для всіх класів. Також, важливою метрикою є швидкість виконання операції, що означає, як швидко модель може обробити зображення. FLOPs – це метрика, яка свідчить про кількість операцій з плаваючою точкою, необхідній моделі. Чим нижче значення FLOPs, тим ефективніша модель детекції.

На рисунку 2.9 представлено метрики для моделей різного розміру YOLOv8, які виконують задачі детекції. Дані моделі зайняли перше місце серед аналогів моделей інших фреймворків, та вважаються одними з найкращих у своєму класі.



Performance

Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) | Pose (C

See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Рисунок 2.9 – Метрики моделі детекції YOLOv8

2.4.3 Трекери на базі YOLOv8

YOLOv8 від Ultralytics представляє два типа трекерів, які можуть бути використані під час трекінгу об’єктів. Нижче, наведено детальний опис кожного трекера:

– ByteTrack – швидкий, та ефективний трекер від Ultralytics, який є простим у використанні та налаштуванні, та дуже точний для короткострокових траєкторій. Але натомість, він є менш точним для довгострокових траєкторій, чутливий до шуму та оклюзій.

– BoT-SORT – дуже точний, особливо для довгострокових траєкторій трекер, стійкий до шуму та оклюзій, але він є повільнішим за ByteTrack, та більш складним у використанні та налаштуванні.

Нижче наведено порівняльна таблиця результатів тестування трекерів ByteTrack [26] та BoT-SORT [27], за допомогою набору даних MOT20 (табл. 2.1). MOT 20 – це набір даних для розв’язання завдань відстеження об’єктів. Він містить відео-записи з реального життя, отримані з різних джерел. Набір даних MOT20 використовується для оцінки та порівняння

алгоритмів відстеження об’єктів, щоб визначити їх ефективність у реальних умовах.

Таблиця 2.1 – Результати тестування трекерів ByteTrack та BoT-SORT [26-27]

Назва трекеру	**MOTA**	**IDF1**	**HOTA**
ByteTrack	77.8	75.2	61.3
BoT-SORT	77.7	76.3	62.6

Результати тестування визначають метрики MOTA, IDF1 та HOTA, за якими можливо оцінити якість трекерів. Метрика MOTA (Multiple Object Tracking Accuracy) – це метрика, яка обчислюється як відношення загальної кількості помилок до загальної кількості реальних об’єктів, що були відстежені.

Метрика IDF1 – це метрика, яка обчислюється як середнє значення між точністю і чутливістю. Вище значення IDF1 вказує на кращу якість відстеження об’єктів.

Метрика HOTA (Higher Order Tracking Accuracy) – це метрика оцінки точності відстеження об’єктів, яка враховує не лише співпадіння та неточності відстеження на рівні об’єктів, а й враховує просторові та часові зв’язки між об’єктами. HOTA надає більш докладну та повну оцінку точності відстеження об’єктів, охоплюючи такі аспекти, як точність, повноту та зв’язковість відстеження.

Результати тестування показали, що обидва трекери YOLOv8 від Ultralytics займають високі місця у даних тестуваннях, визнаючись одними з найкращих трекерів свого типу серед конкурентів.

2.5 Проектування архітектури плагіну

Плагін буде складатися із Front End та Back End частин застосунку. На рисунку 2.10 зображено архітектуру Front End частини застосунку, яка забезпечує інтерактивну комунікацію користувача із системою. В даній частині, користувач буде мати можливість задати необхідну інформацію, для отримання статистики на вебсторінці застосунку, а також, генерації окремого документу із загальною статистикою, яку буде сформовано на основі даних із бази даних. На рисунку 2.11 зображено архітектуру Back End частини застосунку, яка буде написана, як окремий застосунок. Даний застосунок буде виконувати приховану логіку від користувача з трекінгу та підтрахунку кількості людей у приміщенні. А також, взаємодіяти із застосунком iSpy та базою даних, для збереження інформації про вхід, або вихід людини у приміщення, яку в подальшому буде використовувати Front End частина. Також, слід зауважити, що застосунок iSpy має власний інтерфейс, за яким користувач буде мати можливість налаштувати відео потік та переглянути збереженні відеозаписи, викликані із Back End частини.

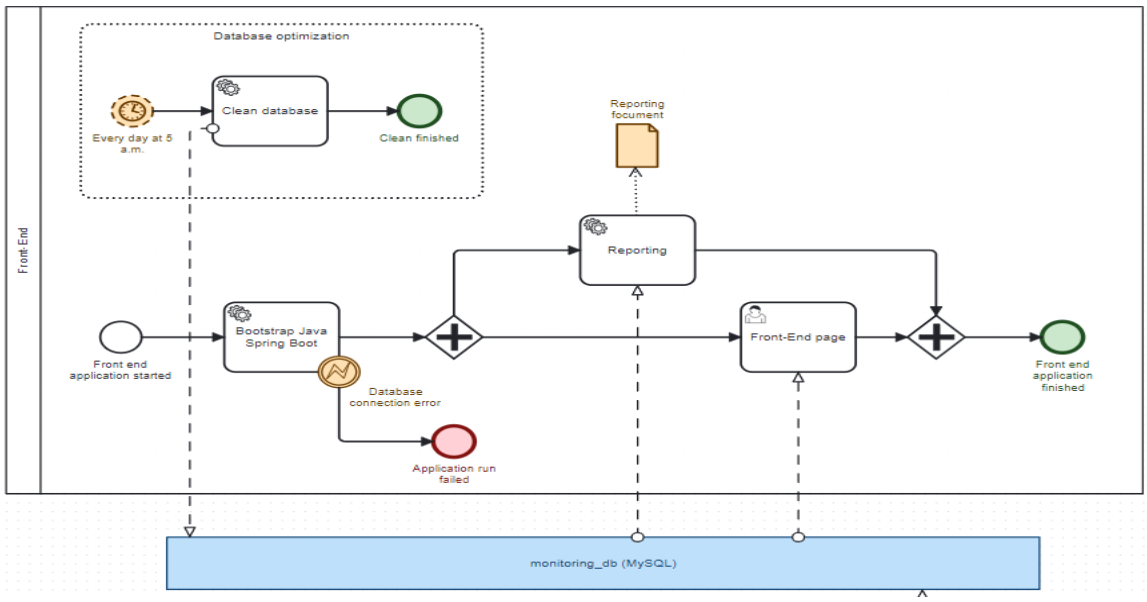


Рисунок 2.10 – Архітектура Front End частини плагіна

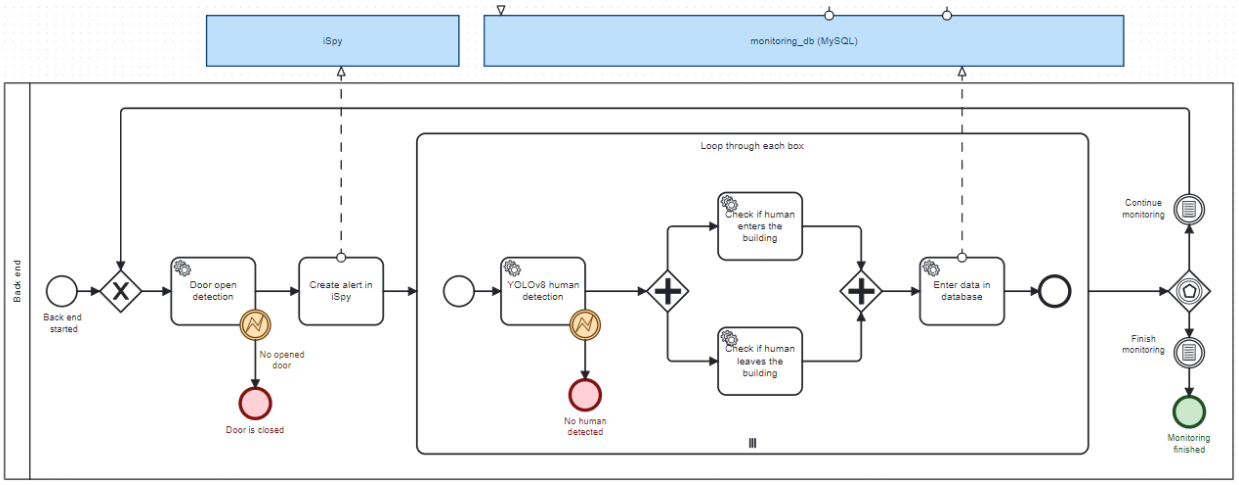


Рисунок 2.11 – Архітектура Back End частини плагіна

Після запуску Back End частини застосунку, всередині програмного коду визначені параметри, які потрібно визначати користувачу самостійно. Параметри MAX_X, MAX_Y, MIN_X, MIN_Y є точками, за якими задається «площина перетину», перетнувши її, об’єкт буде вважатися зайшовшим, або покинувшим приміщення. Також, користувачу потрібно буде змінити параметр порогу яскравості зображення, який буде використовуватися для детекції відкриття двері у приміщенні. Необхідно буде задати чотири координати *x1, x2, y1, y2* будь-якого краю двері, які будуть застосовуватися для визначення площини, яка буде необхідна для детекції відкриття двері. Даний застосунок буде оброблювати відео потік, який необхідно буде задати користувачу власноруч. Back End частина буде оброблювати кожен другий кадр даного відео потоку, також, кожен кадр буде змінювати власний розмір до 1600 пікселів у висоту, та 900 пікселів у ширину. Зміна розміру кадру необхідна для забезпечення оптимізації роботи трекеру, який буде повільніше працювати на зображеннях більшого розміру. Також, трекер від Ultralytics має декілька параметрів:

– persist = True, який вказує, що поточне зображення є наступною у послідовності, та необхідно очікувати треки із попереднього зображення у поточному зображенні;

– stream_buffer = True, який застосовується для оптимізації роботи під час відео потоку

– tracker = "bytetrack.yaml" – це визначення трекеру, який буде застосовуватися у застосунку. За замовчуванням, трекер буде використовувати BoT-SORT;

– classes [0] – це визначення класів, які може визначати детектор, в даному випадку необхідно визначити лише клас людей, який знаходиться на першому місці у списку;

– conf = 50 – це параметр впевненості детектора у трекері, визначає з якого значення впевненості, трекер може вважати об’єкт приналежним до визначеного класу.

2.6 Алгоритми для рішень практичних задач

Плагін iSpy має основну логіку у Back End частині застосунку. Після запуску програми, вона автоматично завантажує модель YOLOv8, та підключається до бази даних. Згодом, плагін підключається до відео потоку, через методи бібліотеки open-cv, якщо виникла помилка під час підключення, то програма завершується, а інакше – застосунок продовжує свою роботу у нескінченному циклі, який має наступні кроки:

Крок 1. Зміна розміру кадрів. Плагін змінює розмір отриманого кадру із відео потоку до 1600 пікселів у ширину, та 900 пікселів у довжину. Також, створює окрему копію поточного кадру, який було вирізано за попередньо заданими координатами верхнього кута двері. Демонстрація відео потоку у окремому вікні.

Крок 2. На перших десяти кадрах, плагін записує вирізані кадри двері у окремий список, попередньо переробивши його у сірий кадр.

Крок 3. Визначення середньої яскравості та порівняння із поточною яскравістю. На одинадцятому кадрі, плагін визначає середнє арифметичне градації сірого у перших десяти кадрах, які були визначені попередньо. Після визначення середнього арифметичного, застосунок отримує градацію сірого поточного кадру, та порівнює його із різницею середнього арифметичного та порогу яскравості. Якщо значення яскравості поточного кадру верхнього кута двері є більшим, або меншим порівнювального значення, то двері вважаються відкритими. Таким чином, при відкритті двері, буде відбуватися різка зміна яскравості у області верхнього кута двері, яка буде помічатися плагіном, та робити наступні кроки.

Крок 4. Після детекції відкриття двері, плагін зробить запит на увімкнення запису до сервісу iSpy.

Крок 5. Запуск трекера. Після детекції відкриття двері, плагін вважає, що на наступних 30 двері будуть вважатися відкритими, це достатньо того, аби застосунок зміг запустити трекер Byte Track від Ultalytics. Він одразу буде перевіряти на наявність знайдених людей, якщо на 31 кадрі, після фіксації відкриття дверей, не було виявлено жодної людини, то двері будуть вважатися закритими. Якщо трекер знайшов людей на кадрі, то він буде продовжувати відслідковувати їх, фіксуючи їх координати та задаючи їм унікальний ідентифікатор. Застосунок запустить відображення знайдених результатів у окремому вікні. Після того, як трекер загубить людину, яку він відслідковував, то двері будуть вважатися закритими, та підрахунок кадрів знов буде починатися з 0 і алгоритм повернеться на крок 2.

Крок 6. Збереження поточних координат у список попередніх кадрів. Після детекції людини на кадрі, трекер визначить поточні координати x_1, x_2, y_1, y_2 знайденого об’єкту. Трекер визначить поточні точки перетину для осі *X*, та *Y*, застосувавши дану формулу:

$$x_n = (x_1 + x_2) / 2,$$

$$y_n = y_2 - ((y_2 - y_1) * 0{,}1).$$

Якщо об’єкт було тільки знайдено, та список порожній, то алгоритм збереже знайдені точки перетину у список попередніх точок перетину та виконає даний крок ще один раз, інакше він виконає наступний крок.

Крок 7. Визначення перетину. Застосунок має область перетину, яку було задано користувачем попередньо. Після підрахунку точок перетину, плагін буде перевіряти, аби точки перетину знаходились у даній області. Після того, як об’єкт потрапить у дану зону, застосунок отримає попередні точки перетину, та виконає наступний крок.

Крок 8. Перевірка входження людини. Після того, як застосунок визначить перетин, він порівняє збільшення значення по осі Y для поточної точки перетину із попередньою точкою перетину. Якщо перевірка пройшла успішно, то алгоритм перейде до Кроку 10.

Крок 9. Перевірка покидання людиною офісу. Після того, як застосунок визначить перетин, він порівняє зменшення значення по осі Y для поточної точки перетину із попередньою точкою перетину. Якщо перевірка пройшла успішно, то алгоритм перейде до Кроку 10.

Крок 10. Запис у базу даних. Алгоритм виконає запит у базу даних, та визначить поточну кількість людей у приміщені. Після цього, алгоритм виконає відповідну запис у базу даних, з інформацією про поточний час, збільшеною, або зменшеною кількістю людей у приміщенні та напрямком руху людини.

Слід зазначити, що застосунок має невелику кількість логіки у Front End частині, яка має три функціонали:

– чистка бази даних. Кожен день, о п’ятій ранку, база даних буде змінювати значення кількості людей на 0, аби уникнути можливих помилок у підрахунку на наступний день;

– визначення поточної кількості людей, та відображення її на сайті застосунку;

– визначення кількості людей, які покинули, увійшли у будівлю за певний проміжок часу, та кількості людей, яка лишилась у будівлі.

2.7 Структура бази даних для збереження знайдених подій

Застосунок має базу даних MySQL, яка має одну базу даних monitoring_db, яка в свою чергу, має одну таблицю office_entries.

Дана таблиця має ID (унікальний ідентифікатор), який є цілим числом, яке автоматичне збільшується. Також, дана таблиця має, поле isEntered, яке представлено булевим типом, яке має True і False, якщо людина прийшла, або вийшла із приміщення відповідно. Дана таблиця має поле peopleCounter, яке визначає кількість людей у поточний момент часу, та поточний момент часу, зафіксований у полі DateTime.

3 РЕАЛІЗАЦІЯ ПЛАГІНУ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи було прийнято рішення розбити основну та другорядну логіку плагіну на дві окремі частини. Основну логіку виконує Back End частина застосунку, та другорядну логіку, яка є менш важливою, виконує Front End частина застосунку.

У ході проектування плагіну, потрібно було сформувати рішення, щодо мови програмування Back End частини застосунку, яка буде оброблювати зображення, та відслідковувати об’єкт. Дане рішення спиралося наявності корисних для даної задачі фреймворків, які підтримує мова програмування. Python – найпопулярніша мова програмування у сфері штучного інтелекту через простий і зрозумілий синтаксис мови, зручності тестування застосунків, великої кількості бібліотек і фреймворків для штучного інтелекту. Саме через наведені вище причини, було визначено застосовувати мову програмування Python [28] для Back End частини застосунку.

Back End частина застосунку має також багато бібліотек та фреймворків, які було встановлено через інтегровану середу розробки (IDE) PyCharm, яка розроблена компанією JetBrains та надає широкі можливості із написання та редагування коду, підтримує інструменти контролю версій, та надає зручний сервіс Python Packages, який встановлює необхідні бібліотеки та фреймворки у проєкт.

Через сервіс Python Packages було встановлено такі бібліотеки, та фреймворки:

NumPy – це бібліотека [29], яка надає підтримку для великих масивів та математичних функцій, дозволяючи зручно та ефективно виконувати обчислення;

OpenCV – це бібліотека [**Error! Reference source not found.**] із комп'ютерного зору, яка надає широкі можливості зручно, та ефективно працювати із комп'ютерним зором, через мову програмування Python;

Mysql – це бібліотека, яка надає можливість підключитися до бази даних MySQL та виконувати базові функції, для роботи з нею;

Ultralytics – це фреймворк, який надає широкий спектр готових рішень для комп'ютерного зору, зокрема він надає можливості просто та зручно використовувати написаний трекінг об'єктів.

Requests – це бібліотека, для виконання HTTP запитів напряму із Python.

Front End частина застосунку, яка містить другорядну логіку, яка не взаємодіє із виконанням логіки з трекінгу об'єктів, містить функціонал, який використовує надану інформацію із бази даних. Дана база даних пов'язує Back End та Front End частини, та являє собою реляційну базу даних MySQL.

Front End частину було вирішено написати на мові програмування Java [31], яка відома своєю надійністю та високою продуктивністю, що робить її популярним вибором для великих підприємств та розробників в усьому світі. Дана мова застосовується в основному в трьох сферах: веброзробка, android-розробка, та автоматизація бізнес процесів. Вона є класичною мовою, яка заснувала великий тренд у ООП (об'єктно орієнтовне програмування), та сьогодні досі залишається одним із найпопулярніших виборів розробників.

Дана мова містить зручний інструмент для управління залежностями Maven, він дозволяє автоматизувати процес збірки та керування залежностями, спрощуючи розробку Java-проєктів. На рисунку 3.1 представлено приклад додавання бібліотеки, застосовуючи Maven.

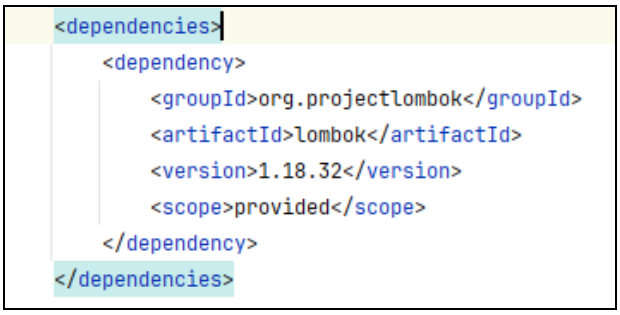


Рисунок 3.1 – Додавання бібліотеки, застосовуючи Maven

Front End частина застосунку використовує фреймворк Spring Boot – це розширення фреймворку Spring, яке надає швидкий спосіб створення вебзастосунків на Java. Він дозволяє розробникам швидко створювати потужні, гнучкі і масштабовані застосунки, зменшуючи необхідність вручну налаштовувати багато складних компонентів. Spring Boot забезпечує автоматичне конфігурування та вбудовані інструменти, такі як вбудований сервер застосунку, що спрощує розробку, тестування та розгортання. Він також постачається з багатьма корисними модулями, які допомагають розробникам зосередитися на бізнес-логіці своїх застосунків, а не на інфраструктурі.

Front End частина плагіну використовує бібліотеку JDBC [32], яка з’єднання коду з реляційними базами даних, бібліотеку lombok для спрощення створення класів, бібліотеку thymeleaf для динамічного виведення даних на HTML сторінці застосунку.

3.2 Підключення до відео потоку

Для використання даного плагіну, користувач має завантажити та встановити сервіс iSpy. Після установки даного сервісу, необхідно перейти за посиланням localhost:8090 у веббраузері користувача та увійти у обліковий

запис даного сервісу. Після авторизації користувача у сервіс iSpy, користувач має додати камеру у даний сервіс.

Для додавання камери у застосунку, користувач має натиснути на кнопку «New Device», показану на рисунку 3.2.

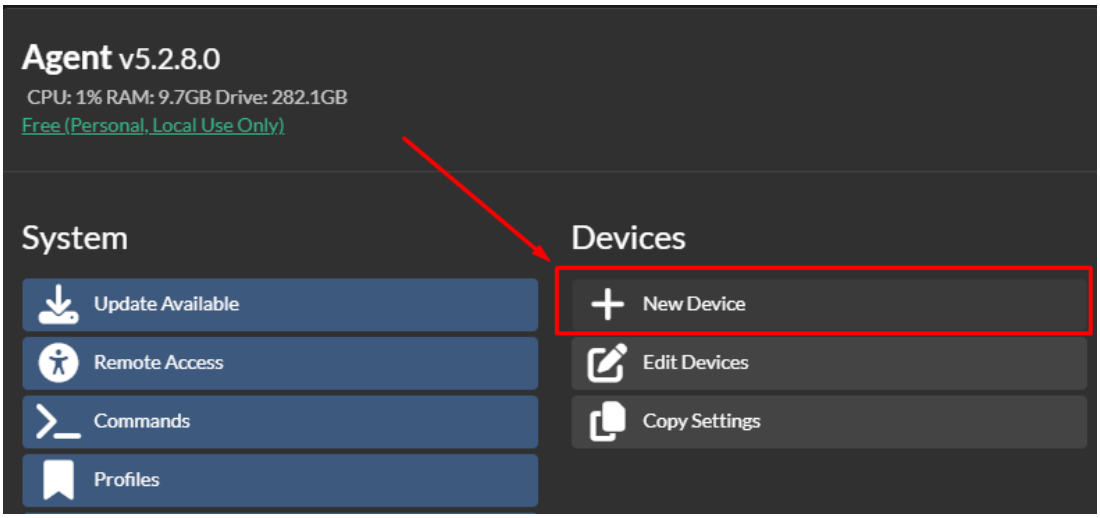


Рисунок 3.2 – Кнопка «New Device» у iSpy

Після цього, користувачу необхідно обрати джерело відео, натиснувши кнопку «Video Source», зображену на рисунку 3.3.

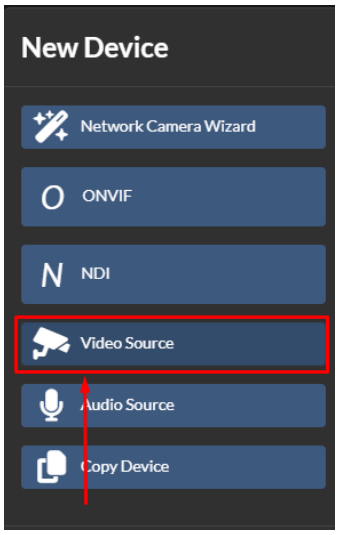


Рисунок 3.3 – Кнопка «Video Source» у iSpy

Після виконаних кроків з додавання камери, зазначених вище, користувачу потрібно буде заповнити поля для нової камери, наприклад, потрібно обрати «Network Camera» у полі «Source Type», після чого, буде надано можливість увести дані для авторизації для цього типу камери (рис. 3.4).

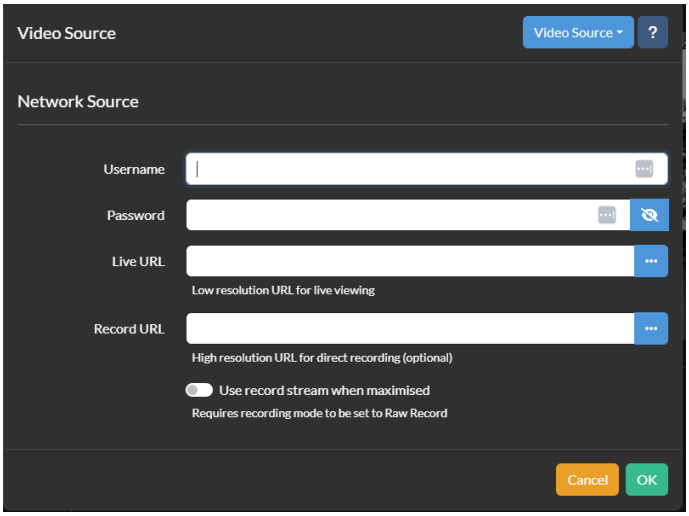


Рисунок 3.4 – Дані для авторизації для мережевої камери у iSpy

Також, для підключення до відео потоку, користувачу потрібно буде надати посилання на мережеву камеру у Back End застосунку. Нижче (рис. 3.5), наведено приклад надання посилання у коді.

```
capture = cv2.VideoCapture('rtsp://LOGIN:PASSWORD@192.168.161.31:111/Streaming/channels/302')
```

Рисунок 3.5 – Посилання на відео поток мережевої камери у Back End частині застосунку

3.3 Підготовка до використання плагіну

Після підключення до відео потоку, користувачу необхідно задати свої параметри всередині програмного коду Back End частини застосунку. Дані

параметри завжди написані великими літерами, які легко буде знайти на початку програмного коду Python. Також, вони виділені окремим коментарем, що полегшить їх пошук користувачам. На рисунку 3.6 представлені дані параметри, які необхідно буде користувачеві замінити.

```
# parameters have UPPER_NAMING
# define parameters on your own
DB_CONNECTION = mysql.connector.connect(
    host="localhost",
    user="root",
    password="qwerty",
    database="monitoring_db"
)
CAPTURE = cv2.VideoCapture('rtsp://USER_LOGIN:USER_PASSWORD@192.168.111.11:555/Streaming/channels/302')
RECORD_START_URL = "http://localhost:8090/command/triggerRecord?oid=2&ot=2"
MAX_X = 845
MIN_X = 598
MIN_Y = 700
MAX_Y = 750
BRIGHTNESS_THRESHOLD = 10
DOOR_REGION = (785, 241, 838, 280)  # (x1, y1, x2, y2)
```

Рисунок 3.6 – Параметри у Back End частині застосунку

Кожен параметр має просте та логічне ім’я, за яким користувачеві буде легко дізнатися, за що відповідає кожен з них.

Також, користувачеві необхідно задати параметри на Front End частині застосунку, замінивши дані у файлі application.yaml на власні, з метою підключення до бази даних. Нижче (рис. 3.7) представлено дані параметри.

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/monitoring_db
    username: root
    password: qwerty
    driver-class-name: com.mysql.cj.jdbc.Driver
```

Рисунок 3.7 – Параметри для Front End частини застосунку

3.4 Демонстрація роботи плагіна

3.4.1 Демонстрація роботи Front End частини плагін

Для запуска Front End частини плагіна, потрібно запустити pluginFront.jar файл, який буде запущено, як Java застосунок на локальному комп'ютері користувача. Для доступу до вебсайту застосунку, необхідно звернутись за посиланням localhost:8080 через веббраузер користувача, нижче (рис. 3.8) представлено головну сторінку вебсайту застосунку.

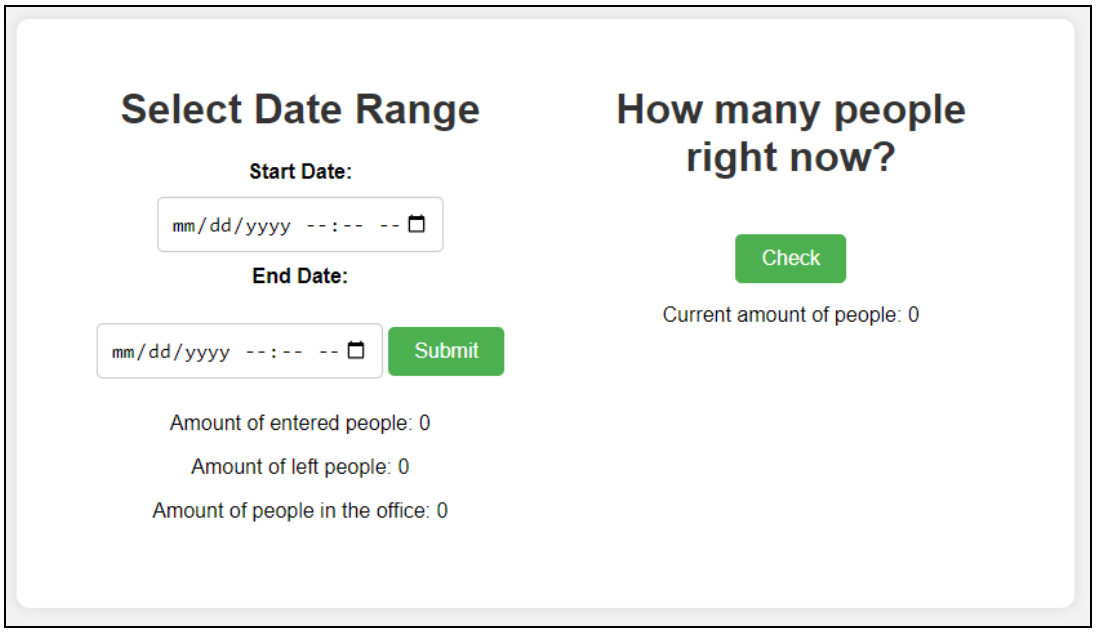


Рисунок 3.8 – Головна сторінка Front End частини застосунку

Дана вебсторінка має два функціонали:

– підрахунок поточної кількості людей у приміщені. Вебзастосунок звертається до бази даних MySQL, та отримує останній запис у базу даних, з якого отримує поточну кількість людей у приміщені. Після отримання поточного числа із бази даних, вебзастосунок відображає на головному сайті кількість людей у приміщені. На рисунку 3.9 представлено даний функціонал, після натискання на кнопку «Check»;

Рисунок 3.9 – Демонстрація функціоналу підрахунку поточної кількості людей у приміщенні

– підрахунок кількості людей, які увійшли та вийшли із приміщення, та демонстрація кількості людей, що залишилися у приміщенні за певний проміжок часу. Користувачу необхідно початковий та кінцевий проміжок часу, та натиснути кнопку «Submit». Після того, вебзастосунок звертається до бази даних MySQL та отримує останній запис у базі даних, який менший, або дорівнює кінцевому проміжку часу. Також, вебзастосунок двічі звертається до бази даних та підраховує кількість людей, що увійшли та вийшли із приміщення. Після виконання попередніх кроків, вебзастосунок демонструє отриману кількість людей на головному сайті. Нижче (рис. 3.10) наведено демонстрацію введення початкового та кінцевого проміжку часу, та на рисунку 3.11 зображено підраховані значення.

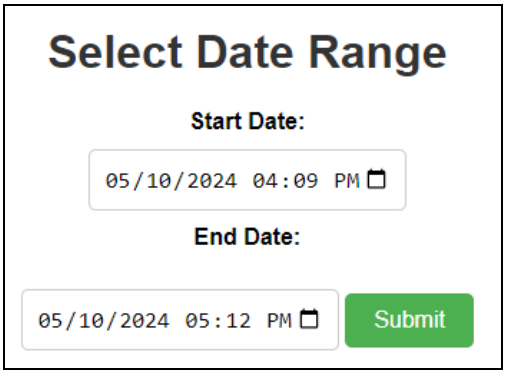


Рисунок 3.10 – Демонстрація введення даних у вебзастосунку

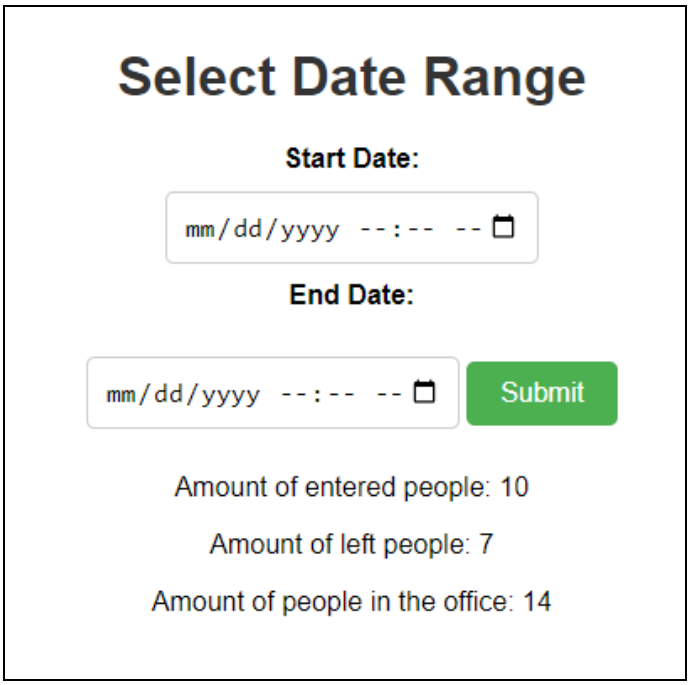


Рисунок 3.11 – Демонстрація підрахованих значень у вебзастосунку

3.4.2 Демонстрація роботи Back End частини плагіна

Для того, аби користувач запустив Back End частину застосунку, потрібно запустити Python файл tracker.py. Застосунок буде запущено та він автоматично підключиться до бази даних, завантажить модель YOLOv8, та підключиться до відео потоку. Якщо застосунок буде мати проблеми під час його запуску, у логах даного застосунку буде написано причину помилки, найчастіше може бути – проблема із підключенням до відеокамери. Користувач має приділити ретельну увагу до даного параметру, бо він є критичним для всієї роботи застосунку. Після успішного запуску застосунку, на головному екрані користувача буде видно пряму трансляцію із заданої камери. На рисунку 3.12 зображено показ відео потоку із камери, запущеного із Back End частини плагіну.

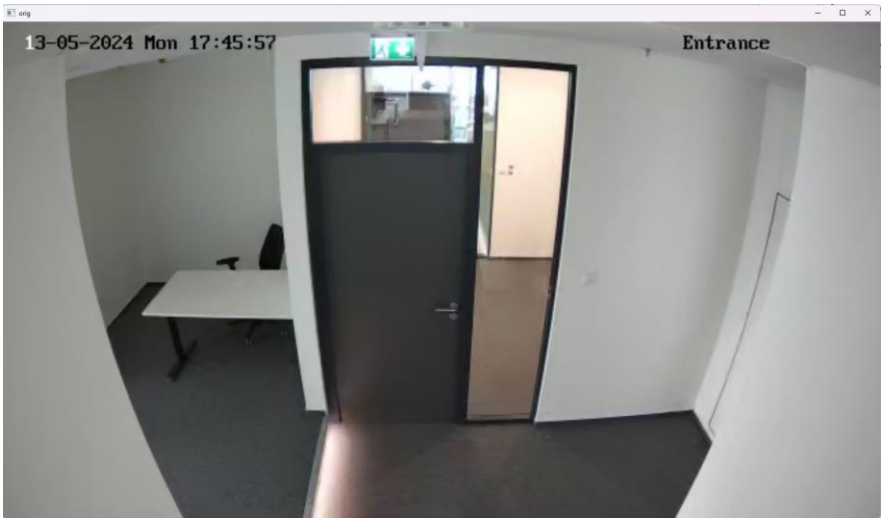


Рисунок 3.12 – Пряма трансляція із відеокамери, показана із Back End частини плагіну

Після успішного запуску, та підключення Back End частини застосунку до відео потоку, застосунок підрахує середню яскравість для окремої області двері, та буде чекати, коли двері відкриються. Після відкриття дверей, застосунок автоматично запустить трекер, та на екрані користувач з’явиться нове вікно із відео трансляцією, яку обробляє трекер від Ultralytics. На рисунку 3.13, людина буде знаходитись у кадрі, але не буде відкривати двері, трекер не буде запущено, та не буде відображатися вікно із знайденою областю людини.

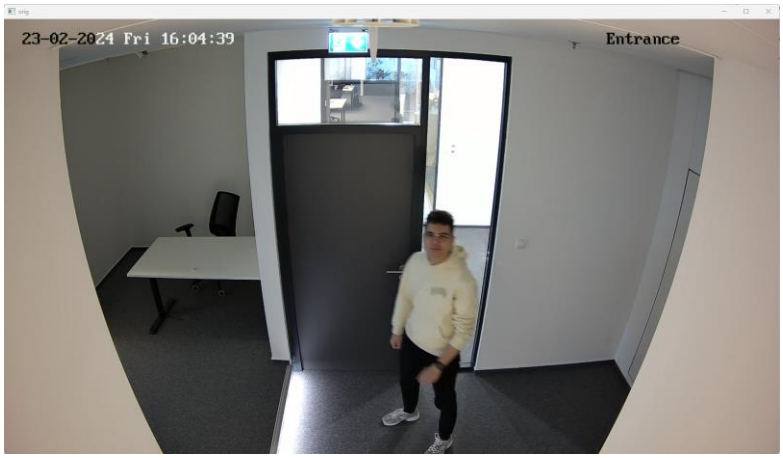


Рисунок 3.13 – Плагін не запустив трекер, тому що не двері не були відкриті

На рисунку 3.14, людина відкрила двері, та почала виходити із будівлі. На екрані користувача почалася відображатися трансляція оброблених кадрів трекером. Також, на екрані користувача відображається «зона перетину», в яку знайдена людина має зайти. Для того, аби трекер зміг визначити, чи зайшла людина до цієї зони, трекер визначає спеціальні «точки перетину», які також відображає на головному екрані користувача.

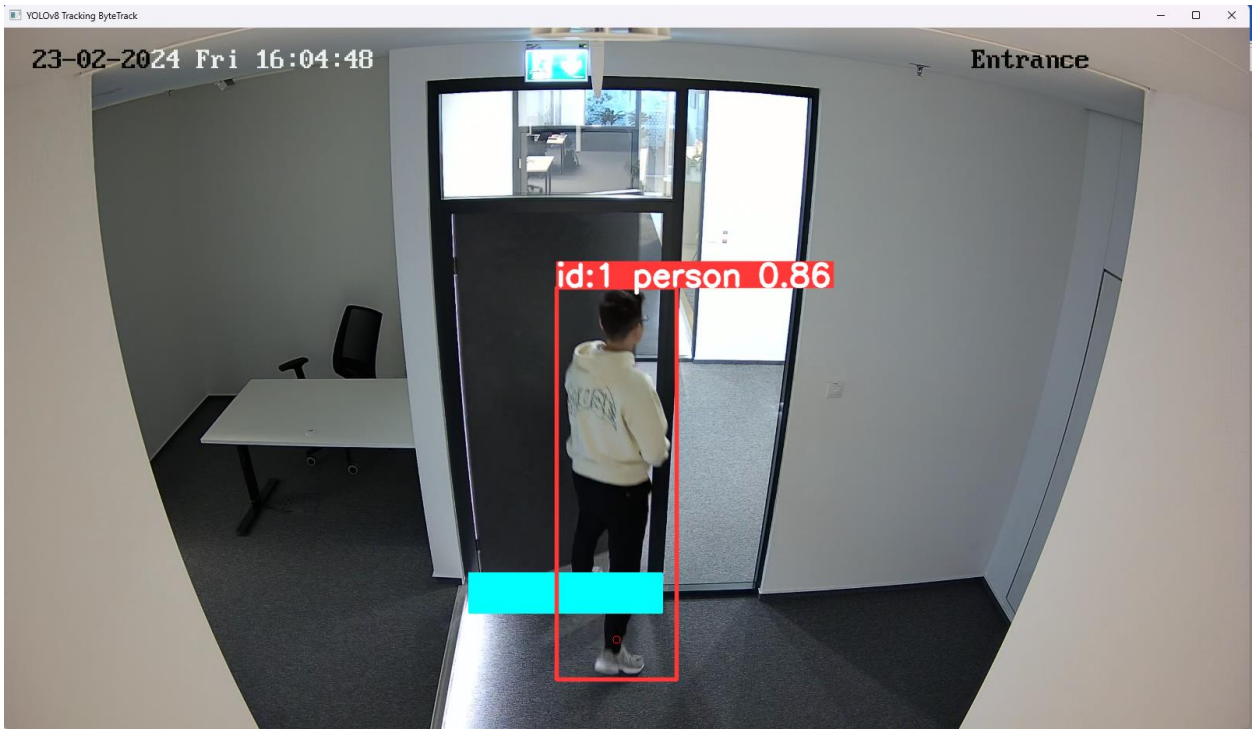


Рисунок 3.14 – Демонстрація обробленого кадру плагіном

Після того, як людина вийшла із будівлі, у базі даних з’являється нова запис із відповідними значеннями (рис 3.15).

	id	isEntered	DateTime	peopleCounter
▶	359	0	2024-05-13 18:12:01	4
*	NULL	NULL	NULL	NULL

Рисунок 3.15 – Запис у базі даних під час виходу людини із будівлі

Також, під час відкриття дверей, плагін запускає запис у застосунку для камер відеоспостережень iSpy, який робить запис відео на 15 секунд. Якщо

двері було відкрито декілька разів одночасно, то запис буде продовжуватися на 15 секунд довше. На рисунку 3.16, відображаються збережені відео із iSpy, які знаходяться на локальному диску користувача, даний шлях збереження відео є встановленим за замовченням у застосунку iSpy.

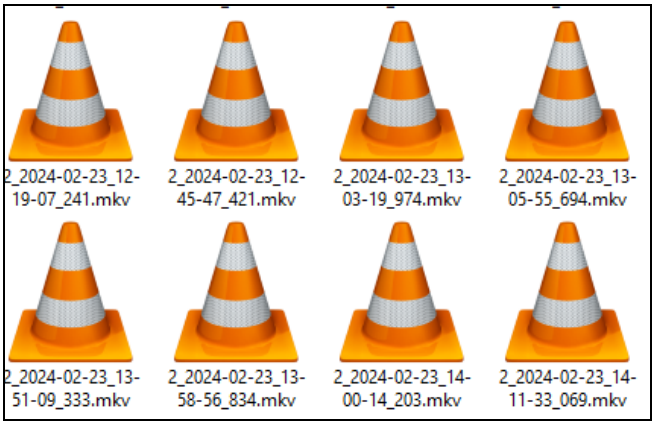


Рисунок 3.16 – Збереженні відео на локальному комп’ютері користувача із iSpy

Після того, як людина відкрила двері, та її почав відслідковувати трекер, двері будуть вважатися відкритими до того моменту, поки людина не вийде із зони видимості камери. Після цього, трекер знову буде чекати на відкриття дверей. На рисунку 3.17, плагін почав відслідковувати людину, яка входить у приміщення:

Рисунок 3.17 – Детекція входу людини у приміщення

Після детекції входу людини у приміщення, плагін зробить новий запис у базу даних із відповідними даними. На рисунку 3.18 представлено даний запис:

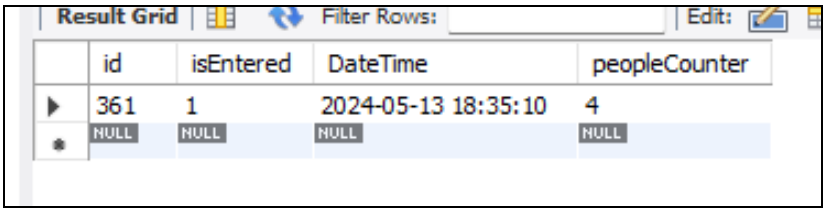


Рисунок 3.18 – Запис у базу даних, при детекції входу людини у приміщення

3.5 Експерименти з гіперпараметрами плагіну

3.5.1 Експерименти із вибору трекера від Ultralytics

Ultralytics надає можливість використовувати два трекера: BoT-SORT, який встановлений за замовченням, та ByteTrack. У ході написання даного плагіну, потрібно було визначити, який із трекерів краще підходить для виконання задач даної роботи. Спочатку, було проаналізовано кожен трекер, та визначено, що згідно документації, трекер BoT-SORT є більш точний та стабільний, але повільніший за ByteTrack, коли ByteTrack є більш швидким та ефективним, але менш стабільним за BoT-SORT.

Було прийнято рішення наочно перевірити обидва трекери. На рисунку 3.19 зображено демонстрація роботи двох трекерів від Ultralytics: BoT-SORT, та ByteTrack. За результатами перевірки, було видно, що трекер BoT-SORT, працював повільніше, та мав невелику перевагу у точності визначення людини у рамці. Але дана похибка не варта того, аби робити даний плагін повільнішим. Трекер ByteTrack достатньо добре справлявся із поставленою задачею, та повністю підходив під умови даного плагіну. Через те, що даний плагін буде запускати трекери на короткострокові терміни, лише у той час, коли двері у приміщенні будуть відкритими, використання трекеру

ByteTrack, який створений для обробки короткострокових відео, ідеально підходить під умови даного проєкту.

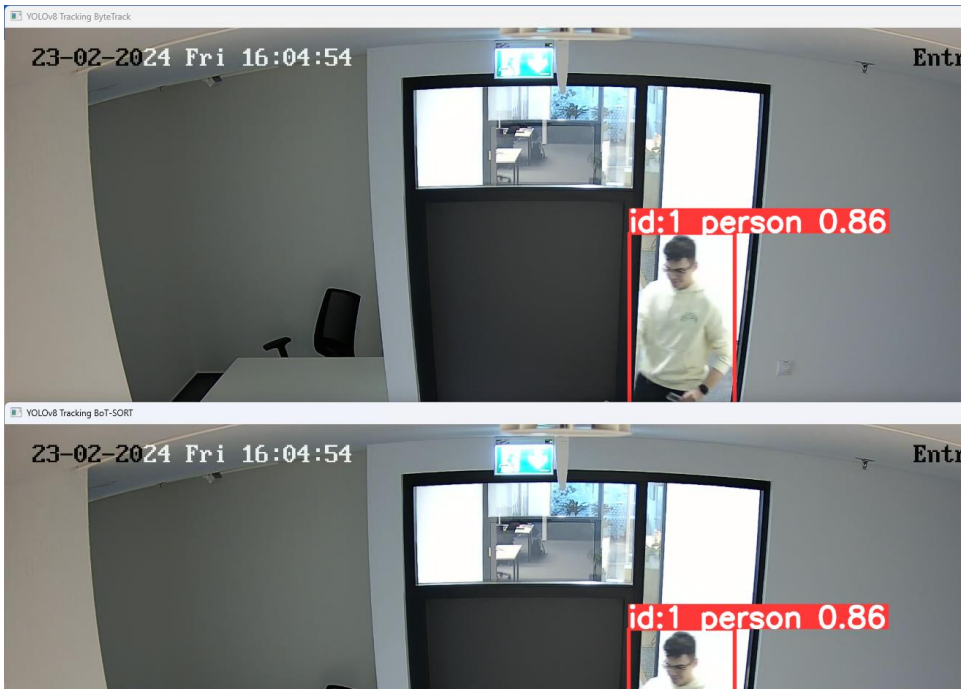


Рисунок 3.19 – Наочне тестування трекерів від Ultralytics

Також, варто зазначити, що трекер ByteTrack набагато легше налаштовувати, ніж трекер BoT-SORT. Якщо застосунок буде розширюватися у майбутньому, з метою покращення трекерів, потрібно буде змінювати трекери всередині їх налаштувань, даний трекер буде легше адаптувати під власні задачі, ніж трекер BoT-SORT.

3.5.2 Експерименти із вибору моделі детекції від Ultralytics

Даному плагіну необхідно буде працювати із ведеопотоком у реальному часі. Саме тому, для даного застосунку, питання швидкості і точності обробки є досить важливим.

На даним момент, існує YOLOv9 [33], але вона досі знаходиться у розробці, а отже це свідчить про те, що Ultralytics не гарантує стабільну

версію даної моделі. Загалом, нові версії YOLO від Ultralytics зосереджувалися на покращенні точності виявлення об’єктів, швидкості роботи моделі, ефективності використання пам’яті та простоти використання. Саме тому, доцільно використовувати моделі YOLOv8, які уявляють собою найстабільнішу, та найточнішу версії моделей від Ultralytics, серед стабільних версій моделей детекції.

Моделі YOLOv8 мають різні типи задач. Для вирішення задачі даного плагіну, потрібно було обрати одну із великої кількості моделей YOLO. Моделі для вирішення задач детекції, сегментації, та визначення поз було обрано для проведення тестування. Потрібно було визначити, моделі для яких задач краще підходили для даного застосунку. На першому етапі розробки даного плагіну, було використано моделі для кожної, наведеної вище задачі. Спочатку, було використано моделі сегментації. Для специфікації використання даної моделі, необхідно визначити її використання у програмного коді Back End частини застосунку. На рисунку 3.20, зображено специфікація використання моделі сегментації у коді:

```
model = YOLO('yolov8n-seg.pt')
```

Рисунок 3.20 – Специфікація використання моделі сегментації YOLOv8

На рисунку 3.21, зображено тестування моделі сегментації, при роботі трекера від Ultralytics із відеопотоком. У ході тестування даної моделі, стало зрозуміло, що даний тип моделей сильно збільшує час обробки кадру, навіть наймаліший розмір даної моделі, не допоміг пришвидшити обробку кадрів. У порівнянні з іншими типами моделей, дані моделі визначають певні сегменти – це області, які відображаються дуже чіткими мережами між об’єктом інтересом, та іншими об’єктами на зображеннях. Але швидкість обробки зображення, та занадто велике використання ресурсів комп’ютера роблять вибір даної моделі – нераціональним.

Рисунок 3.21 – Тестування роботи моделі сегментації

Наступним кроком – було тестування даного плагіну, під час роботи із моделями детекції поз (yolov8n-pose.pt). Під час тестування, людина змінювала власний одяг на інший, аби повністю поміняти свій образ, а також, перетиналась із іншою людиною, аби визначені бокси у об’єкту переплуталися між собою місцями. Було виявлено, що дана модель найкраще підходить для даних випадків, вона найкраще серед усіх інших моделей відслідковувала об’єкти, котрі значно змінюють свій образ. Але вона використовує багато ресурсів комп’ютера, та час на виконання обробки трекінгу значно зростав, у порівнянні із моделями детекції, які найшвидше виконували поставлені задачі. На рисунку 3.22 зображено тестування даної моделі, перша людина була одягнена у білу куртку, а на рисунку 3.23, друга людина одягла її, та перетнулася із першою людиною.

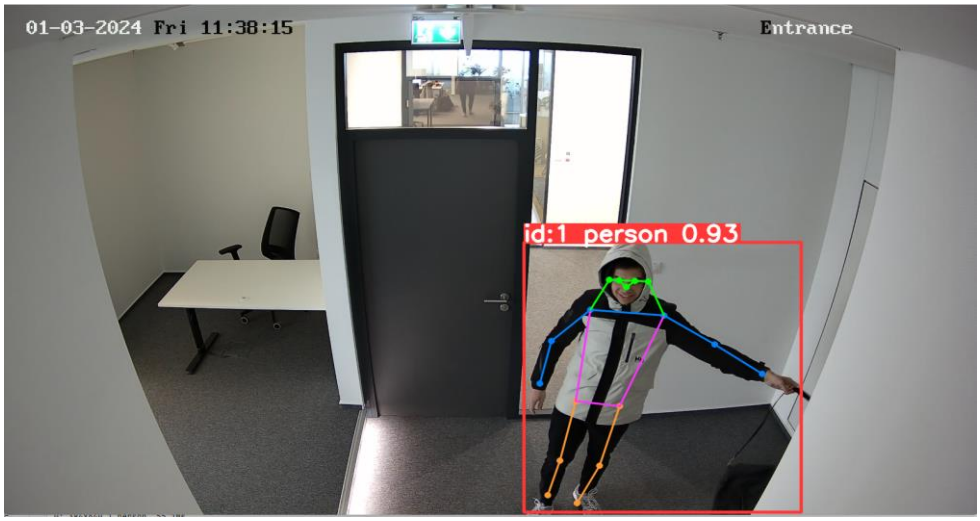


Рисунок 3.22 – Детекція першої людини у білій куртці

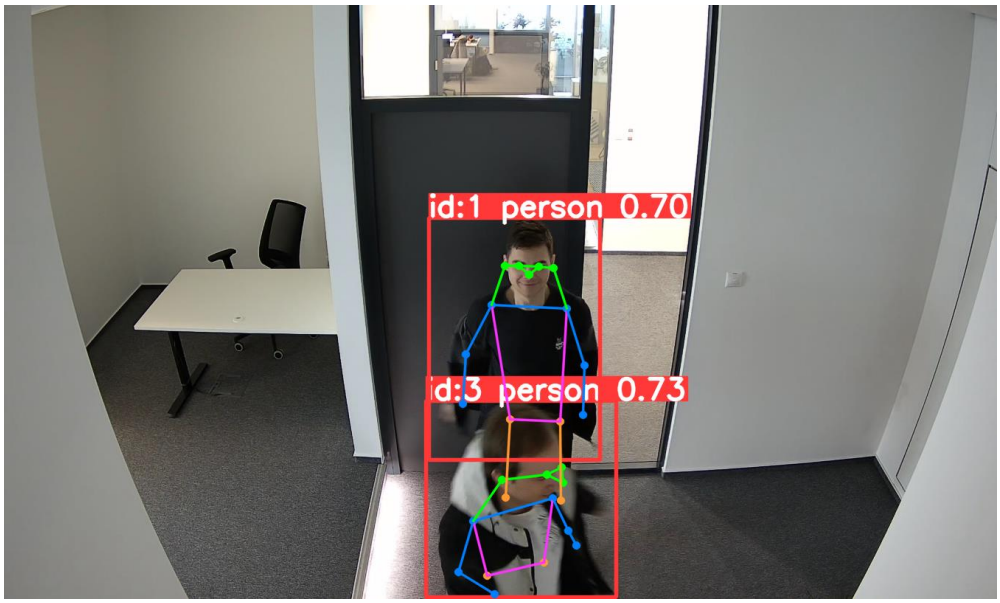


Рисунок 3.23 – Збереження id людини, незважаючи на зміну одягу

У даному випадку, коли камера направлена на світлу кімнату, яка може змінювати свою яскравість під час дня, де чітко видно відкриті двері, де об’єкти у більшості випадків будуть лише виходити, або заходити у будівлю, моделі детекції поз використовують занадто багато ресурсів комп’ютера, та обробка їх трекером займає забагато часу. Дані моделі краще підходять для детекціїї об’єктів у нестандартному положенні, наприклад, коли людина впала, або сидить. На рисунку 3.24 зображено детекцію сидячої людини, використовуючи модель yolov8n-pose.pt. На рисунку 3.25, використовується

модель модель yolov8n.pt. Порівнявши дані рисунки, видно, що детекція сидячих об’єктів працює краще на моделі детекції поз, ніж на стандартній моделі детекції. Було виявлено дві людини, за допомогою моделі детекції поз, коли за допомогою стандартної моделі детекції, було виявлено лише одну людину, та впевненість у тому, що знайдений об’єкт – це людина, була досить низька.

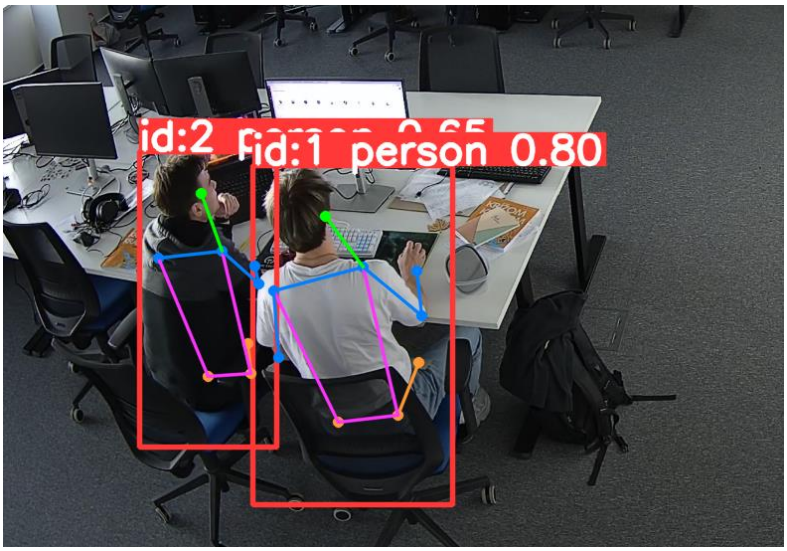


Рисунок 3.24 – Детекція сидячої людини, використовуючи модель детекції поз

Рисунок 3.25 – Детекція сидячої людини, використовуючи стандартну модель детекції

У ході експериментів із вибором моделей, стандартна модель детекції працювала найшвидше серед усіх трьох моделей. Також, вона досить точно визначала об’єкти на зображені. Для даної ситуації, стандартна модель детекції ідеально підходила для виконання даної задачі. Для виконання даної задачі, саме час виконання був критично важливим, бо даний плагін працює у режимі реального часу, та має швидко детермінувати об’єкти, та відслідковувати їх. Модель детекції об’єктів має різні розміри (рис. 3.26). Для даного застосунку, модель yolov8n.pt найкраще підходила для використання тому, що вона була найшвидша серед усіх інших моделей, проте, даний розмір моделі був найлегшим. Використання більшого розміру впливало на швидкість застосунку, саме тому, було прийнято рішення використати найменший розмір моделі.

Рисунок 3.26 – Розміри моделі детекції YOLOv8

3.6 Практичне застосування плагіну

Даний застосунок буде корисний у багатьох сфера життєдіяльності. Back End частина надає доступ до бази даних MySQL, в якій буде міститися інформація про всі входи та виходи користувачів протягом часу, коли даний плагін буде використовуватися. Нижче, на рисунку 3.27 зображена база даних monitoring_db.

id	isEntered	DateTime	peopleCounter
192	0	2024-05-09 10:01:16	5
193	0	2024-05-09 10:01:16	4
194	0	2024-05-09 10:01:16	3
195	1	2024-05-09 10:01:17	4
196	1	2024-05-09 10:01:17	5
197	1	2024-05-09 10:01:17	6
198	1	2024-05-09 10:01:18	7
199	1	2024-05-09 10:01:18	8
200	1	2024-05-09 10:01:18	9
201	1	2024-05-09 10:01:18	10
202	1	2024-05-09 10:01:18	11
203	1	2024-05-09 10:01:18	12
204	1	2024-05-09 10:01:18	13
205	1	2024-05-09 10:01:18	14
206	1	2024-05-09 10:01:19	15
207	0	2024-05-09 10:06:41	14
208	1	2024-05-09 10:16:19	15
209	1	2024-05-09 11:41:53	16

Рисунок 3.27 – База даних monitoring_db

Дану базу даних можуть використовувати будь-які сервіси, які можуть підключатися до бази даних MySQL. Варто зазначити, що дана база даних вже використовується Front End частиною плагіна, який за допомогою бібліотеки mysql-connector-java, яку використовує бібліотека JDBC, підключається до бази даних да використовує її для надання певного звіту користувачу у реальному часі (рис. 3.28).

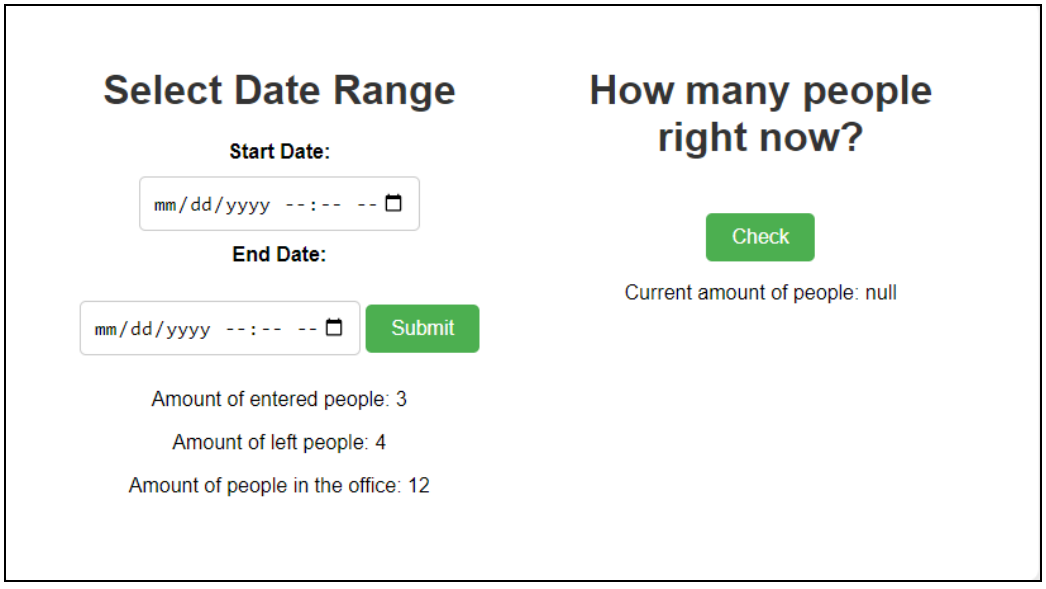


Рисунок 3.28 – Звіт від Front End частини застосунку

Використовуючи надану аналітику, користувач одразу буде мати змогу зрозуміти кількість людей, які увійшли, вийшли та кількість людей, що залишились у будівлі.

Надані дані із бази даних можливо використовувати у сферах охорони здоров'я. Наприклад, якщо у будівлі сталася надзвичайна ситуація, дані із бази даних можливо буде використати рятівними службами. Вони будуть мати можливість одразу визначити кількість осіб, які знаходяться, або знаходилися у будівлі, та кількість осіб, яка змогла покинути приміщення. Варто зазначити, що використання даного плагіну під час надзвичайної ситуації, наприклад – пожежі сильно залежить від працездатності обладнання, якщо камера та підключення буде працювати належним чином, то рятівні служби зможуть використовувати даний плагін. Якщо камеру було пошкоджено, наприклад – вибухом, або підключення було перервано, то даний плагін працює як звітній механізм, який буде мати чіткий звіт кількості людей, які були у будівлі на момент надзвичайної ситуації.

Також, використовуючи застосунок для камер відеоспостереження iSpy, рятувальні служби будуть мати змогу ідентифікувати людей, які вийшли та прийшли у будівлю. На рисунку 3.29, зображено сторінку timeMachine від сервісу iSpy, яку використовуючи, користувач зможе легко пересуватися між записаними відео.

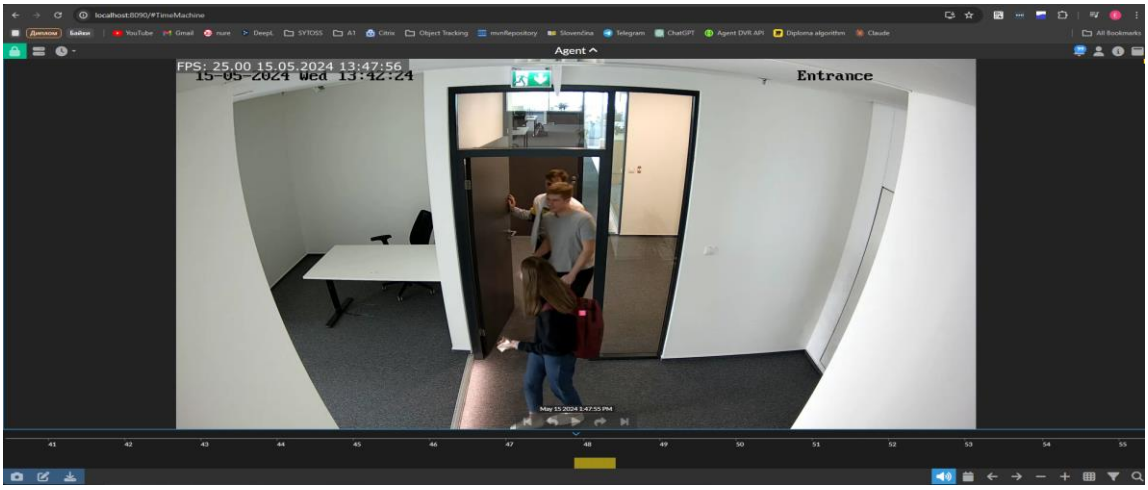


Рисунок 3.29 – TimeMachine від iSpy

Також, застосунок iSpy має сторінку Timeline, яка використовується для зручної навігації через записані відео із камери відеоспостереження. На рисунку 3.30, зображено дану сторінку. Дану сторінку найзручніше переглядати, якщо потрібно швидко визначити час, коли камера робила запис у базу даних. Також, за допомогою колеса мишки, можливо змінити масштаб зображення.

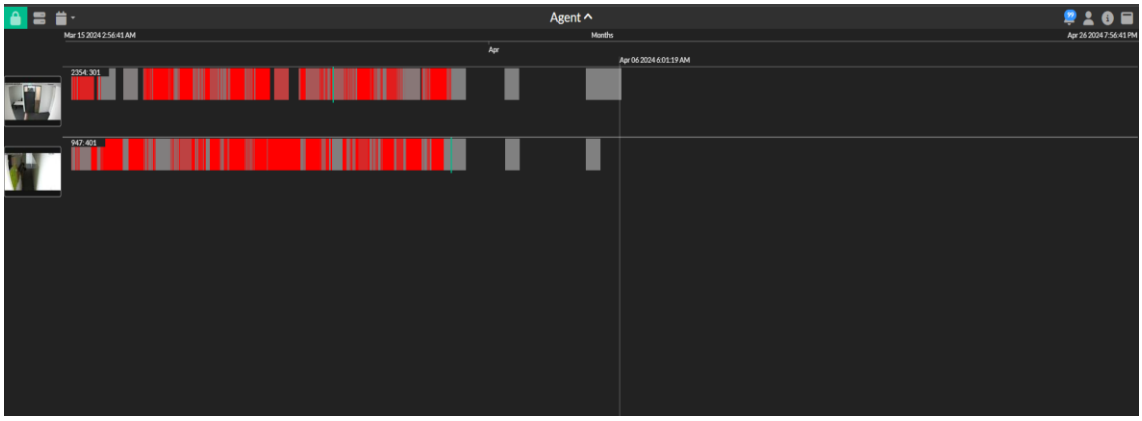


Рисунок 3.30 – Timeline у iSpy

За допомогою сторінки Recording, рятівні служби мають змогу переглянути список файлів, які є записами із відеоспостереження, та управляти ними, видаляючи, або завантажуючи на хмарне сховище iSpy. На рисунку 3.31, представлено сторінку Recording у iSpy.

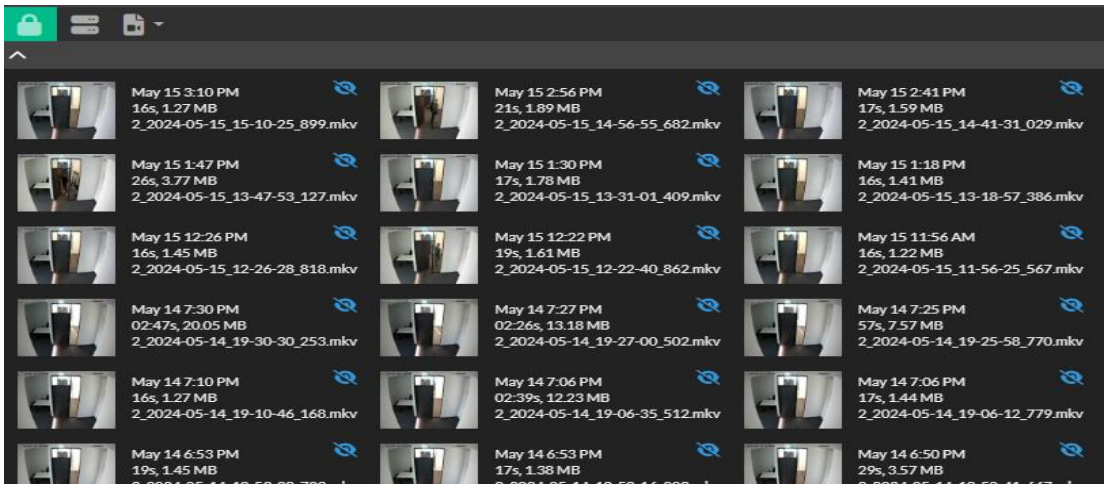


Рисунок 3.31 – Recording у iSpy

Базу даних даного плагіна можливо використовувати для надання даних для аналітиків, які згодом, можуть використовувати ці дані у власних потребах. Наприклад, ці дані можуть бути інтегровані у сервіс Celonis. Celonis — це платформа для інтелектуального аналізу та управління бізнес-процесами за допомогою аналізу даних і технологій штучного інтелекту. Celonis широко використовується в різних галузях, таких як фінансові послуги, виробництво, логістика, телекомунікації та державний сектор, для підвищення ефективності бізнес-операцій, скорочення витрат та поліпшення обслуговування клієнтів. Дані із бази даних можуть бути експортовано через найпопулярніші формати для експорту даних, або через пряме підключення до бази даних MySQL. На рисунку 3.32, одна із статистик сервісу Celonis, який використовує дані із бази даних плагіна.

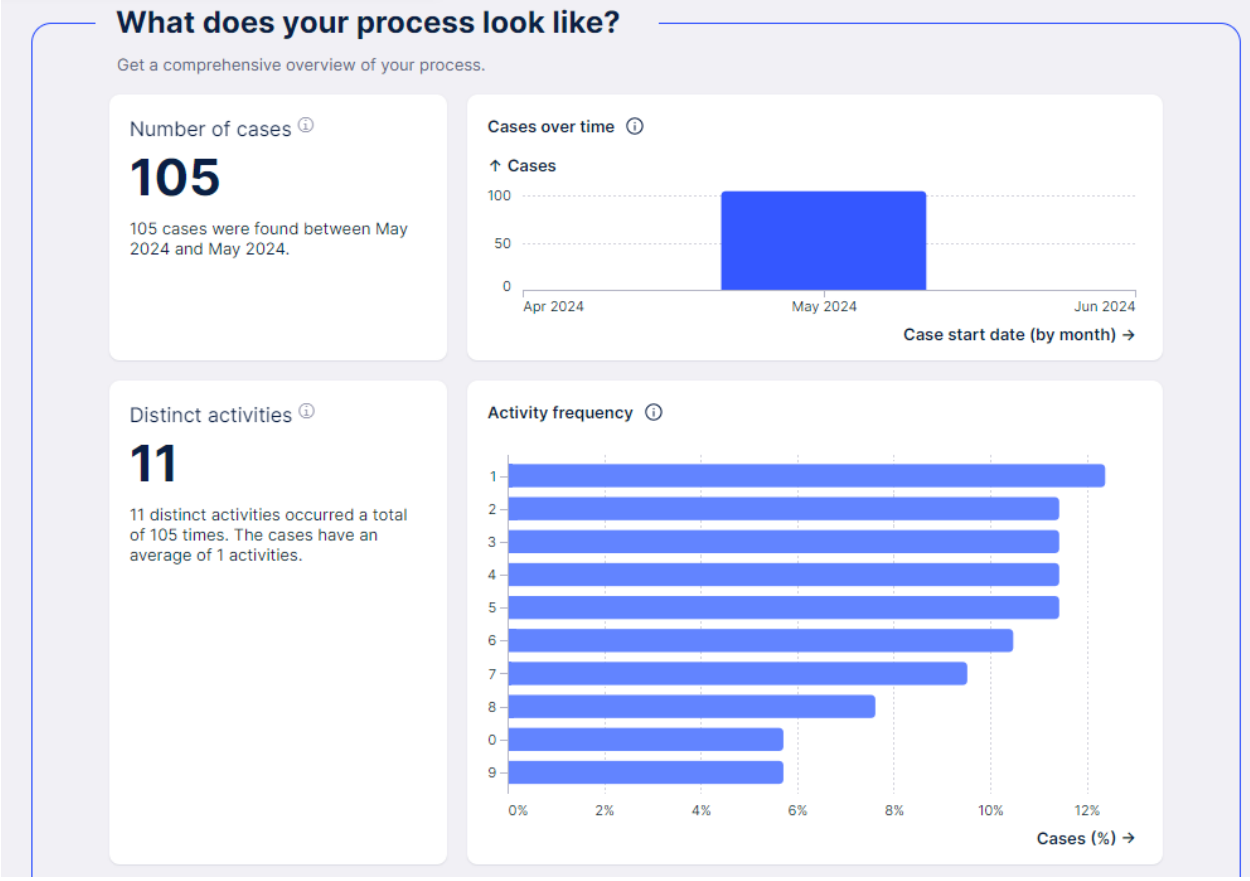


Рисунок 3.32 – Звіт, автоматично сгенерований Celonis

На даному звіті (рис. 3.32), можливо побачити, що всього, з моменту запуску програми, було 105 записів у базу даних, з яких найчастіша кількість людей у приміщенні була – одна людина. Один із найрідших випадків – це перебування в офісі дев'яти людей. Також, можливо задати кількість людей, що є небажаною у приміщенні, та за допомогою Celonis, відслідковувати кількість випадків, коли кількість людей перевищувала небажане число. Наприклад, на рисунку 3.33, була поставлена задача, що якщо людей у приміщенні більше п'яти, то це не бажана кількість людей, та на даному рисунку Celonis відобразив результат аналітики.

Рисунок 3.33 – Відсоток небажаної кількості людей у приміщенні

Сервіс Celonis можливо легко конфігурувати під власні потреби. Отже, аналітики, використовуючи надані дані із бази даних, будуть мати можливість проводити аналітику під будь-які потреби, використовуючи даний сервіс. Наприклад, Celonis, використовуючи дані із бази даних плагіну, дозволяє визначати критичні кількості людей, у приміщенні, та визначати масові паніки, або натовпи. Визначення кількості людей у приміщенні

дозволить забезпечити їхню безпеку та вжити необхідні заходи своєчасно для запобігання травм та загибелі.

Також, надані дані із бази даних плагіну можливо використовувати для забезпечення ефективного управління приміщенням. Наприклад, за допомогою наданих даних, можливо підрахувати оцінку завантаженості приміщення, або згодом вжити необхідних заходів для оптимізації використання приміщення людьми.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено плагін для визначення кількості відвідувачів на основі технології трекінгу з інтеграцією в сервіс відеоспостереження iSpy. У ході виконання кваліфікаційної роботи, було проаналізовано застосунки для камер відеоспостереження, знайдено переваги та недоліки найпопулярніших застосунків, розглянуто методи детекції та трекінгу об’єктів, та проаналізовано бібліотеки та фреймворки для їх імплементації. Також, було спроектовано архітектуру плагіну для визначення кількості відвідувачів, розроблено алгоритм для детекції входу та виходу людини із будівлі, вивчено особливості інтеграції у сервіс для камер відеоспостереження iSpy, за допомогою iSpy API. Також, було спроектовано базу даних для зберігання інформації, щодо визначених входів, виходів, та поточної кількості людей у приміщенні.

У ході тестування розробленого плагіну, були проведені експериментальні дослідження, щоб визначити, яка комбінація моделей детекції та трекінгу з фреймворку Ultralytics якнайкраще підходять для задачі підрахунку кількості людей у приміщенні в реальних умовах. Експерименти проводилися на власному датасеті, зібраному з камер відео спостереження. Серед великої кількості моделей для різних задач, було обрано комбінацію з моделі детекції yolov8n-pt та трекеру byteTrack, сумісне використання яких дозволило проводити підрахунок людей на робочому датасеті в реальному часі з високою точністю (accuracy=0,97). Випадки, в яких плагін мав помилки у коректному визначені людей, що входять або виходять, пов’язані насамперед з наявністю значних оклюзій, де зображення людей значно перетиналися, або повністю затуляли один одного. Але відсоток випадків значних оклюзій у робочому датасеті складав лише 10%. У подальшому необхідно більш детально дослідити це питання та модернізувати алгоритм для покращення його роботи у випадках значних оклюзій.

Також у ході тестування даного застосунку, було визначено, які параметри системи варто визначити, для забезпечення найефективнішої роботи плагіну. Даний плагін має системні параметри, які задаються користувачем власноруч в середині програмного коду, у майбутньому, вони мають бути адаптовані таким чином, щоб користувач мав змогу використовувати графічний інтерфейс для задання даних параметрів.

Розроблений плагін було інтегровано із сервісом Celonis, для забезпечення ефективної звітності, використовуючи дані із бази даних застосунку.

Даний плагін має значний потенціал у його використанні у різних сферах, де існує задача підрахунку кількості відвідувачів у приміщенні, наприклад, у торгових залах, офісах, навчальних аудиторіях, кінотеатри, стадіони. А також, даний застосунок може бути використаний для сфери охорони здоров’я та для служб порятунку, своєчасно забезпечуючи необхідною інформацією. Даний плагін має великий потенціал для надання аналітичної інформації користувачам, які працюють у сферах менеджменту будівлі, аби вони мали змогу більш ефективно використовувати робочий простір.

Результати роботи апробовано у вигляді 2 тез доповідей під час XV-ої Міжнародної науково-практичної конференції FOSS-2024 «Free and open source software» [34], Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті», де робота зайняла 1 місце [35].

Також результати досліджень були представлено на Студентському науковому конкурсі з міжнародною участю в Братиславському університеті економіки та менеджменту (Bratislava University of Economics and Management, BUEM), де робота в категорії рівня бакалавр посіла перше місце, та основні результати будуть опубліковано в Науковому журналі BUEM (робота проходить стадію рецензування [36]).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, Indonesian Journal of Electrical Engineering and Computer Science, vol. 33, no. 1, pp. 113-125. DOI: 10.11591/ijeecs.v33.i1.pp113-125.

2. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. CEUR Workshop Proceedings Vol. 3403. pp. 69-86. ISSN 1613-0073 https://ceur-ws.org/Vol-3403/paper6.pdf.

3. iSpy application review. URL: https://www.ispyconnect.com/ (дата звернення 14.03.2024).

4. How CCTV security cameras work. URL: https://www.atotalsolution.com/blog/your-complete-guide-to-cctv-security-cameras/ (дата звернення 15.03.2024).

5. Xeoma review. URL: https://felenasoft.com/xeoma/en/ (дата звернення 17.03.2024).

6. Xeoma features. URL: https://felenasoft.com/xeoma/en/features/ (дата звернення 17.03.2024).

7. Frigate review. URL: https://frigate.video/ (дата звернення 17.03.2024).

8. iSpy motion detection. URL: https://www.ispyconnect.com/docs/ispy/motion-detection# (дата звернення 18.03.2024).

9. iSpy customer tracker. URL: https://www.iplugs.eu/pl_tracker.html (дата звернення 22.03.2024).

10. Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Scientific Journal of Bratislava University of Economics and

Management «Public Administration and Regional Development, Economics, Management and Marketing», 19(1), pp. 158-183.

11. Sharma, V. K., & Mir, R. N. (2020). A comprehensive and systematic look up into deep learning based object detection techniques: A review. Computer Science Review, 38, 100301.

12. Security cameras overview. URL: https://reolink.com/blog/cctv-camera-types/ (дата звернення 22.03.2024).

13. Zhang, Y., Wang, T., Liu, K., Zhang, B., & Chen, L. (2021). Recent advances of single-object tracking methods: A brief survey. Neurocomputing, 455, 1-11.

14. Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2021). Multiple object tracking: A literature review. Artificial intelligence, 293, 103448.

15. Multi object tracking. URL: https://www.researchgate.net/publication/333524664_Multi-Person_Tracking_Based_on_Faster_R-CNN_and_Deep_Appearance_Features (дата звернення 23.03.2024).

16. Яковлева, О. В., & Кускова, І. В. (2006). Дослідження результатів сегментації зображень методом матриць збігів. Вісник Національного технічного університету «ХПІ». №39 - С.164 -171.

17. А.Р. Ковтуненко, О.В. Яковлева, В.А. Любченко, & О.В. Янголенко (2020) Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників. Вісник Національного технічного університету ХПІ (3). 24-31.

18. Yakovleva O., Nebeský L, Liakhov P. (2023). Research methods of texture image analysis to solve the texture search problem. Proceedings of the IV International Scientific and Practical Conference "The world of modern technologies and inventions". Vienna, Austria. 2023. pp. 252-261 URL: https://isg-konf.com/the-world-of-modern-technologies-and-inventions/.

19. Basic concept of Kalman filtering. URL: https://uk.wikipedia.org/wiki/Файл:Basic_concept_of_Kalman_filtering.svg (дата звернення 28.03.2024).

20. Object tracking problems. URL: https://encord.com/blog/object-tracking-guide/ (дата звернення 05.04.2024).

21. DeepSORT algorithm. URL: https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/ (дата звернення 06.04.2024).

22. MDNet algorithm. URL: https://arxiv.org/pdf/1510.07945 (дата звернення 06.04.2024).

23. GOTURN algorithm. URL: https://davheld.github.io/GOTURN/GOTURN.pdf (дата звернення 06.04.2024).

24. iSpy API. URL: https://ispysoftware.github.io/Agent_API/ (дата звернення 08.04.2024).

25. Ultralytics review. URL: https://docs.ultralytics.com/ (дата звернення 10.04.2024).

26. ByteTrack metrics. URL: https://github.com/ifzhang/ByteTrack (дата звернення 11.04.2024).

27. BoT-SORT metrics. URL: https://github.com/NirAharon/BoT-SORT (дата звернення 11.04.2024).

28. Python, W. (2021). Python. Python releases for windows, 24.

29. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

30. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

31. Bloch, J. (2017). Effective java. Addison-Wesley Professional.

32. Reese, G. (2000). Database Programming with JDBC and JAVA. " O’Reilly Media, Inc.".

33. Ultralytics YOLOv9. URL: https://docs.ultralytics.com/models/yolov9/ (дата звернення 18.04.2024).