МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ І. М. Журавська

« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV

Спеціальність 123 Комп'ютерна інженерія

123 – КБР.ПЗ.00 – 405.21910521

Студент _____О. О. Михайлов

підпис

«__» _____ 202__ р.

Керівник д-р техн. наук, проф. _____ С. В. Пузирьов

підпис

«__» _____ 202__ р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної бакалаврської роботи

Видано студенту групи 405 факультету комп'ютерних наук

Михайлову Олександру Олександровичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
__Автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV

Затверджена наказом по ЧНУ від «__» _____ 2023 р. № _____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
___Очікуваним результатом роботи є: система, яка буде в реальному часі розпізнавати обличчя з відеопотоку IP-камери та зіставляти з моделлю з датасету. Вхідними даними роботи є фото та скріншоти з обличчям людини та відеопотік з IP-камери.

4. Перелік питань, що підлягають розробці
1) огляд ринку технологій відеоспостереження за приміщенням та сфери їх застосування;
2) огляд можливостей IP-камер та способи їх використання й підключення;
3) огляд можливостей OpenCV;
4) моделювання приміщення з рахунком встановлених камер;
5) підключення IP-камер;
6) розробити алгоритм розпізнавання обличь методами Python та OpenCV;

7) розробити систему моніторингу приміщення за допомогою IP-камер та OpenCV;

5. Перелік графічних матеріалів

Слайди презентації

6. Завдання до спеціальної частини

1) дослідити вимоги щодо облаштування робочих зон

2) визначити правила пожежної безпеки для закритого приміщення

3) визначити поняття та основні засади відеоспостереження на робочому місці

4) визначити законність відеоспостереження роботодавцем за своїми працівниками

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва, к.т.н., доцент	кафедра екології Медичного інституту ЧНУ імені Петра Могили	Спеціальна частина з охорони праці

Керівник роботи

канд. фіз.-мат. наук, доцент Пузирьов Сергій Володимирович

(посада, прізвище, ім'я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

Михайлов Олександр Олександрович

(прізвище, ім'я, по батькові студента)

(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної бакалаврської роботи

Тема: Автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	20.12.2022	26.12.2022	Виконано
2.	Огляд літератури за темою роботи	21.01.2023	06.02.2023	Виконано
3.	Складання календарного плану КБР	08.02.2023	09.02.2023	Виконано
4.	Аналіз предметної області	11.02.2023	17.02.2023	Виконано
5.	Розробка проєктних рішень	18.02.2023	27.03.2023	Виконано
6.	Моделювання апаратної частини	28.03.2023	16.04.2023	Виконано
7.	Розробка програмного забезпечення та тестування	17.04.2023	17.05.2023	Виконано
8.	Написання розділу з охорони праці	18.05.2023	22.05.2023	Виконано
9.	Відгук керівника КБР	23.05.2023	24.05.2023	Виконано
10.	Оформлення КР та презентації	24.05.2023	29.05.2023	Виконано
11.	Перший передзахист КБР	30.05.2023	30.05.2023	Виконано
12.	Другий передзахист КБР	13.06.2023	13.06.2023	Виконано
13.	Завершення оформлення КБР та презентації	14.06.2023	17.06.2023	Виконано
14.	Рецензування	17.06.2023	18.06.2023	Виконано
15.	Захист кваліфікаційної роботи	27.06.2023	28.06.2023	Виконано

Розробив здобувач ВО __Михайлов Олександр Олександрович _ _____

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 2023 р.

Керівник роботи _канд. фіз.-мат. наук, доцент Пузирьов С. В.___ _____

(підпис)

«__» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV»

Студент гр. 405 Михайлов Олександр Олександрович

Керівник: канд. фіз.-мат. наук, доцент Пузирьов С. В.

Бакалаврська робота досліджує можливості автоматизації моніторингу приміщень з використанням IP-камер та бібліотеки OpenCV. IP-камери, підключені до мережі Інтернет, дозволяють отримувати доступ до зображень з будь-якого місця з Інтернет-підключенням, забезпечуючи ефективний контроль і безпеку приміщень. OpenCV, надійна бібліотека комп'ютерного зору, надає інструменти для обробки зображень, виявлення об'єктів, розпізнавання руху та багато іншого. Розглянуто види IP-камер, їх характеристики й області застосування. Практичне значення одержаних результатів полягає у створенні системи моніторингу приміщення за допомогою IP-камер та бібліотеки OpenCV з розпізнаванням обличчя.

Мета: розроблення системи автоматизованого моніторингу приміщень, використовуючи IP-камери та бібліотеку OpenCV з розпізнаванням обличчя.

Поставлена мета вимагає вирішеннях таких **завдань**:

1) дослідження спектру застосування систем моніторингу приміщень за допомогою IP-камер і OpenCV;
2) аналітичний огляд літератури та документації про IP-камери;
3) обґрунтування вибору моделей камер;
4) налагодження та обробка принципів встановлення IP-камер в приміщенні;
5) розроблення програмного забезпечення для розпізнавання обличь.

Об’єкт: методи та технології налаштування IP-камер в приміщенні та розпізнавання обличь на базі OpenCV.

Предмет: система зі зв’язкою з декількох IP-камер з підтримкою програмного забезпечення, реалізованого за допомогою бібліотеки OpenCV, що може розсновувати обличчя.

Практичне значення одержаних результатів полягає у створенні системи моніторингу приміщення за допомогою IP-камер та бібліотеки OpenCV з розпізнаванням обличчя.

Кваліфікаційна бакалаврська робота містить: перелік скорочень, вступ, три розділи, висновок, перелік джерел посилання та три додатки.

У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет та завдання дослідження, які необхідно виконати для досягнення поставленої мети.

В першому розділі проведено аналіз сфер застосування систем відео нагляду за приміщеннями в сучасному світі. Описано можливості та особливості IP-камер. Коротко описано способи обробки відео. Виокремлено правила та вимоги для систем моніторингу приміщення.

Другий розділ містить опис процесу обрахування та моделювання системи моніторингу приміщення та налагодження апаратної частини.

У третьому розділі описано розробку програмного забезпечення за допомогою OpenCV. Описано методи розпізнавання обличчя.

У висновку описано результати виконання кваліфікаційної роботи.

Додатки містять код програмного забезпечення та звіт з перевірки роботи на плагіат.

В спеціальній частині з охорони праці розглянуто вимоги щодо облаштування робочих зон, правил пожежної безпеки для закритих приміщень, поняття та основні засади відеоспостереження на робочому місці

Кваліфікаційна робота містить 70 сторінку (без додатків), 50 рисунків, 18 джерел посилання, 3 додатки.

Ключові слова: IP-камер, OpenCV, комп’ютерний зір, розпізнавання, Інтернет-підключення.

ABSTRACT

of the Bachelor’s Thesis

"Automation of premises monitoring using IP cameras and OpenCV"

Student: Mikhailov Oleksandr Oleksandrovych

Supervisor: Candidate of Physical and Mathematical Sciences, Associate Professor Puzyrev S. V.

The bachelor's thesis explores the possibilities of automating the monitoring of premises using IP cameras and the OpenCV library. IP cameras connected to the Internet allow access to images from any location with an Internet connection, providing effective control and security of premises. OpenCV, a robust computer vision library, provides tools for image processing, object detection, motion recognition, and much more. The types of IP cameras, their characteristics, and areas of application are considered. The practical significance of the results obtained is to create a room monitoring system using IP cameras and the OpenCV library with face recognition.

Objective: to develop an automated room monitoring system using IP cameras and the OpenCV library with face recognition.

This goal requires the following tasks:

1) studying the range of applications of indoor monitoring systems using IP cameras and OpenCV;

2) analytical review of literature and documentation on IP cameras;

3) justification of the choice of camera models;

4) setting up and processing the principles of installing IP cameras in the premises;

5) development of software for face recognition.

Object: methods and technologies for setting up indoor IP cameras and face recognition based on OpenCV.

Subject: a system with a connection of several IP cameras with the support of software implemented using the OpenCV library that can detect faces.

The practical significance of the results is to create a room monitoring system using IP cameras and the OpenCV library with face recognition.

The bachelor's thesis contains: a list of abbreviations, an introduction, three chapters, a conclusion, a list of references, and three appendices.

The introduction defines the relevance of the topic, formulates the purpose, object, subject, and tasks of the research that need to be performed to achieve the goal.

The first chapter analyzes the areas of application of video surveillance systems for premises in the modern world. The capabilities and features of IP cameras are described. The methods of video processing are briefly described. The rules and requirements for indoor monitoring systems are highlighted.

The second section describes the process of calculating and modeling a room monitoring system and debugging the hardware.

The third section describes software development using OpenCV. The face recognition methods are described.

The conclusion describes the results of the qualification work.

The appendices contain the software code and a report on plagiarism checking.

In the special part on labor protection, the requirements for the arrangement of work areas, fire safety rules for enclosed spaces, the concept and basic principles of video surveillance in the workplace are considered

The qualification work contains 70 pages (without appendices), 50 figures, 18 references, 3 appendices.

Keywords: IP cameras, OpenCV, computer vision, recognition, Internet connection.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ІЧ	– Інфрачервоне випромінювання
ООП	– Об'єктно-орієнтоване програмування
ПЗ	– Програмне забезпечення
AI	– Artificial Intelligence
DDNS	– Dynamic Domain Name System
HD	– High Definition
IDE	– Integrated Development Environment
IP	– Internet Protocol
LBP	– Local Binary Patterns
OpenCV	– Open Source Computer Vision Library
PCA	– Principal Component Analysis
PoE	– Power over Ethernet
SVM	– Support Vector Machine
Ultra HD	– Ultra High Definition
UPnP	– Universal Plug and Play

ВСТУП

Мабуть вже стало звичним бачити різноманітні камери в різних приміщеннях, закладах, супермаркетах, банках. У сучасному світі, де технології швидко розвиваються, автоматизація стає невід'ємною частиною нашого повсякденного життя. Одним з важливих аспектів автоматизації є моніторинг приміщень. Відстеження подій і контроль за безпекою стають все більш важливими завданнями для багатьох організацій та приватних осіб. Один із засобів, який допомагає досягти цих цілей, є використання IP-камер та бібліотеки OpenCV.

IP-камери – це веб-камери, які з'єднані з мережею Інтернет. Вони здатні записувати відео та передавати його по мережі, що дозволяє отримувати доступ до зображень з будь-якого місця, де є підключення до Інтернету. Завдяки цим камерам ми можемо відстежувати події, контролювати приміщення та забезпечувати безпеку як в домашніх умовах, так і в комерційних об'єктах.

OpenCV (Open Source Computer Vision Library) – це бібліотека відкритого програмного забезпечення, яка надає набір інструментів для обробки зображень і комп'ютерного зору. Вона пропонує широкі можливості для розпізнавання об'єктів, виявлення руху, аналізу зображень та багато іншого. Поєднання IP-камер із бібліотекою OpenCV дозволяє створити потужну систему моніторингу, яка автоматично аналізує відео та надає корисну інформацію для прийняття рішень.

У даній роботі розглянуто можливості автоматизації моніторингу приміщень за допомогою IP-камер та OpenCV. Досліджено методи виявлення об'єктів, визначення руху, розпізнавання обличь, а також інші функції, які можуть бути корисними для забезпечення безпеки та ефективного управління приміщеннями та як ці технології можуть бути застосовані в різних сферах, від домашньої безпеки до бізнесу.

Автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV відкриває широкі перспективи для забезпечення безпеки та ефективності у різних сферах. Ця технологія є потужним інструментом, який може допомогти нам більш точно спостерігати за подіями, виявляти потенційні загрози та приймати своєчасні рішення. В подальших розділах більш детально розглянуто можливості та застосування цієї технології.

Мета: розроблення системи автоматизованого моніторингу приміщень, використовуючи IP-камери та бібліотеку OpenCV з розпізнаванням обличчя.

Поставлена мета вимагає вирішеннях таких **завдань**:

1) дослідження спектру застосування систем моніторингу приміщень за допомогою IP-камер і OpenCV;
2) аналітичний огляд літератури та документації про IP-камери;
3) обґрунтування вибору моделей камер;
4) налагодження та обробка принципів встановлення IP-камер в приміщенні;
5) розроблення програмного забезпечення для розпізнавання обличь.

Об’єкт: методи та технології налаштування IP-камер в приміщенні та розпізнавання обличь на базі OpenCV.

Предмет: система зі зв’язкою з декількох IP-камер з підтримкою програмного забезпечення, реалізованого за допомогою бібліотеки OpenCV, що може розсновувати обличчя.

Практичне значення одержаних результатів полягає у створенні системи моніторингу приміщення за допомогою IP-камер та бібліотеки OpenCV з розпізнаванням обличчя.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Використання IP-камер у сучасному світі

У сучасному світі вже стало звичним бачити камери на кожному кроці, на будь-якому стовпі, у громадських приміщеннях. Звичайно, безпека займає високий рівень пріоритетності, тому для налагодження систем моніторингу приміщень важливо використовувати якісного софту та камер (рис.1.1).

Рисунок 1.1 – IP-камера внутрішнього спостереження D-Link DCS-4802E

IP-камери, також відомі як мережеві камери, є пристроями, які використовують Інтернет-протокол для передачі відео та аудіо сигналів через комп'ютерну мережу. Вони є основним компонентом систем відеоспостереження та моніторингу.

Однією з особливостей IP-камер є висока якість зображення. Вони зазвичай мають вищу роздільну здатність і якість зображення порівняно з аналоговими камерами, тому можуть записувати відео у форматі HD або навіть 4K Ultra HD, що дозволяє отримати деталізоване зображення.

IP-камери підключаються до комп'ютерної мережі, що дозволяє отримувати доступ до відео з будь-якого місця за допомогою Інтернет-

з'єднання. Це робить їх ідеальними для віддаленого спостереження та контролю.

З точки зору багатофункціональності, вони мають доволі зручні функції, такі як виявлення руху, нічна зйомка, запис, система сигналізації та інші можливості. Вони можуть надсилати сповіщення, якщо виявляється певна подія або порушення.

Камери прості у встановленні та налаштуванні. Вони можуть працювати як самостійні пристрої або бути частиною більшої системи моніторингу. Кількість камер може бути легко розширена залежно від потреб користувача [18].

IP-камери можуть інтегруватись з іншими системами, такими як системи контролю доступу, системи віддаленого доступу. Завдяки мережевому підключенню, користувач може віддалено переглядати живе відео та записи з камер з використанням комп'ютерів, смартфонів або планшетів. Це дозволяє власникам контролювати та спостерігати за своїм майном, навіть коли вони знаходяться далеко від нього.

Багато IP-камер оснащені функцією нічного бачення, що дозволяє записувати відео у темряві. Вони використовують ІЧ-підсвічування, що не видимі для людського ока, щоб освітлювати об'єкти в області зйомки (рис.1.2).

Рисунок 1.2 – Вид з камери з ІЧ-підсвічуванням

Багато камер мають змінні об'єктиви, що дозволяє користувачам вибирати об'єктив залежно від потрібного кута огляду. Це дає можливість налаштувати камеру для різних вимог та сценаріїв використання.

IP-камери можуть мати компактний дизайн та можуть бути легко приховані або встановлені в недосяжних місцях. Це дозволяє забезпечити недоступність та зберігати конфіденційність спостережуваних об'єктів.

Одним найважливішим плюсом є те, що IP-камери мають інтеграція з OpenCV. OpenCV надає широкі можливості обробки відео, аналізу об'єктів, розпізнаванню образів та багато іншого. Це відкриває шлях до розширення можливостей IP-камер за допомогою обробки відеоданих з використанням OpenCV.

Камери, спільно з OpenCV, можуть використовуватись для виявлення та відстеження об'єктів у потоці відео. Це може бути корисно в ситуаціях відеоспостереження, де потрібно відстежити або знайти потрібні нам об’єкти, наприклад, відрізнити собаку від стола або знайти на відео тільки машини.

IP-камери можуть використовуватись для розпізнавання обличь з використанням OpenCV. Це може мати застосування в системах контролю доступу, де можна автоматично ідентифікувати людей на основі їхнього обличчя [5].

Також за допомогою такого симбіозу можна робити аналіз поведінки. Наприклад, можна виявити небажану активність або незвичайні дії, такі як відкладання предметів, рух в заборонені зони або неправильне поводження.

Цікавою й надзвичайно складною темою для AI є визначення емоцій. З використанням OpenCV, IP-камери можуть аналізувати обличчя людей на наявність емоцій [4].

Також IP-камери можуть використовуватись для моніторингу та оптимізації споживання електроенергії в приміщеннях. Наприклад, за допомогою аналізу відео можна визначити активні та неактивні зони, що

дозволяє автоматично регулювати освітлення, опалення або кондиціонування повітря залежно від присутності людей.

Віртуальна реальність та розширена реальність: IP-камери в поєднанні з OpenCV можуть бути використані для створення віртуальних або розширених реальностей. Наприклад, можна розпізнавати об'єкти або мітки у реальному часі та накладати на них віртуальні елементи, такі як інформаційні плакати, напрямні стрілки або 3D-моделі.

Це лише кілька прикладів застосування IP-камер та OpenCV у сфері моніторингу приміщень. Завдяки їхній гнучкості, потужності та можливостям аналітики, системи моніторингу приміщень стають все більш розповсюдженими та інноваційними в різних галузях.

1.2 Області застосування систем моніторингу приміщень

Системи моніторингу приміщень використовуються в різних ситуаціях, де потрібно забезпечити безпеку, контроль та нагляд за приміщеннями.

Одна з найпоширеніших областей застосування систем моніторингу приміщень – це звичайне відеоспостереження. IP-камери встановлюються для запису відео з приміщень з метою нагляду, контролю та виявлення подій. Наприклад, в торгових центрах, системи відеоспостереження використовуються для контролю за касовими зонами, протидії крадіжкам та виявлення небажаної поведінки (рис.1.3) [1].

Рисунок 1.3 – Вид з камер супермаркету

В приватних будинках або квартирах, такі системи можуть виявляти несанкціонований доступ та сповіщати власників або служби безпеки(рис.1.4). У комерційних закладах, таких як банки або магазини, системи моніторингу використовуються для виявлення крадіжок, попередження злочинів і забезпечення безпеки персоналу та клієнтів [7].

Рисунок 1.4 – Вид з камери приватного будинку

Системи моніторингу приміщень можуть бути інтегровані з системами контролю доступу, що дозволяє обмежувати та контролювати доступ до певних зон. Наприклад, у корпоративних офісах, співробітникам може бути надана картка доступу, яка використовується для входу до обмежених зон.

Система моніторингу записує дані про доступ та може відслідковувати несанкціоновані спроби доступу.

Системи моніторингу можуть використовуватись для оптимізації споживання енергії в приміщеннях. Наприклад, за допомогою датчиків з можливостями вимірювання освітленості, температури та вологості, системи моніторингу приміщень можуть автоматично керувати освітленням, опаленням та кондиціонуванням повітря.

Системи моніторингу можуть бути обладнані датчиками виявлення пожежі та інших аварійних ситуацій. Вони можуть автоматично сповіщати про пожежу або витік газу, активувати сигнали тривоги та запускати системи пожежогасіння.

У промислових приміщеннях системи моніторингу використовуються для спостереження за робочими процесами та оптимізації роботи. Наприклад, виробничі лінії можуть бути обладнані камерами, що відстежують рух товарів та контролюють роботу обладнання для покращення ефективності та якості виробництва.

Також такі системи можуть збирати дані та аналізувати їх для генерації звітів та статистики щодо активності та подій у приміщеннях. Наприклад, в магазинах можуть використовуватись системи аналітики відео, щоб аналізувати поведінку покупців, їх переваги та реакції на товари з метою вдосконалення маркетингових стратегій.

У приміщеннях, пов'язаних з транспортом, системи моніторингу використовуються для контролю та моніторингу транспортних вузлів, паркінгів, аеропортів та залізничних станцій. Наприклад, системи відеоспостереження можуть допомагати відстежувати рух транспорту, виявляти незаконну паркування або випадки порушення правил дорожнього руху (рис.1.5).

Рисунок 1.5 – Вид з камери паркінгової зони

Це лише кілька прикладів областей застосування систем моніторингу приміщень, а їх потенціал безмежний і може бути адаптований до різних потреб та вимог різних галузей.

Якщо розглядати конкретні приклади застосування систем моніторингу приміщень, то навести такі приклади:

1) магазин: камери встановлюються в різних зонах магазину, обладнаних функціями розпізнавання обличь та аналізу відео. Система може виявляти в очікуваних районах незвичайну активність, таку як крадіжки або поведінку, що викликає підозру. Коли такі події виявляються, оператор отримує сповіщення, щоб прийняти відповідні заходи;

2) офісна будівля: Встановлення камер з алгоритмами виявлення руху та аналізу відео у коридорах та спільних зонах. Система може автоматично включати освітлення та вентиляцію відповідно до виявленої активності. Наприклад, якщо приміщення не використовується, система може знижувати енергоспоживання, забезпечуючи економію енергії;

3) банківське відділення: Встановлення IP-камер з функцією розпізнавання обличь для ідентифікації клієнтів та перевірки їх з внутрішньою базою даних. Система може автоматично відкривати двері для відомих

клієнтів та вимагати додаткову перевірку для незнайомих осіб, забезпечуючи високий рівень безпеки та контролю доступу;

4) готельна галузь: Встановлення IP-камер з функцією розпізнавання номерних знаків на паркінговому місці. Система може автоматично реєструвати прибуття та виїзд гостей, забезпечуючи зручну систему контролю доступу та управління паркуванням;

5) медичний заклад: Використання IP-камер та OpenCV для моніторингу критичних зон, наприклад, палат інтенсивної терапії або аптечних складів. Система може виявляти неправильну активність, наприклад, невідомих осіб у зонах обмеженого доступу, що забезпечує додатковий рівень безпеки для пацієнтів та медичного персоналу;

6) виробничий комплекс: Встановлення IP-камер з алгоритмами виявлення пожежі та диму. Система може автоматично сповіщати пожежну службу та активувати систему пожежогасіння при виявленні вогню або диму, що допомагає швидко реагувати на пожежну небезпеку та запобігати поширенню вогню.

Ці приклади демонструють різноманітність можливостей застосування автоматизації моніторингу приміщень з використанням IP-камер та OpenCV. Ці технології відкривають широкі перспективи для покращення безпеки, ефективного управління та оптимізації різних сфер діяльності.

1.3 Засоби обробки відео

Навіть якщо ми встановимо собі IP-камеру, то нам необхідно якимось чином обробляти дані, що ми отримуємо з систем моніторингу. На допомогу приходить чудова бібліотека OpenCV.

OpenCV – це відкрите програмне забезпечення, яке надає набір функцій і алгоритмів для комп'ютерного зору та обробки зображень в реальному часі. Воно було розроблено з метою створення ефективних та доступних інструментів для розробників у галузі комп'ютерного зору.

OpenCV написано на C++ і має бібліотеки для розвитку на різних мовах програмування, включаючи Python та Java. Вона працює на різних операційних системах, таких як Windows, Linux, macOS, Android та iOS.

Основні функціональні можливості OpenCV включають:

1) зчитування та запис зображень і відео: OpenCV може завантажувати зображення з різних форматів (наприклад, JPEG, PNG) та відео з веб-камери, IP-камери або файлів. Вона також дозволяє зберігати зображення та відео у різних форматах;

2) обробка та аналіз зображень: OpenCV має вбудовані функції для обробки та аналізу зображень, таких як зміна розміру, зміна контрастності, розмиття, виявлення країв, видалення шуму та інші. Вона також надає інструменти для роботи з кольором, включаючи конвертацію кольорових просторів та виявлення об'єктів за кольором;

3) візуалізація та відображення зображень: OpenCV дозволяє відображати зображення та відео, а також створювати графічні інтерфейси користувача для взаємодії зі зображеннями. Це корисно для відлагодження та візуалізації результатів обробки зображень;

4) виявлення та відстеження об'єктів: OpenCV має функції для виявлення об'єктів у потоці відео або зображенні. Це включає детектори обличь, людей, автомобілів, руху та багато іншого. Крім того, OpenCV надає можливість відстежувати об'єкти в часі, визначати їхню траєкторію та передбачати майбутнє розташування;

5) комп'ютерне зорове співставлення: OpenCV містить алгоритми для вирішення завдань співставлення зображень, таких як знаходження збігів, відповідностей та розв'язання задач геометричного співставлення. Це корисно для створення систем аугментованої реальності, реконструкції 3D-сцен або знаходження об'єктів на основі їхнього вигляду;

6) обробка відеопотоку в реальному часі: OpenCV забезпечує можливість обробляти відеопотік в реальному часі, що робить його ідеальним

інструментом для застосувань, які вимагають низької затримки. Це може бути відеоспостереження, системи безпеки, розпізнавання жестів, доповненої реальності та багато інших;

7) сегментація зображень: OpenCV надає інструменти для сегментації зображень, тобто розділення зображення на окремі області або об'єкти. Це може бути корисно для виділення об'єктів із зображення, розпізнавання контурів або створення масок для подальшої обробки;

8) калібрування камери: OpenCV містить інструменти для калібрування камери, що дозволяє визначати параметри камери, такі як фокусна відстань, спотворення лінзи та геометричні перетворення. Це необхідно для точної обробки зображень, наприклад, у геометричному відображенні або розрахунку відстані;

9) машинне навчання та глибоке навчання: OpenCV має підтримку для інтеграції з бібліотеками машинного навчання та глибокого навчання, такими як TensorFlow і PyTorch. Це дозволяє використовувати потужні моделі для завдань класифікації, розпізнавання об'єктів, сегментації та інших, що базуються на зображеннях [13];

10) розпізнавання обличь та біометрія: OpenCV має вбудовані алгоритми для розпізнавання обличь, включаючи методи навчання на основі прикладів. Це використовується у системах спостереження банків або багатих компаній з великою кількістю працівників.

1.4 Правила та вимоги для систем моніторингу приміщення

Правила налагодження й установки систем моніторингу приміщень включають багатоетапний план підготовки. Перш за все треба визначити мету системи моніторингу. Необхідно конкретизувати місце за яким ми плануємо слідкувати і які результати хочемо отримати. Наступним кроком є оцінка потреб щодо обладнання моніторингу, такі як камери, датчики, системи збору даних тощо. Обрання надійних, сумісних з нашою системою компонентів є

важливою задачею. Далі треба визначити оптимальні місця для розташування камер, датчиків та іншого обладнання, враховуючи сліпі зони.

Наступним етапом є розміщення усієї системи на території необхідної нам зони. Треба підключити обладнання до електромережі та мережі передачі даних. Необхідно переконайтеся, що всі з'єднання правильно налаштовані та працюють стабільно. Наступний крок – налаштування програмного забезпечення. Треба встановити необхідне програмне забезпечення для керування та моніторингу системи. Далі наступає етап перевірки роботи системи моніторингу, включаючи камери, датчики та тестування передачі даних. Потім налаштовуємо параметри системи моніторингу, такі як розмір зображення, чутливість датчиків, порогові значення тощо [9].

Важливо забезпечити належну документацію процесу налагодження системи моніторингу. Треба забезпечити підтримку для користувачів, які використовують систему, і надати контактні дані для звернення в разі необхідної допомоги.

Також, після налагодження й установки системи моніторингу, рекомендується провести періодичну перевірку та обслуговування обладнання для забезпечення його надійної роботи.

Важливо також враховувати відповідні нормативні вимоги та законодавство, що стосується моніторингу приміщень, зокрема правила щодо конфіденційності даних, приватності і захисту персональних даних.

В Україні, нормативні вимоги та законодавство, що стосуються систем моніторингу приміщень, включають наступне:

1) закон України «Про захист персональних даних»: Цей закон регулює збір, зберігання, використання та захист персональних даних. Застосовується до систем моніторингу, які збирають персональні дані про фізичних осіб. Заборонено збирати та використовувати персональні дані без попередньої згоди власника цих даних;

2) конституція України: Конституція гарантує право на приватність і конфіденційність особи. Системи моніторингу повинні дотримуватись цих принципів та не порушувати особисті права громадян;

3) закон України «Про охорону праці»: Цей закон встановлює вимоги щодо безпеки праці та охорони здоров'я працівників. При встановленні систем моніторингу потрібно слідкувати за безпекою працівників, які знаходяться під спостереженням;

4) закон України «Про телекомунікації»: Цей закон встановлює правила щодо передачі і обробки інформації в телекомунікаційних мережах. При використанні систем моніторингу потрібно дотримуватись цих правил та забезпечити конфіденційність і безпеку передачі даних.

Важливо мати на увазі, що законодавство може підлягати змінам та оновленням, тому рекомендується періодично перевіряти актуальність нормативних вимог та консультуватись з юридичними фахівцями, щоб дотримуватись всіх вимог і забезпечити відповідність системи моніторингу вимогам законодавства.

Висновки до розділу 1

Дослідження показало, що використання IP-камер спільно з бібліотекою OpenCV дозволяє автоматизувати процес моніторингу приміщень, забезпечуючи більш високий рівень безпеки та зручності. IP-камери забезпечують високу якість зображення, великий радіус та можливість віддаленого доступу до відеопотоку. Бібліотека OpenCV, зокрема, надає широкі можливості для обробки відео, виявлення руху, розпізнавання обличь та інші аналітичні функції.

Застосування IP-камер та OpenCV підвищує ефективність моніторингу приміщень, допомагаючи виявляти незвичайні події та реагувати на них вчасно. Розпізнавання обличь дозволяє ідентифікувати особи та контролювати доступ до обмежених зон.

Крім безпеки, автоматизація моніторингу приміщень також сприяє ефективному управлінню ресурсами. Це може мати значний вплив на економію енергії, забезпечення комфорту та оптимізацію робочих процес.

Аналіз результатів показав, що автоматизація моніторингу приміщень з використанням IP-камер та OpenCV є перспективним рішенням з багатьма перевагами.

По-перше, ця технологія дозволяє знизити людський фактор та помилки, оскільки система здатна працювати безперервно, алгоритми аналізу відео точніше і швидше виявляють незвичайні події, а розпізнавання обличчя дозволяє ідентифікувати осіб з високою точністю. Це допомагає покращити рівень безпеки та вчасно реагувати на потенційні загрози.

По-друге, автоматизація моніторингу забезпечує зручність управління приміщеннями. Відеопотік може бути доступний в режимі реального часу через Інтернет, що дозволяє оперативно контролювати ситуацію навіть на віддаленій відстані. Додатково, інтеграція з іншими системами безпеки, такими як системи контролю доступу або пожежної сигналізації, дозволяє створювати комплексні рішення для забезпечення безпеки приміщень.

2 РОЗРОБКА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Види IP-камер спостереження

Міні-камери

Це внутрішні відеокамери в мініатюрному плоскому квадратному або прямокутному корпусі, для настільного або настінного монтажу. Міні-камери через їх специфічну форму часто називають кубовими камерами, хоча останнім часом через різноманітність форм визначальним фактором при віднесенні камери до цієї секції є її мініатюрні розміри і наявність зовнішніх роз'ємів, відкритих для підключення живлення і мережі [15].

Рисунок 2.1 – Міні-камера D-Link DCS-931L

В основному такі камери використовуються в малобюджетному «домашньому» відеоспостереженні.

До переваг міні-камер можна віднести невеликі габарити, і досить широкий функціонал в невеликому корпусі, це Wi-Fi, вбудований мікрофон, вбудований динамік, ІЧ підсвічування, PoE та інші.

Практично всі мініатюрні камери дозволяють підключатися до камери віддалено, через інтернет, без використання статичної IP-адреси або таких функцій, як DDNS і UPnP.

Ця функція була розроблена, щоб полегшити неспеціалізованим користувачам налаштування віддаленого доступу до камери. Вона дозволяє переглядати відео з камери онлайн, переглядати відеоархів з SD-карти, управляти поворотним пристроєм і використовувати інші функції.

Плюси:

– низька ціна;

– електроживлення в комплекті.

Мінуси:

– наявність поворотного шарніра робить камеру більш вразливою для диверсійних дій, зловмисник може повернути камеру в сторону на час скоєння злочину;

– використання зовнішнього відкритого роз'єму живлення камери дозволяє зловмиснику дуже легко скинути живлення від камери на час злочину;

– зовнішній доступ до карти пам'яті SD може призвести до втрати відеоархіву.

Боді-камери

Камери з фіксованою коробкою є традиційним типом камер для систем відеоспостереження. Такі камери зазвичай розміщуються на видному місці, також добре видно напрямок їх огляду. Ось чому вони ідеально підходять для запобігання злочинності.

Рисунок 2.2 – Боді-камера D-Link DCS-8627LH

Боді-камери дозволяють вирішувати будь-які, навіть конкретні і нестандартні завдання, підбираючи необхідний об'єктив, корпус і аксесуари. Корпусні відеокамери мають форму паралелепіпеда, вони стабільно присутні в лінійці всіх провідних виробників.

Корпус забезпечується, як правило, змінною лінзою і без неї. Це дозволяє окремо вибрати камеру і окремо підібрати об'єктив саме під параметри вашої системи. Крім того, боді-камери для внутрішнього застосування дозволяють підібрати індивідуальний захисний чохол, наприклад, зі склоочисником, що практично на невизначений час розширює сферу використання боді-камер.

Плюси:

– Важливою перевагою боді-камер є можливість індивідуального підбору об'єктива, але заощадивши на якому можна втратити всі переваги пристрою;

– Камери великі, їх важко не помітити. Злочинці, місця, обладнані системами відеоспостереження, вважають більш ризикованими, і вибирають місця, де відсутні системи відеоспостереження. Відповідно, чим більше камера, тим більша ймовірність, що злочинець її помітить.

Мінуси:

– Як правило, ціна вища;

– Наявність поворотного шарніра робить камеру більш вразливою для диверсійних дій, зловмисник може повернути камеру в сторону на час скоєння злочину. Крім того, що сам інцидент не увійде в відео, стежити за цим теж складно, але для цього можна мати сервісну аналітику.

Циліндричні камери

Циліндричні камери – нерухомі камери в циліндричному корпусі більш компактні і легкі в порівнянні зі звичайними нерухомими боді-камерами. Вони готові до зовнішнього використання, а значить, можуть встановлюватися як всередині приміщення, так і зовні без додаткового захисного кожуха.

Рисунок 2.3 – Циліндрична камера D-link DCS-7413

Всі стаціонарні циліндричні камери оснащені вбудованим ІЧ-підсвічуванням, яке дозволяє вести спостереження при слабкому освітленні або в повній темряві. Як і звичайні боді-камери, циліндричні камери дуже легко орієнтуються в просторі. Однак стаціонарні циліндричні камери не підтримують зміну об'єктива.

Циліндричні камери мають пару переваг перед фіксованими натільними камерами. По-перше, циліндрична форма корпусу при однаковій товщині стінок має значно більшу механічну міцність в порівнянні з кузовом у вигляді паралелепіпеда.

По-друге, боді-камера у вигляді паралелепіпеда більше сприятиме накопиченню снігу на тілі, ніж циліндрична, але і це ще не все, більше снігу означає більше води, яка з'являється при виході сонця, ця вода буде текти виключно в сторону об'єктива, а вночі знову буде морозно, а це вже загрожує появою бурульок перед об'єктивом.

У випадку з циліндричним корпусом ця небезпека практично зводиться до нуля.

Також варто відзначити, що перевагою циліндричних камер стане козирок, що захищає від сонця з можливістю регулювання. Якщо камера оснащена фіксованим козирком, ефект захисту від сонця буде мінімальним.

Плюси:

– Циліндрична форма корпусу більше підходить для зовнішньої експлуатації, особливо в умовах російської зими;

– Нерозбірна конструкція більшості циліндричних камер, ceteris paribus, забезпечить кращий пило- і вологозахист.

Мінуси:

– Неможливість змінити лінзу дещо звужує спектр застосування.

Купольні камери

Купольна камера – це нерухома камера, яка поставляється в компактному куполоподібному корпусі. Його особливість - стриманий, що не привертає увагу погляд. У більшості випадків це буде оптимальним рішенням для використання всередині приміщень.

Рисунок 2.4 – Купольна камера D-Link DCS-4603/UPA

Цей тип камер не дозволяє зловмисникам дізнатися, куди спрямований об'єктив камери, а значить, зловмисник не розуміє, які можуть бути «мертві зони» камери, що може позитивно позначитися на безпеці об'єкта.

Важливою перевагою купольних камер є їх висока стійкість до диверсійних дій, тобто камера не може просто відвернутися в іншу сторону, як у випадку з боді-камерами. Зламати його, звичайно, можна, але це вже тягне за собою серйозні витрати, так як камера коштує досить дорого. Крім того,

злом камери не залишиться непоміченим і спричинить за собою розслідування.

Купольні камери можуть оснащуватися об'єктивом із фіксованою або змінною фокусною відстанню. У деяких моделях об'єктив може бути взаємозамінним. Є можливість кріплення на вертикальній площині спеціальним кронштейном. Внутрішній механізм регулювання дозволяє повертати об'єктив камери в будь-який з осей.

Найчастіше купольні камери встановлюються в приміщенні, хоча випускаються купольні камери, кліматичне оформлення яких дозволяє встановлювати їх на відкритому повітрі, але робиться це досить рідко. Причина в тому, що практично всі купольні камери не мають козирка, що захищає від сонця. Але навіть в цьому правилі є винятки.

Оскільки купольні камери часто використовуються в місцях, де пристрій легко доступний, наприклад, ліфті, «антивандальні» купольні камери користуються великим попитом. Ну і, як і у всіх IP-камер, живлення купольної камери за допомогою PoE, незважаючи на трохи вищу ціну.

Плюси:

– Завдяки мінімалістичному стриманому дизайну вони підходять для всіх типів приміщень;

– Більш висока вандальна стійкість в порівнянні з фіксованим опором кузова через відсутність брекета.

Мінуси:

– Для регулювання об'єктива необхідно розібрати фотоапарат (зняти купол), це може негативно позначитися на пило- і вологозахисті, і ускладнити монтаж;

– У більшості випадків вони не підходять для зовнішнього використання, оскільки немає захисту від прямих сонячних променів

Купольні камери EyeBall

Цей тип камер є різновидом купольної камери, хоча куполів в її класичному розумінні не має. Основною відмінністю є відсутність класичного прозорого купола, що значно спрощує регулювання і регулювання кута огляду і фокусної відстані.

Рисунок 2.5 – Купольна камера EyeBall D-Link DCS-4802E/UPA

Крім того, цей форм-фактор менше за розмірами, ніж стандартні купольні камери, а через відсутність прозорого купола він ще менш помітний на білому тлі.

Плюси:

– Для класичних купольних камер невеликим недоліком є геометричні спотворення зображення і відблиски, які характерні для будь-якого прозорого купола. Камери EyeBall не мають купола, а тому не мають відблисків спотворення зображення, нехай і мінімальних;

– Простіша та швидша установка та налаштування.

Мінуси:

– Відкриті елементи позиціонування і настройки можуть бути використані зловмисниками для саботажу.

Панорамні камери FishEye

Є ще один варіант псевдоPTZ IP камер, це панорамні відеокамери. Це не окремий форм-фактор, а відеокамера зі спеціальним панорамним об'єктивом «риб'яче око».

Рисунок 2.6 – Панорамна камера FishEye D-Link DCS-6010L

Такий об'єктив знімає сцену на 360 градусів, а процесор провайдера виправляє неминучі спотворення і формує підсумкове панорамне зображення, завдяки чому користувач переміщує готове зображення в кадрі, в той час як сама камера нікуди не повертається.

Кругове зображення може бути перетворено в прямокутні зображення різних форматів за допомогою програмного забезпечення камери. Можна отримати одинарну або подвійну панораму, а також чотири різних виду в одному кадрі (квадрорежим, що імітує роботу чотирьох окремих камер). Ви можете конвертувати безпосередньо в образ портативної системи або зображення, яке вже було записано.

2.2 Планування розташування системи спостереження приміщення

Для створення системи моніторингу необхідного нам приміщення потрібно перш за все створити схему плану для організації подальшої роботи

з обладнанням. Для створення основного плану приміщення можна обрати різні програми.

AutoCAD – професійна програма для комп'ютерного проєктування, яка надає широкі можливості для створення детальних планів і схем. Вона часто використовується архітекторами, інженерами та дизайнерами.

SketchUp – інтуїтивно зрозуміла програма, яка дозволяє швидко створювати 3D-моделі будинків та інших приміщень. Вона підходить для широкого кола користувачів, включаючи архітекторів, дизайнерів та навіть домашніх користувачів.

Floorplanner – це веб-сервіс, який дозволяє створювати плани приміщень у вигляді 2D-схеми. Він має простий інтерфейс та ряд корисних функцій, таких як додавання меблів та інтерактивну навігацію.

Sweet Home 3D – безкоштовна програма з відкритим кодом для створення планів та візуалізації приміщень. Вона має простий інтерфейс і дозволяє швидко розташовувати меблі, створювати стіни, додавати освітлення та інші деталі.

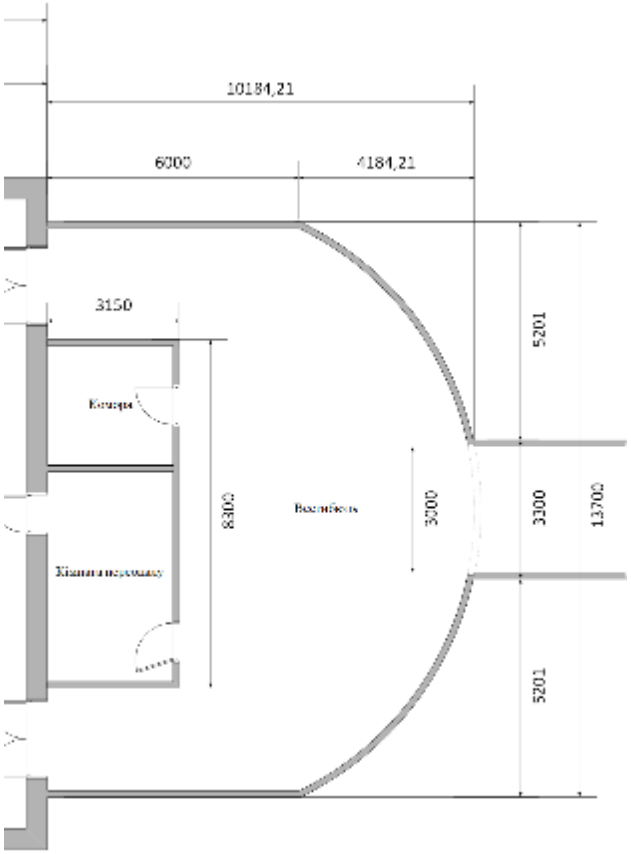
RoomSketcher – веб-сервіс, який дозволяє створювати плани приміщень у вигляді 2D-схеми або 3D-моделей. Він має багато функцій, включаючи можливість додавати кольори, текстури та меблі, а також переглядати результати в режимі віртуальної реальності.

Microsoft Visio – це програма для векторного графічного проєктування і створення схем, діаграм, планів та інших візуальних представлень. Вона розроблена компанією Microsoft і має широкий спектр застосувань у багатьох галузях, включаючи бізнес, технології, інженерію та організацію даних.

У даній роботі було використано Autodesk AutoCAD та Microsoft Visio.

В якості основного приміщення було використано власно змодельоване приміщення підводного холу під назвою «Норей». За допомогою ПЗ Microsoft Visio 2021 було створено основний план приміщення (рис. 2.7).

а)



б)

Рисунок 2.7 – Загальний план приміщення: а)хол, зовнішній коридор; б)вестибюль, комора, кімната персоналу

Таблиця 2.1 – Розміри приміщень

Назва приміщення	**Розміри**	**Висота**
Зовнішній коридор, мм	3000 × 17794	3500
Хол, мм	20000 × 15000	8000
Вестибюль, мм	10184 × 13700	3000
Кімната персоналу, мм	3000 × 5000	3000
Комора, мм	3000 × 2850	3000

У програмі Autodesk AutoCAD було реалізовано дизайн приміщення (рис. 2.8).

а)

б)

в)

Рисунок 2.8 – Рендери холу підводного готелю «Норей»

AutoCAD використовувався для створення 3D-моделі приміщень, що дозволило отримати реалістичне уявлення про загальний вигляд і функціональність. Дизайн основного приміщення використовувався для візуального розуміння об’ємів, освітлення та налаштування виглядів з камер. Враховувати необхідно було й освітлення під водою, щоб обрати необхідні лампи та елементи дизайну.

2.3 Моделювання системи відеоспостереження

Для початку моделювання треба обрати камери (табл. 2.2), що будуть задовольняти потреби щодо покриття територій. Було обрано 3 екземляри, що відповідають задачам нагляду за приміщенням різного призначення. Всі обрані камери моють однакові матриці CMOS та формат відео H.264.

Першою задачею була організація нагляду за невеликими приміщеннями по типу комори, коридорного проходу та приміщення стійки реєстратури з видом на відвідувача. Для цього гарно підходить камера D-Link DCS-7413, що має роздільну здатність в 1920×1080 до 15 кадрів/сек, кут огляду

в 77.4° (табл. 2.2). Головне, що повинна робити камера – це знімати обличчя людей які потапляють в її кут зору.

Таблиця 2.2 – Характеристики камери D-Link DCS-7413

Модель	D-Link DCS-7413
Зображення	
Способи підключення	Fast Ethernet
Фокусна відстань	4 мм
Максимальна роздільна здатність	1920×1080 до 15 кадрів/сек
Кути огляду	77.4°
Формати відео	H.264, MJPEG, MPEG-4
Матриця	CMOS

Другою задачею була організація нагляду за великим холом в 300м2. Для забезпечення наглюду за всією площею треба взати камери з більшим кутом огляду, таку як D-Link DCS-5615. Ця камера добре підходить для встановлення відеонагляду за великим ділянками.

Таблиця 2.3 – Характеристики камери D-Link DCS-5615

Модель	D-Link DCS-5615
Зображення	
Способи підключення	Fast Ethernet
Фокусна відстань	4.3 мм

Максимальна роздільна здатність	1920×1080 до 15 кадрів/сек
Кути огляду	88°
Формати відео	H.264, MJPEG, MPEG-4
Матриця	CMOS

Третім завданням була організація нагляду за прохідним приміщенням, де вже не є першочерговою задачею детальне зображення обличчя. Головне було налаштувати відеонагляд.. Для цього підішла камера типу риб’яче око D-Link DCS-6010L. Вона буде контролювати велику велику площу і забезпечувати нагляд за безпекою.

Таблиця 2.4 – Характеристики камери D-Link DCS-6010L

Модель	D-Link DCS-6010L
Зображення	
Способи підключення	Fast Ethernet
Фокусна відстань	1.25 мм
Максимальна роздільна здатність	1920×1080 до 15 кадрів/сек
Кути огляду	360°
Формати відео	H.264, MJPEG, MPEG-4
Матриця	CMOS

Моделювання системи відеоспостереження (рис. 2.9) виконується в IP Video System Design Tool. Це програмне забезпечення, призначене для допомоги в проєктуванні та розробці комплексних відеосистем. Вона надає інженерам і дизайнерам можливість створювати, моделювати й аналізувати відеосистеми перед їхньою фізичною реалізацією.

Основною метою Video System Design Tool є полегшення та прискорення процесу проектування відеосистем, даючи змогу користувачам візуалізувати різні компоненти системи та оцінити їхню взаємодію. Програма пропонує інтуїтивно зрозумілий інтерфейс, де користувачі можуть обирати і розміщувати різні елементи відеосистеми, такі як камери, монітори, відеореєстратори, комутаційне обладнання та інші компоненти [17].

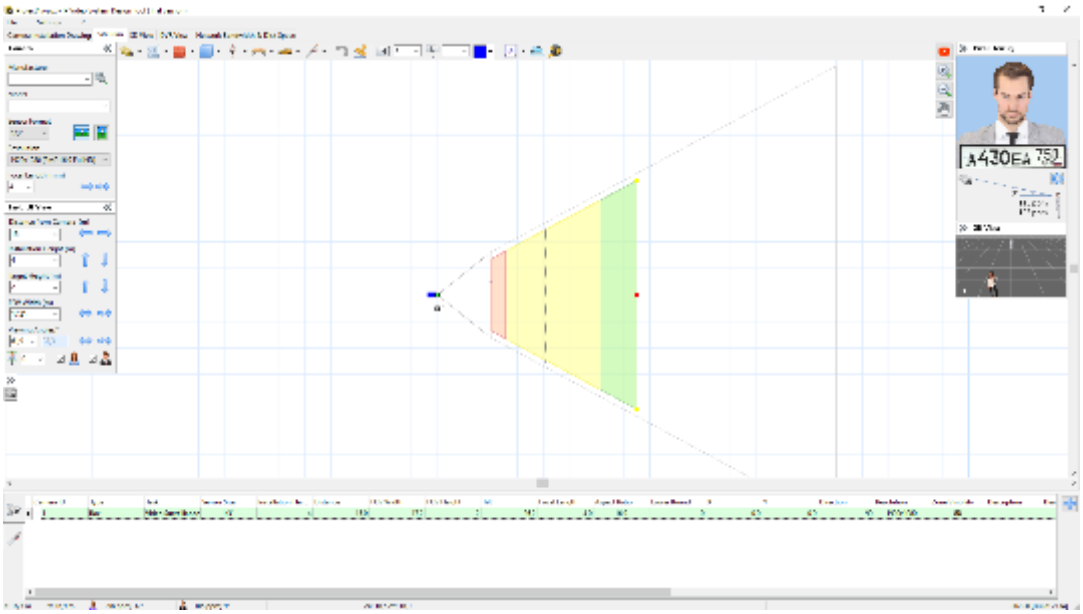
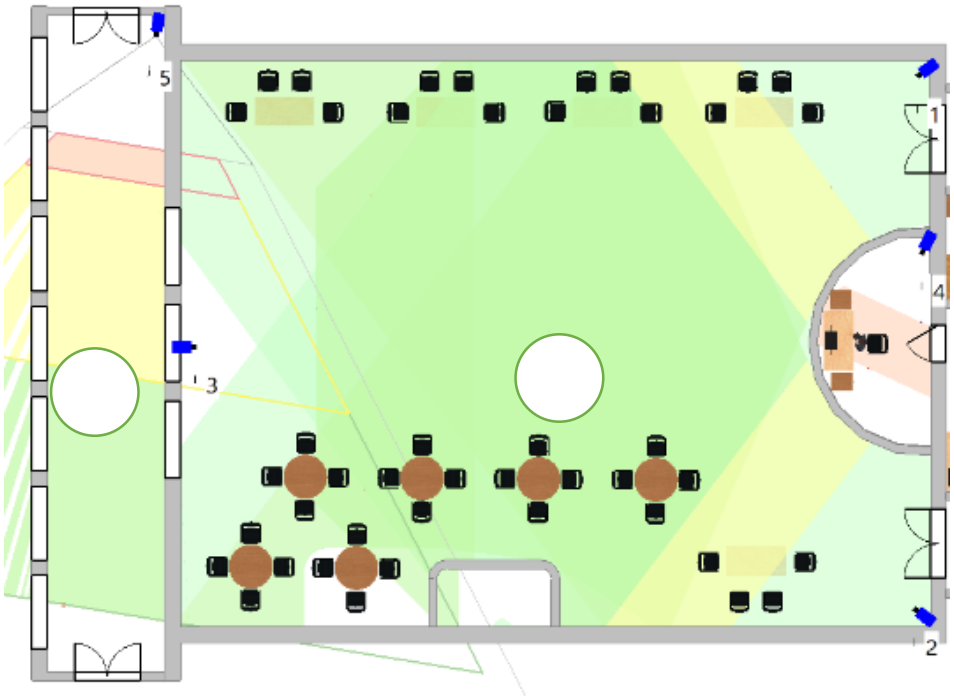
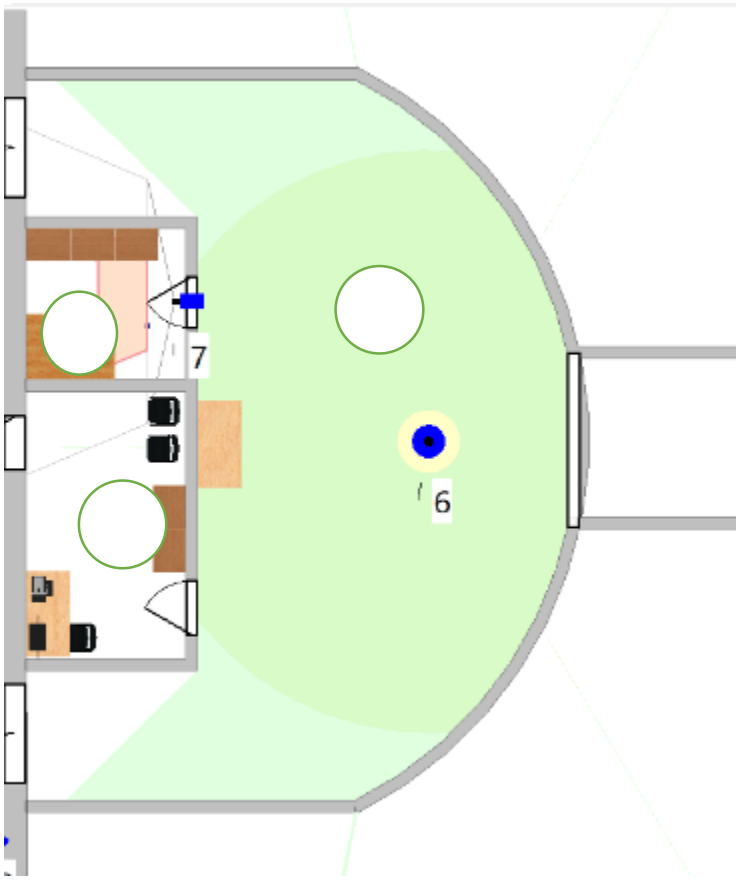


Рисунок 2.9 – Моделювання системи відеоспостереження

Оскільки камери монтуються на стелю та на стіни на висоті 3 - 4 метри від підлоги, отримаємо малі мертві зони. Головне покрити максимальну видимість робочої зони (рис. 2.10).



а)



б)

Рисунок 2.10 – Приклад перегляду встановлених камер

а)

б) в)

Рисунок 2.11 – Вигляд з камери D-Link DCS-7413

а)

б)

в)

Рисунок 2.12 – Вигляд з камери D-Link DCS-5615

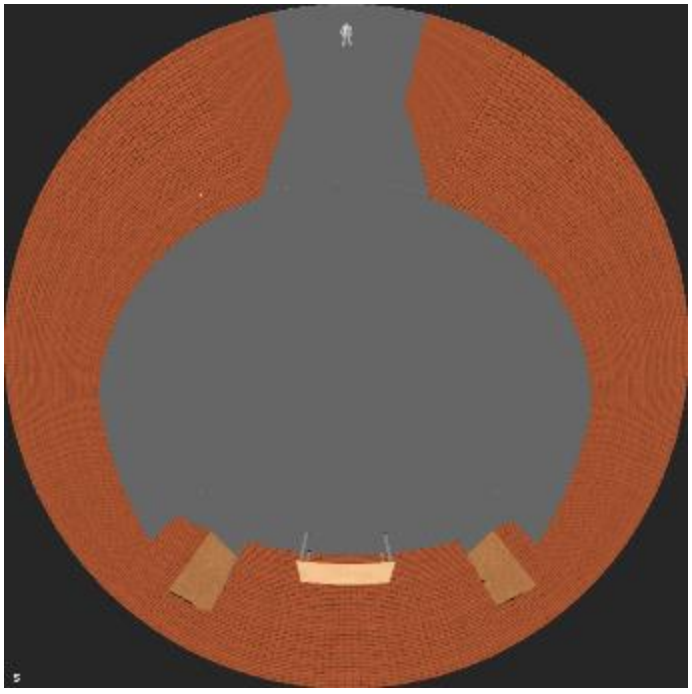
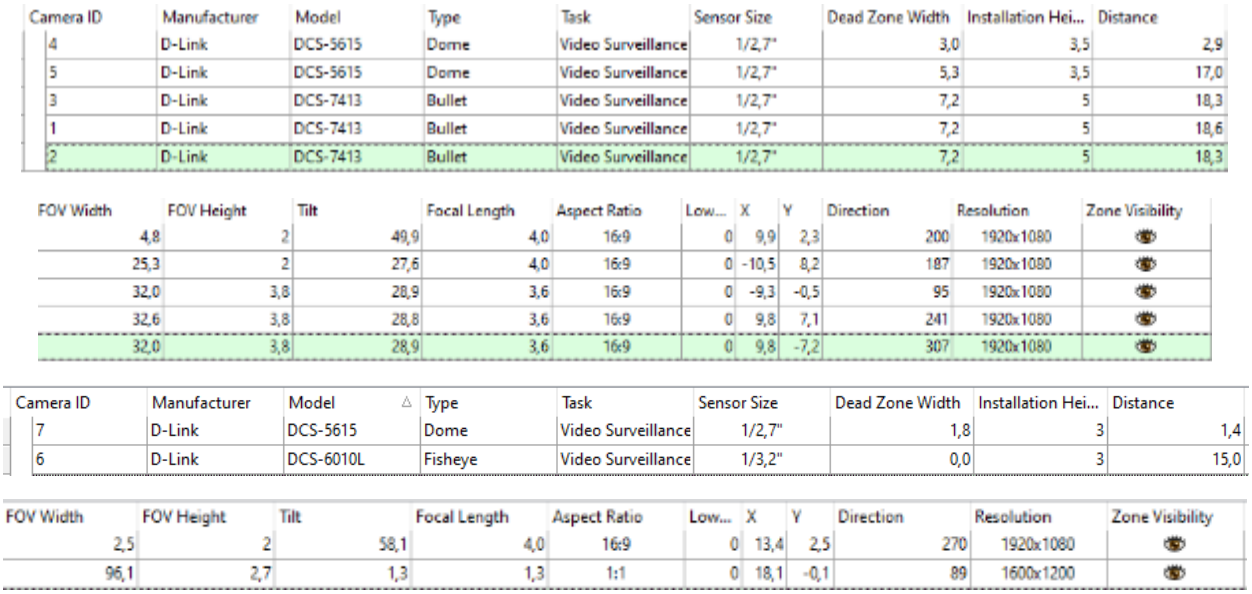


Рисунок 2.13 – Вигляд з камери D-Link DCS-6010L

Монтажну схему камер можна передивитись у середовищі IP Video System Design Tool (рис. 2.14).



Camera ID	Manufacturer	Model	Type	Task	Sensor Size	Dead Zone Width	Installation Hei...	Distance
4	D-Link	DCS-5615	Dome	Video Surveillance	1/2,7"	3,0	3,5	2,9
5	D-Link	DCS-5615	Dome	Video Surveillance	1/2,7"	5,3	3,5	17,0
3	D-Link	DCS-7413	Bullet	Video Surveillance	1/2,7"	7,2	5	18,3
1	D-Link	DCS-7413	Bullet	Video Surveillance	1/2,7"	7,2	5	18,6
2	D-Link	DCS-7413	Bullet	Video Surveillance	1/2,7"	7,2	5	18,3

FOV Width	FOV Height	Tilt	Focal Length	Aspect Ratio	Low...	X	Y	Direction	Resolution	Zone Visibility
4,8	2	49,9	4,0	16:9	0	9,9	2,3	200	1920x1080	
25,3	2	27,6	4,0	16:9	0	-10,5	8,2	187	1920x1080	
32,0	3,8	28,9	3,6	16:9	0	-9,3	-0,5	95	1920x1080	
32,6	3,8	28,8	3,6	16:9	0	9,8	7,1	241	1920x1080	
32,0	3,8	28,9	3,6	16:9	0	9,8	-7,2	307	1920x1080	

Camera ID	Manufacturer	Model	Type	Task	Sensor Size	Dead Zone Width	Installation Hei...	Distance
7	D-Link	DCS-5615	Dome	Video Surveillance	1/2,7"	1,8	3	1,4
6	D-Link	DCS-6010L	Fisheye	Video Surveillance	1/3,2"	0,0	3	15,0

FOV Width	FOV Height	Tilt	Focal Length	Aspect Ratio	Low...	X	Y	Direction	Resolution	Zone Visibility
2,5	2	58,1	4,0	16:9	0	13,4	2,5	270	1920x1080	
96,1	2,7	1,3	1,3	1:1	0	18,1	-0,1	89	1600x1200	

Рисунок 2.14 – Монтажна схема

При встановленні масивної системи моніторингу з кількома камерами необхідно забезпечити їх підключення до однієї точки, тобто до мультиплексора зо допомогою витої пари та технології Fast Ethernet.

Fast Ethernet є стандартом мережі, який розширює можливості оригінального Ethernet, забезпечуючи швидкісну передачу даних на швидкості до 100 мегабіт на секунду (Mbps). Це означає, що Fast Ethernet має швидкість передачі даних в 10 разів швидше, ніж стандартний Ethernet, який працює на швидкості 10 Mbps.

Мультиплексор – це електронний пристрій або система, яка використовується для комбінування декількох вхідних сигналів в один вихідний канал. Він виконує функцію передачі даних з декількох джерел через один канал з метою забезпечення економії пропускної здатності та ресурсів.

Мультиплексор має набір вхідних портів, до яких підключаються окремі вхідні сигнали, і один вихідний порт, з якого отримується об'єднаний сигнал. Контролер (вхідний селектор) вибирає, який вхідний сигнал повинен бути переданий на вихідний порт. Залежно від режиму роботи мультиплексора, може бути різна кількість вхідних сигналів та різні способи управління вибором.

Основна функція мультиплексора полягає в тому, щоб передати вибраний вхідний сигнал на вихідний канал, використовуючи відповідні комбінації керуючих сигналів. Це дозволяє об'єднувати дані з різних джерел (наприклад, камер відеоспостереження) і передавати їх через обмежену кількість каналів або обладнання, що забезпечує економію ресурсів та спрощує процес обробки сигналу.

В якості мультиплексора будемо використовувати сервер з встановленим програмним забезпеченням для відеоспостереження, що дозволяє налаштувати сервер таким чином, щоб він виконував функцію мультиплексора.

Використання сервера на базі Linux в якості мультиплексора має свої переваги, такі як гнучкість управління та можливість розширення функціональності через різноманітні програми та налаштування.

Така система дротовим підключенням з мультиплексором забезпечує захищене з’єднання, що є дуже важливим для систем, які повинні слідкуваиза безпекою в приміщенні.

2.4 Розрахунки системи моніторингу

Модель D-Link DCS-5615 та D-Link DCS-7413, роздільна здатність 1920×1080. Модель D-Link DCS-6010L, роздільна здатність 1600×1200.

$$T = H \times V, \qquad (2.1)$$

де H – роздільність по вертикалі, пікселів;

V – роздільність по горизонталі, пікселів.

В роботі присутні камери з роздільною здатністю 1920×1080 та 1600×1200, тому:

$$T1 = 1920 \times 1080 = 2073600 \text{ (пікіселів)} \quad (2.2)$$

$$T2 = 1600 \times 1200 = 1920000 \text{ (пікіселів)} \quad (2.3)$$

$$P_N = T \times gl, \quad (2.4)$$

де gl – глибина кольору, біт.

Для CMOS матриці глибина кольору – 24.

$$P1_N = 2073600 \times 24 = 6075 \text{ (Кбайт)} \quad (2.5)$$

$$P2_N = 1920000 \times 24 = 5625 \text{ (Кбайт)} \quad (2.6)$$

Обчислимо розмір стиснутого кадру.

$$P_Z = \frac{P_N}{k}, \quad (2.7)$$

де k – ступінь стиснення кодеку.

Для усіх камер використовується кодек H.264, оскільки у результаті він дає якісне зображення та значно зменшує розмір вихідного файлу. Згідно з інформацією з навчального посібника, ступінь стиснення даного алгоритму – 1:270.

$$P1_Z = \frac{6075}{270} = 23 \text{ (Кбайт)} \quad (2.8)$$

$$P2_Z = \frac{5625}{270} = 20{,}8 \text{ (Кбайт)} \quad (2.9)$$

Мережевий трафік для внутрішніх камер D-Link DCS-5615 та D-Link DCS-7413:

$$S_T = \frac{23 \times 1024 \times 8 \times 15 \times 3}{1000000} = 8{,}5 \text{ (Мбіт/с)} \quad (2.10)$$

Мережевий трафік для внутрішньої камери D-Link DCS-6010L:

$$S_T = \frac{20{,}8 \times 1024 \times 8 \times 15 \times 1}{1000000} = 2{,}6 \text{ (Мбіт/с)} \quad (2.12)$$

Перевірка мережевого трафіку стосовно зжатого кадру (рис. 2.15) розраховано в програмі IP Video System Design Tool.

Resolution	Compression	Frame Size*, KB	FPS	Days	Cameras	Recording %	Bandwidth, Mbit/s	Disk Space, GB	Bitrate, kbit/s
1920x1080 (2MP 16:9 FullHD)	?	23,0	15	21	6	100	17,0	3 845,9	2826
1600x1200 (2MP 4:3)	?	20,0	15	21	1	100	2,5	557,4	2457

Рисунок 2.15 – Розрахунок трафіку

Розрахуємо обсяг відеоархіву, необхідний для збереження запису з камер за тиждень у рівнянні.

Обсяг відеоархіву буде розраховано за наступною формулою:

$$V_{HHD} = \frac{(S_{DCS-5615}+S_{DCS-7413}+S_{DCS-6010L})\times day \times hour \times minutes \times sec}{1000000 \text{ x } 8} \quad (2.13)$$

$$V_{HHD} = \frac{(8{,}5+8{,}5+2{,}6)\times 21\times 24\times 60\times 60}{1000000 \text{ x } 8} = 4{,}5(\textit{Тбайт}) \quad (2.14)$$

Перевіримо обсяг за допомогою встановлених утиліт програми IP Video System Design Tool (рис. 2.16).

Resolution	Compression	Frame Size*, KB	FPS	Days	Cameras	Recording %	Bandwidth, Mbit/s	Disk Space, GB	Bitrate, kbit/s
1920x1080 (2MP 16:9 FullHD)	?	23,0	15	21	6	100	17,0	3 845,9	2826
1600x1200 (2MP 4:3)	?	20,0	15	21	1	100	2,5	557,4	2457

Рисунок 2.16 – Розрахунок обсягу мережевого сховища

2.5 Підключення апаратного забезпечення через IP-адресу

Для представлення принципу підключення апаратного забезпечення використалась Wi-Fi міні камера Camsoy T30 F2-IP (рис. 2.17).

Рисунок 2.17 – Wi-Fi міні камера Camsoy T30 F2-IP

Таблиця 2.5 – Характеристики камери

Тип	Бездротова
Інфрачервона підсвітка	до 8 м
Роздільна здатність відео	1920x1080 1280x720 640x480
Виробник і тип матриці	CMOS
Розмір матриці	1/4
Формат відео	H.264
Кут огляду по горизонталі	140°
Частота запису	20-25 кадрів/сек
Слот для карт пам'яті	MicroSD
Максимальний обсяг карти пам'яті	64
Робоча напруга	DC-5V 400MA/h
Діапазон робочих температур	Від -10° до 60 °C
Розміри	52 * 25 * 15 мм
Вага	25 г

Для першого підключення камери необхідно завантажити застосунок HDMiniCam на свій смартфон. Далі необхідно увімкнути камеру та зачекати поки вона з’явиться у списку вибору WiFi, після чого до неї треба під’єднатися (рис. 2.18).

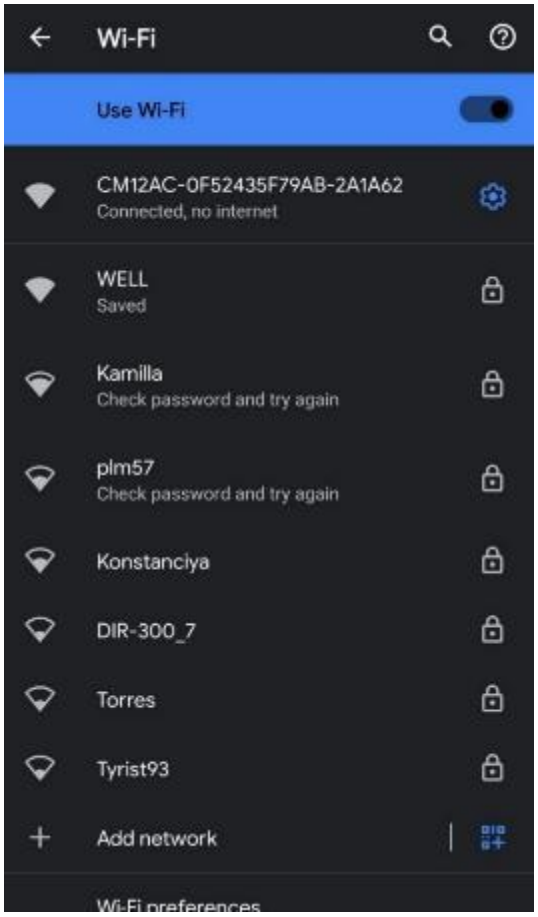


Рисунок 2.18 – камера у списку вибору WiFi

Тепер треба зайти в застосунок де камера автоматично повинна додатися (рис.2.19).

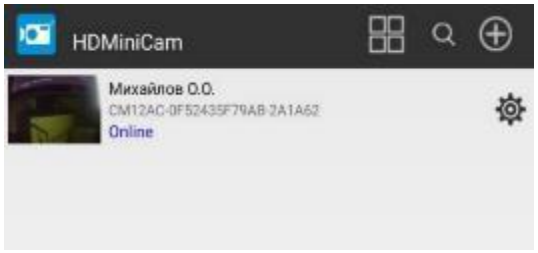
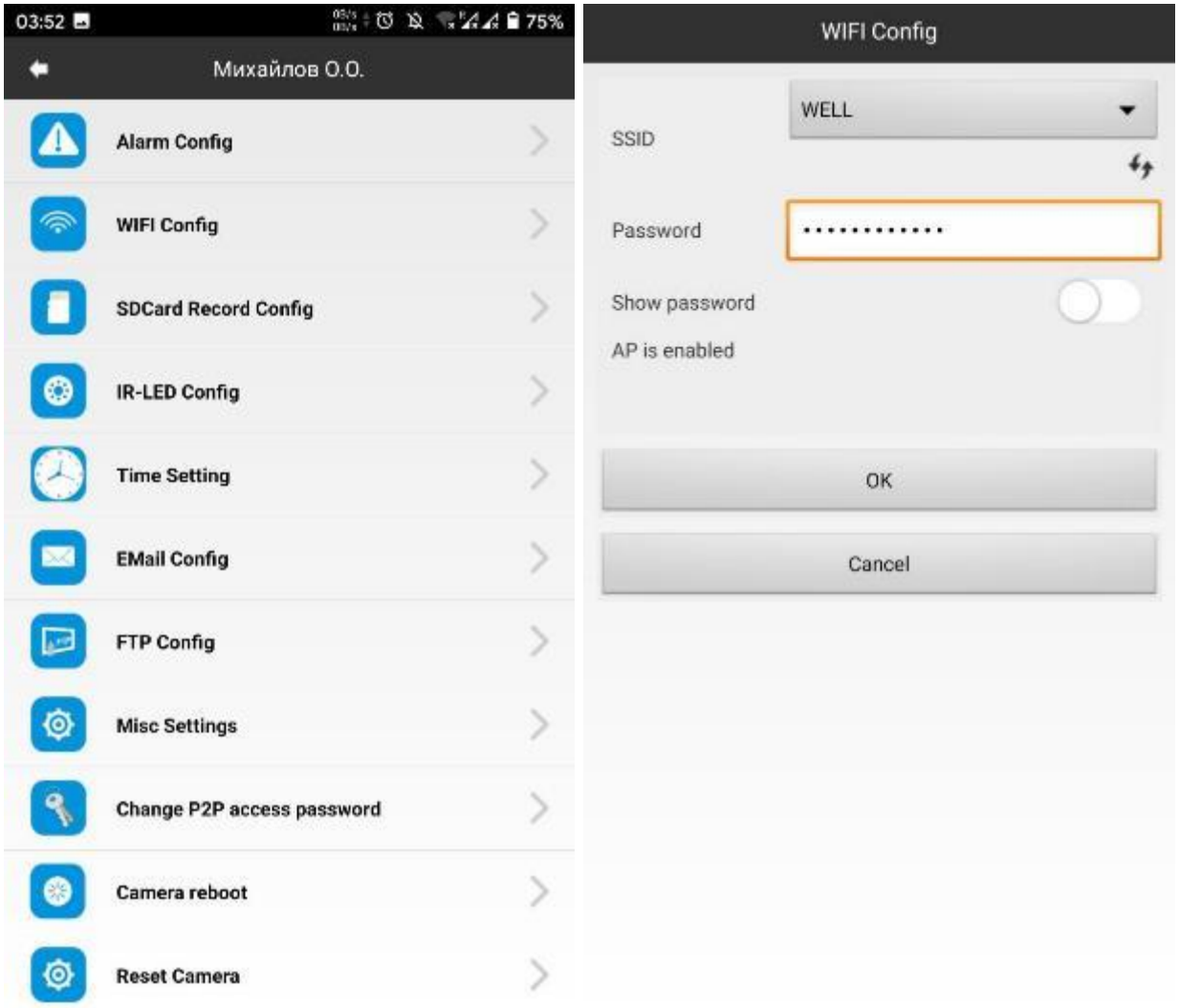


Рисунок 2.19 – камера у застосунку HDMiniCam

Заходимо та перевіряємо працездатність, та вивід зображення. Щоб до камери можна було б підключатися через локальну мережу, а в подальшому через інтернет, необхідно зайти в налаштування та з'єднати камеру з своїм WiFi (рис.2.20).



а) б)

Рисунок 2.20 – скріншоти налаштування WiFi з'єднання:
а) список налаштувань камері, б) налаштування WiFi

Тепер можна підключатися до камери через IP адресу, знайти її можна зайшовши до каталогу користувачів свого роутеру (рис.2.21).

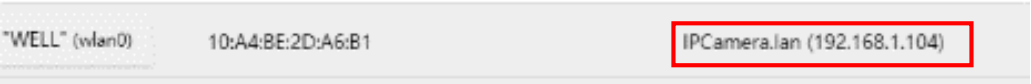
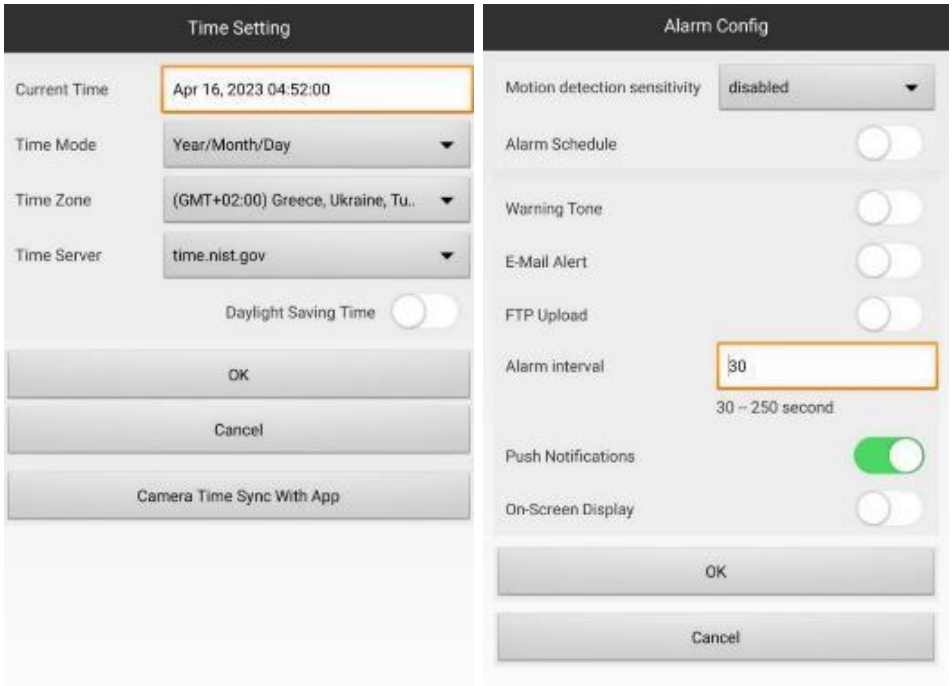


Рисунок 2.21 – камера у списку у списку клієнтів роутера

У моєму випадку випадку IP камери – 192.168.1.104. Цю адресу можна вставити до адресного рядка та перейти безпосередньо до інтерфейсу камери (рис.2.22).

Рисунок 2.22 – вигляд з камери на веб-сторінці

Налагодження режиму запису за графіком та/або при наявності руху в полі зору камери (рис.2.23).



а) б)

Рисунок 2.23 – скріншоти налаштування запису при наявності руху в полі

зору камери: а) налаштування часу та часового поясу, б) налаштування сигналізації

Додатково в браузерній версії можна налаштувати запис відео на комп’ютер (рис.2.24 та 2.25).

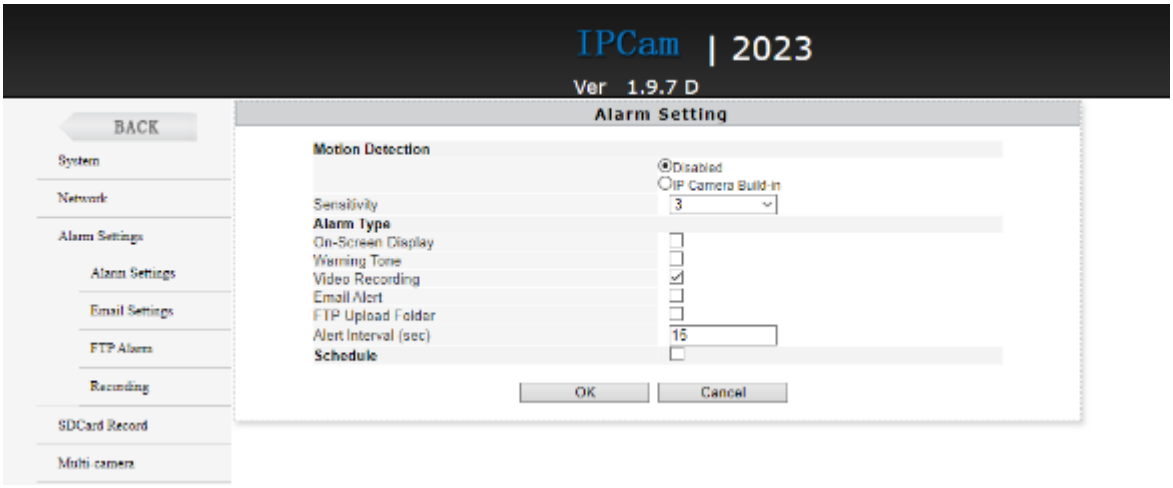


Рисунок 2.24 – налаштування запису відео

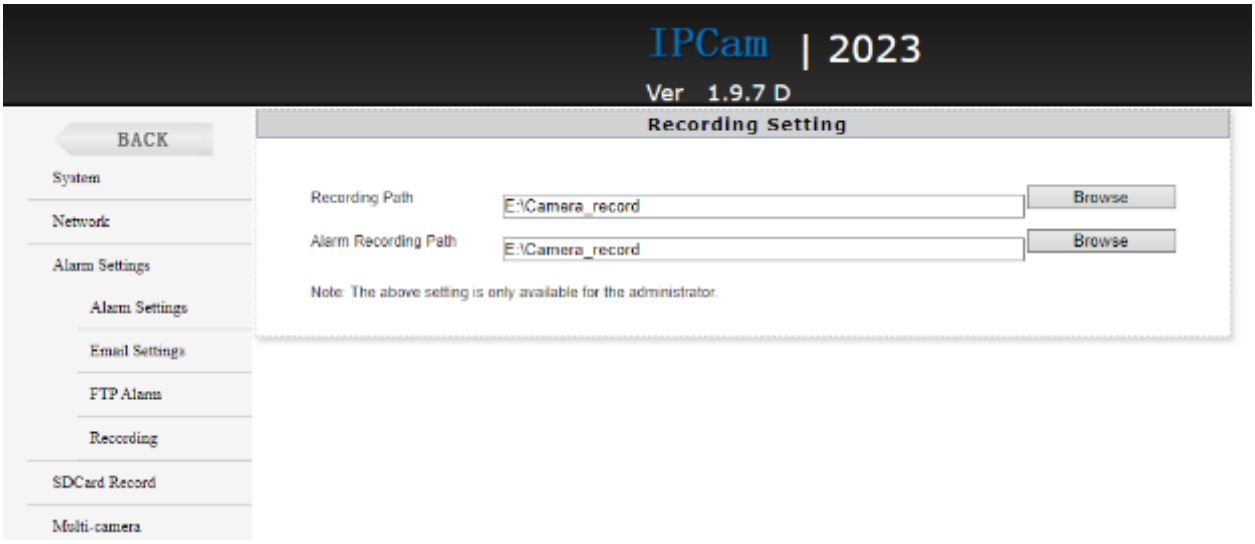


Рисунок 2.25 – налаштування шляху для зберігання запису відео

Щоб камера була доступна в Інтернеті, треба зробити проброс портів.

На прикладі мого роутера, для пробросу портів необхідно зайти в налаштування роутера, Мережа – Міжмережевий екран – Port Forwards. Далі треба створити новий запис.

У вікні створення даємо власну назву, вказуємо протокол TCP+UPD, вихідну зону залишаємо wan, на зовнішній порт назначаємо 5080 (можна будь-

який), зону призначення залишаємо lan, Обираємо IP-адресу нашої камери, у внутрішній порт треба ввести порт камери, в моєму випадку це 80 (рис.2.26).

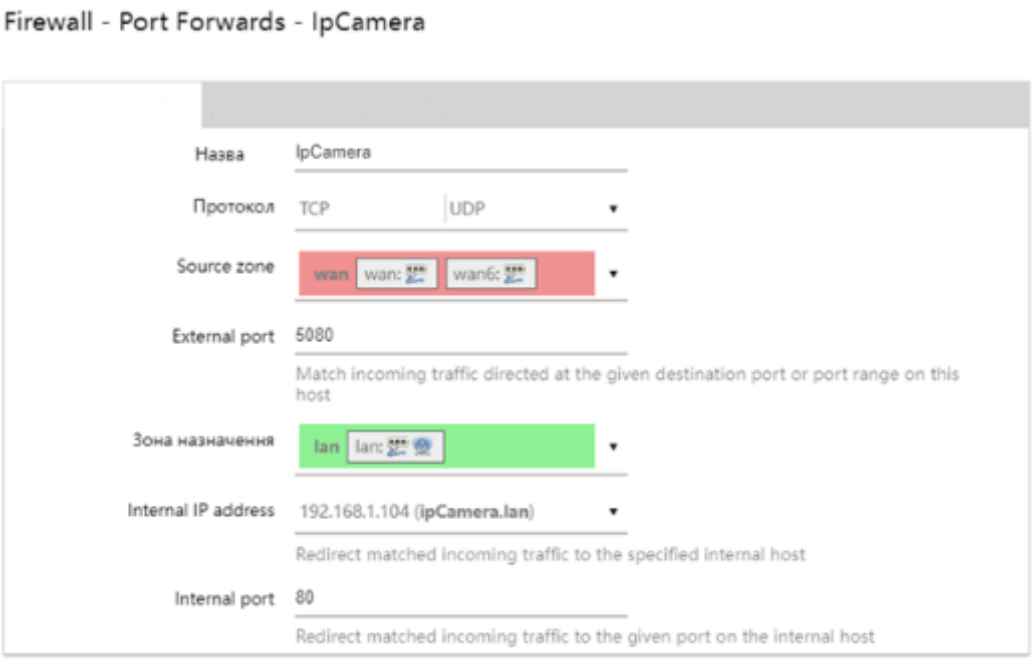


Рисунок 2.26 – налаштування порту для камери

Зберігаємо налаштування та застосовуємо всі зміни що ми зробили на роутері. Тепер до камери можна зайти в интернеті. Щоб це зробити спочатку треба дізнатися свій зовнішній IP, зробити це можливо за допомогою сервісу https://2ip.ua/. Зовнішня адреса повинна бути статичною, якщо ні то треба звернутися до провайдера.

Рисунок 2.27 – зовнішня IP-адреса на https://2ip.ua/

Отже, щоб тепер отримати доступ до камери необхідно в пошукову строку ввести http://141.101.**.67:5080

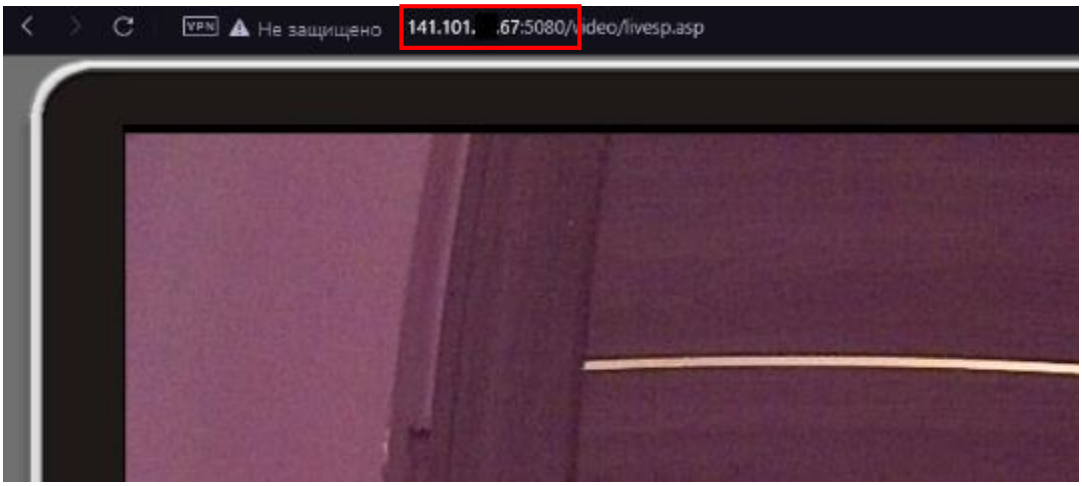


Рисунок 2.28 – доступ до віддаленої камери

Рисунок 2.29 – доступ до камери з комп’ютера

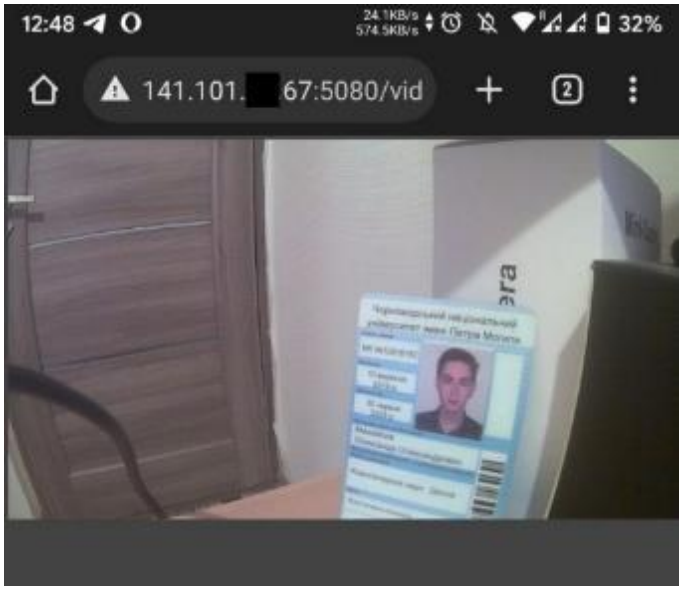


Рисунок 2.30 – доступ до камери з телефона

Висновки до розділу 2

Була розроблена та оформлена відповідним чином вся документація, зазначена у технічному завданні. При побудуванні системи першочерговою задачею є проєктування майбутньої мережі, тому що завдяки правильно визначеній топології мережі можна значно збільшити швидкість та функціональність системи, зменшити витрати на її створення та обслуговування.

При створенні розділу були використані такі програмні засоби:

1) Microsoft Word 2021 – текстовий процесор для роботи з електронними документами; оформлення звітів та відповідної документації;

2) Microsoft Visio 2021 – середовище для роботи з діаграмами та схемами; розробка структурних схем поверхів будинку, а також схем розміщення обладнання, інженерних рішень щодо протипожежної безпеки;

3) IP Video System Design Tool – середовище для проєктування мережі відеоспостереження;

4) Autodesk AutoCAD для – середовище для проєктування 2D та 3D планів приміщень й об’єктів та розробки дизайну.

Окрім програмних продуктів, було використано багато теоретичних, лекційних матеріалів, матеріалів з он-лайн бібліотек та навчальних посібників щодо характеристик телекомунікаційного обладнання, правил безпеки, стандартів Міжнародної електротехнічної комісії.

У майбутньому технології комп’ютерних мереж можуть сильно змінитися. Основним поштовхом до розвитку є поширення бездротових технологій та розробка штучного інтелекту. хоча зараз ці технологій не ідеальні та мають багато помилок та недоліків, але вони розвиваються і з кожним днем привносять щось нове.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Мова програмування Python

Python – це високорівнева, інтерпретована мова програмування, яка була розроблена з метою простоти, читабельності і зручності використання. Основні принципи, на яких базується Python, включають лаконічність, читабельність коду і експресивність.

Синтаксис Python дуже простий і легко читається. Замість використання фігурних дужок або ключових слів, як це робиться в багатьох інших мовах програмування, Python використовує відступи для визначення блоків коду. Це забезпечує чітку структуру і робить код більш зрозумілим.

Python – це інтерпретована мова програмування, що означає, що код виконується рядок за рядком без необхідності його компіляції. Це полегшує швидкість розробки, тестування і налагодження програм.

Є підтримка об'єктно-орієнтовану парадигму програмування. Ви можете визначати класи, створювати об'єкти, використовувати спадкування, поліморфізм та інші принципи ООП. Це дозволяє створювати більш структурований і повторно використовуваний код.

Python має велику стандартну бібліотеку, яка включає в себе багато корисних модулів і функцій для різних задач. Наприклад, можна використовувати модуль ‘math’ для математичних обчислень, ‘os’ для роботи з операційною системою, ‘datetime’ для роботи з датами і часом та багато інших. Крім того, існує велика кількість сторонніх бібліотек і фреймворків, що розширюють функціональність Python для різних галузей, таких як наука про дані, веб-розробка, штучний інтелект тощо.

Python працює на багатьох платформах, включаючи Windows, macOS і Linux. Це означає, що ви можете писати код на Python один раз і запускати його на різних операційних системах без необхідності вносити значні зміни.

Ця мова програмування застосовується в різних сферах, таких як веб-розробка, наука про дані, штучний інтелект, розробка ігор, мережеве програмування, автоматизація та багато інших. Його популярність і активне співтовариство розробників роблять Python однією з найпоширеніших мов програмування сьогодення.

3.2 Інтегроване середовище розробки PyCharm

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно було випущено в 2010 році і з тих пір стало одним з найпопулярніших інструментів для розробки на Python.

Історія PyCharm починається з того, що JetBrains вже мав успішні IDE для інших мов програмування, таких як Java (IntelliJ IDEA), PHP (PhpStorm) та інших. З великим попитом на Python серед розробників, компанія вирішила створити спеціалізоване IDE для цієї мови.

Перша версія PyCharm була представлена в 2010 році, і вона вразила спільноту Python розробників своїми можливостями та продуктивністю. З часом PyCharm став ще більш популярним і отримав широке визнання серед розробників Python завдяки своїм інноваціям та функціональності [16].

Основні можливості PyCharm включають:

1) Редактор коду: PyCharm надає потужний редактор коду з розумінням контексту, автодоповненням, підсвічуванням синтаксису та багатьма іншими функціями, які допомагають писати чистий та ефективний код;

2) Відлагодження: PyCharm має вбудовану систему відлагодження, яка дозволяє встановлювати точки зупинки, крокувати по коду, переглядати значення змінних та аналізувати стек викликів. Це допомагає виявляти й усувати помилки в програмі;

3) Управління проектами: PyCharm дозволяє легко створювати та керувати проектами на Python. Він автоматично розпізнає структуру проекту,

надає інструменти для управління залежностями, віртуальними середовищами та допомагає організувати робочий процес;

4) Рефакторинг: PyCharm має набір інструментів для автоматичного рефакторингу, що полегшує перерозподіл коду, зміну імен та оптимізацію структури проекту. Він допомагає покращити читабельність та якість коду.

5) Підтримка контролю версій: PyCharm має вбудовану підтримку популярних систем контролю версій, таких як Git, Mercurial, Subversion та інших. Він дозволяє виконувати коміти, синхронізувати зміни, розв'язувати конфлікти злиття гілок та багато іншого;

6) Підтримка веб-розробки: PyCharm надає зручність розробки веб-додатків з використанням Python, включаючи підтримку HTML, CSS, JavaScript та фреймворків, таких як Django та Flask;

7) Інтеграція з інструментами: PyCharm легко інтегрується з іншими популярними інструментами та сервісами, такими як бази даних, системи збирання проектів, сервіси хмарного хостингу та інші.

Зокрема, PyCharm надає дві версії: Community та Professional. Community версія є безкоштовною та має базовий набір функцій, тоді як Professional версія надає розширені можливості, такі як підтримка додаткових фреймворків, баз даних та інших інструментів.

Завдяки своїм потужним можливостям та зручному інтерфейсу, PyCharm став одним з найбільш улюблених інструментів для розробки на мові Python серед професіоналів та початківців розробників.

В даній програмі було створено поєкт “face_recognition»” (рис.3.1).

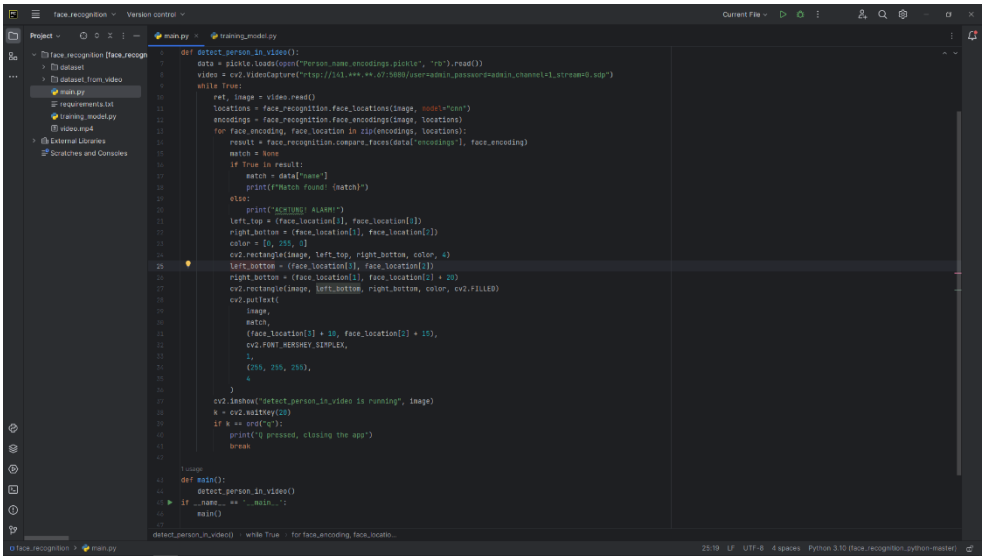


Рисунок 3.1 – Інтегроване середовище розробки PyCharm

Проєкт має таку структуру (рис. 3.2):

1) dataset – папка що зберігає в собі фото для тренування на певне обличчя;

2) dataset_from_video – папка, в яку зберігаються скріншоти обличчя, знятих з відеофайлу, для формування матеріалу для основного датасету;

3) main.py – основна частина з кодом розроюни системи розпізнавання обличчя та особи;

4) requirements.txt – файл, в я кому прописані залежності для проєкту;

5) trining_model.py – модуль для тренування моделі певної особи;

6) video.mp4 – відеоматеріал для створення датасету.

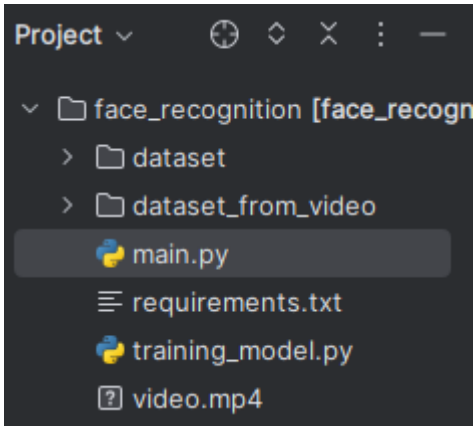


Рисунок 3.2 – Структура проєкта

3.3 Бібліотека OpenCV

OpenCV – це бібліотека комп'ютерного зору з відкритим вихідним кодом для аналізу, класифікації та обробки зображень. Вона широко використовується в таких мовах, як C, C++, Python та Java.

Для початку роботи з OpenCV треба в середовищі програмування в Python проєкті у консоль вписати таку команду [12]:

```
pip install opencv-python
```

Чекаємо поки бібліотека скачається та встановиться. Після цтого можна вже приступати до роботи з OpenCV [6].

3.3.1 Трохи про пікселі та колірні простори

Кожне зображення складається з набору пікселів. Піксель – це будівельний блок зображення. Якщо уявити зображення у вигляді сітки, то кожна клітинка сітки містить один піксель, де точка з координатами (0, 0) відповідає верхньому лівому куту зображення. Наприклад, припустимо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків і 300 стовпців. Разом наше зображення має 400×300 = 120 000 пікселів.

У більшості зображень пікселі представлені двома способами: у відтінках сірого і в колірному просторі RGB (рис.3.3). У відтінках сірого кожен піксель має значення від 0 до 255, де 0 відповідає чорному кольору, а 255 - білому. А значення між 0 і 255 набувають різних відтінків сірого, де значення ближче до 0 є темнішими, а значення ближче до 255 – світлішими.

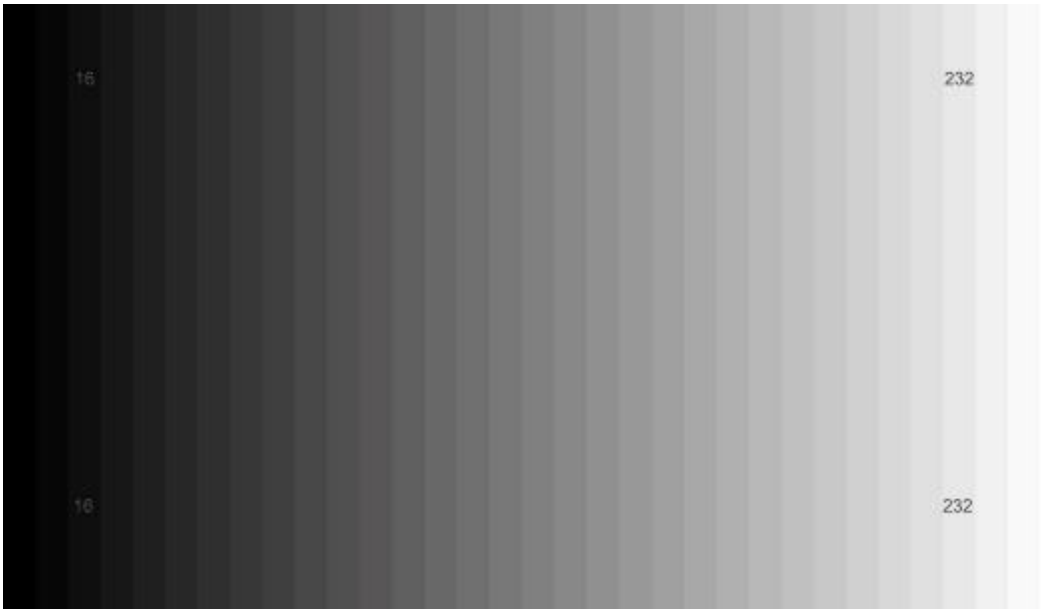


Рисунок 3.3 – Відтінки сірого

Кольорові пікселі зазвичай представлені в колірному просторі RGB (red, green, blue), де одне значення відповідає червоному компоненту, одне – зеленому, і одне – синьому. Кожна з трьох складових представлена цілим числом від 0 до 255 включно, яке вказує на те, скільки кольору міститься в пікселі. Припускаючи, що кожен компонент представлений в діапазоні [0,255], для представлення насиченості кожного кольору буде достатньо 8-бітового цілого без знаку. Потім ми об'єднуємо значення всіх трьох компонентів у кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонент повинна дорівнювати 255: (255, 255, 255). Тоді, щоб отримати чорний колір, кожен з компонентів повинен дорівнювати 0: (0, 0, 0) (рис.3.4) [10].

Рисунок 3.4 – кольоровий простір RGB

3.3.2 Імпорт бібліотеки OpenCV

Перше, що нам потрібно зробити, це імпортувати бібліотеку. Існує кілька способів імпорту, найпоширеніший з яких – використання виразу [11]:

```
import cv2
```

Існує й альтернативна конструкцію для імпорту цієї бібліотеки:

```
from cv2 import cv2
```

3.3.3 Завантаження, відображення та збереження зображення

```
def loading_displaying_saving():
    img = cv2.imread('girl.jpg', cv2.IMREAD_GRAYSCALE)
    cv2.imshow('girl', img)
    cv2.waitKey(0)
    cv2.imwrite('graygirl.jpg', img)
```

Для завантаження зображення ми використовуємо функцію *cv2.imread()*, де в першому аргументі вказується шлях до зображення, а в другому, необов'язковому, ми вказуємо, в якому колірному просторі ми хочемо прочитати наше зображення. Для читання зображення у RGB – *cv2.IMREAD_COLOR*, у відтінках сірого – *cv2.IMREAD_GRAYSCALE*. За

замовчуванням цей аргумент приймає значення *cv2.IMREAD_COLOR*. Ця функція повертає двовимірний (для зображення у відтінках сірого) або тривимірний (для кольорового зображення) масив NumPy. Форма масиву для кольорового зображення має вигляд висота x ширина x 3, де 3 - це байти, по одному байту для кожного компонента. У відтінках сірого все трохи простіше: висота x ширина.

За допомогою *cv2.imshow()* ми виводимо зображення на екран. Ми передаємо назву нашого вікна як перший аргумент функції, а зображення, яке ми завантажили з диска, як другий аргумент, але якщо не вказати *cv2.waitKey()* далі, то зображення миттєво закриється. Ця функція зупиняє виконання програми до тих пір, поки не буде натиснута клавіша, яку ми передали як перший аргумент. Для того, щоб будь-яка клавіша була врахована, передається 0.

Функцію *cv2.imwrite()* використовують для запису зображення у файл формату jpg (ця бібліотека підтримує всі популярні формати зображень: png, tiff, jpeg, bmp і т.д., тому ми можемо зберегти наше зображення у будь-якому з цих форматів), де першим аргументом є ім'я та розширення, а наступним параметром – саме зображення, яке ми хочемо зберегти.

3.3.4 Доступ до пікселів та маніпулювання ними

Для того, щоб дізнатися висоту, ширину та кількість каналів зображення можна використовувати атрибут *shape*:

```
print("Height:"+str(img.shape[0]))
print("Width:" + str(img.shape[1]))
print("Number of channels:" + str(img.shape[2]))
```

Важливо пам'ятати, що для зображень у відтінках сірого функція *img.shape[2]* буде недоступна, оскільки ці зображення представлені у вигляді 2D масиву.

Щоб отримати доступ до значення пікселя, нам достатньо вказати координати x та y пікселя, який нас цікавить. Важливо також пам'ятати, що бібліотека OpenCV зберігає канали RGB у зворотному порядку, тоді як ми мислимо в термінах червоного, зеленого і синього, OpenCV зберігає їх у порядку синього, зеленого і червоного:

```
(b, g, r) = img[0, 0]
print("Red: {}, Green: {}, Blue: {}".format(r, g, b))
```

Спочатку візьмемо піксель, який знаходиться в точці (0, 0). Цей піксель, як і будь-який інший піксель, представлений у вигляді кортежу. У наступному рядку виводимо значення кожного каналу на екран. Доступ до значень пікселів досить простий, і можна маніпулювати значеннями пікселів так само легко:

```
img[0, 0] = (255, 0, 0)
(b, g, r) = img[0, 0]
 print("Red: {}, Green: {}, Blue: {}".format(r, g, b))
```

У першому рядку встановлюємо значення пікселя (0, 0) на (255, 0, 0), потім знову беремо значення цього пікселя і виводимо його на екран, таким чином я отримую наступний вивід на консоль:

```
Red: 251, Green: 43, Blue: 65
Red: 0, Green: 0, Blue: 255
```

3.4 Принцип розпізнавання обличчя з фото OpenCV

Для початку треба розібратися, як розпізнати обличчя на фотографії. По-перше, потрібно знайти, де на фотографії розташоване обличчя людини, і не переплутати його з годинником на стіні та кактусом на підвіконні. Здавалося б, просте завдання для людини, але не таке вже й просте для комп'ютера. Для того, щоб знайти обличчя, треба виділити його основні складові, такі як ніс, лоб, очі, губи тощо. Для цього використовуються шаблони (так звані примітиви Хаара (рис.3.5)) [2] [15].

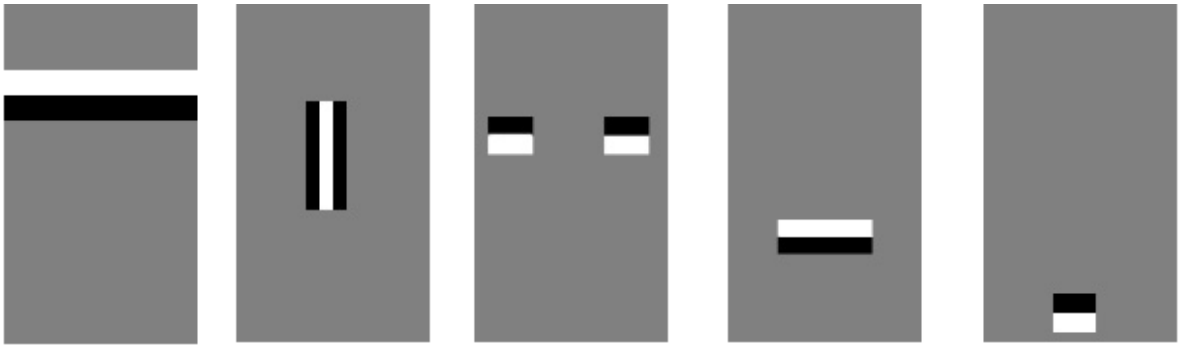


Рисунок 3.5 – Примітиви Хаара

Якщо візерунки відповідають певним ділянкам на зображенні, будемо вважати, що на ньому є людське обличчя. Насправді таких патернів набагато більше. Для кожного з них підраховується різниця між яскравістю білих і чорних ділянок. Це значення порівнюється з еталонним і приймається рішення, чи є на зображенні частина людського обличчя, чи ні.

Цей метод називається методом Віоли-Джонса (також відомий як каскади Хаара). Уявімо, що на зображенні не одне велике обличчя, а багато маленьких. Якщо ми застосуємо шаблони до всієї картинки, то не знайдемо жодного обличчя, тому що вони будуть меншими за шаблони. Для того, щоб шукати обличчя різного розміру на всій фотографії, використовується метод ковзного вікна. Саме в межах цього вікна відбувається обчислення примітивів. Вікно ніби ковзає по всьому зображенню. Після кожного проходження зображення вікно збільшується, щоб знайти більші обличчя (рис.3.6).

Рисунок 3.6 – Принцип роботи методу Віоли-Джонса

Отже, обличчя на зображенні знайдено, але як дізнатися, що це обличчя саме тієї людини, яку ми шукаємо? Для вирішення цієї проблеми ми скористаємося алгоритмом локальних бінарних шаблонів (Local Binary Patterns). Його суть полягає в тому, що ми ділимо зображення на частини і в кожній частині кожен піксель порівнюється з сусідніми 8 пікселями. Якщо значення центрального пікселя більше, ніж сусідніх, ми пишемо 0, інакше – 1. І так для кожного пікселя ми отримуємо деяке число (рис.3.7). Потім на основі цих чисел розраховується гістограма для всіх частин, на які ми розділили фотографію (рис.3.8). Всі гістограми з усіх частин об'єднуються у вектор, що характеризує зображення в цілому. Якщо ми хочемо дізнатися, наскільки схожі обличчя, ми повинні обчислити вектор для кожного з них і порівняти їх [8].

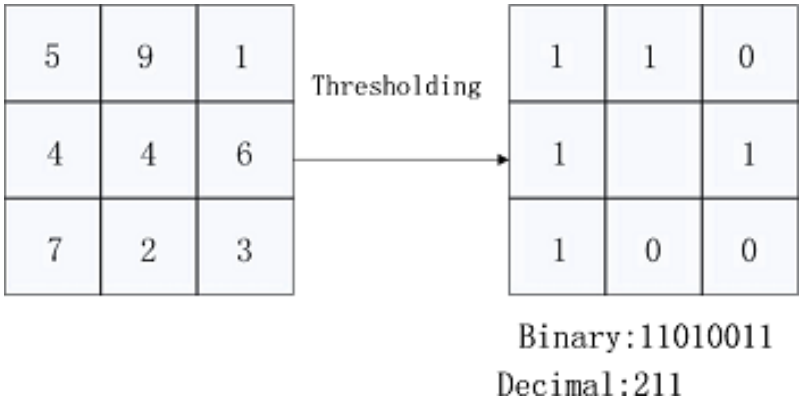


Рисунок 3.7 – Бінарізація пікселів

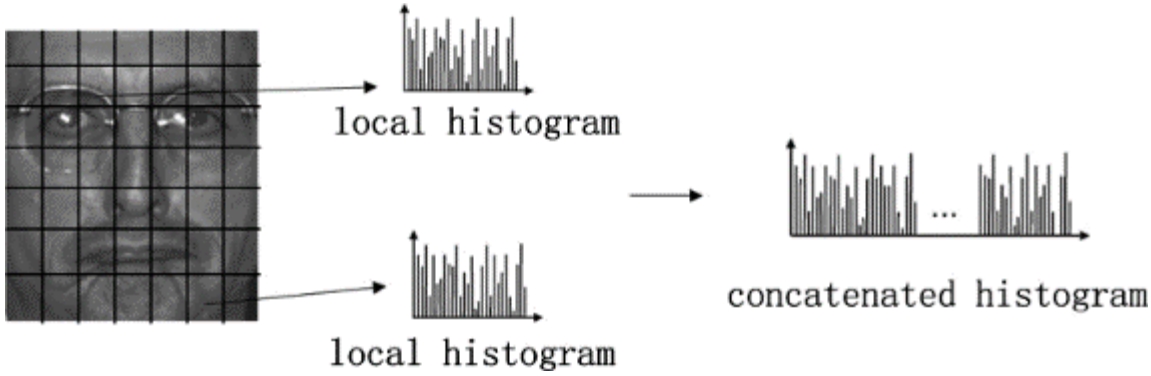


Рисунок 3.8 – Вихідна гістограма

Імпортуємо необхідні бібліотеки:

```
import cv2, os
```

```
import numpy as np
from PIL import Image
```

Для бібліотеки OpenCV існує доволі багато каскадів Хаара для розпізнавання не тільки обличь, а й предметів, номерів, емоцій тощо. Використаємо каскад для розпізнавання обличчя:

```
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
```

Ми використовуємо локальні бінарні шаблони для розпізнавання:

```
recognizer = cv2.createLBPHFaceRecognizer(1,8,8,8,123)
```

Параметр *cascadePath* містить ім'я файлу з уже підготовленими значеннями для розпізнавання обличь. Цей файл можна взяти з каталогу OpenCV (opencv\build\etc\haarcascades\).

Далі треба створити об'єкт *CascadeClassifier* і об'єкт розпізнавання облич *LBPHFaceRecognizer*. Розглянемо докладніше останній, а точніше, його параметри. Перші два значення 1 і 8 характеризують околиці пікселя (рис.3.9).

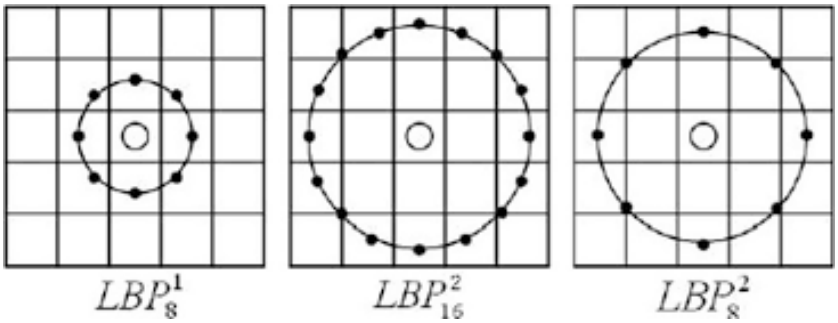


Рисунок 3.9 – Графічне зображення об’єкту LBPHFaceRecognizer

Тобто, перше число – це радіус, в якому ми вибираємо пікселі, а друге – кількість цих пікселів. Чим більше пікселів в околі точки, яку ми виділимо, тим точніше буде наше розпізнавання.

Наступні параметри (8,8) характеризують розмір областей, на які ми ділимо вихідне зображення з обличчям. Чим він менший, тим більше буде таких областей і тим якіснішим буде розпізнавання.

І, нарешті, останнє значення – це довірчий поріг, який визначає порогове значення для розпізнавання обличчя. Чим менший поріг, тим більша впевненість алгоритму в тому, що на фото зображено відоме обличчя. Поріг означає, що коли впевненість низька, алгоритм просто вважає це обличчя незнайомим. У цьому випадку поріг дорівнює 123.

Далі напишемо функцію, яка знаходить обличчя людей на всіх фотографіях і зберігає їх.

```
def get_images(path):
    image_paths = [os.path.join(path, f) for f in os.listdir(path) if not
f.endswith('.happy')]
    images = []
    labels = []

    for image_path in image_paths:
        gray = Image.open(image_path).convert('L')
        image = np.array(gray, 'uint8')

        subject_number                                                    =
int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))
        faces    =    faceCascade.detectMultiScale(image,    scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in faces:
            images.append(image[y: y + h, x: x + w])
            labels.append(subject_number)
            cv2.imshow("", image[y: y + h, x: x + w])
            cv2.waitKey(50)
    return images, labels
```

Для прикладу використано базу даних обличь під назвою Yale Faces. На кожній фотографії є 15 людей з різними виразами обличчя.

Кожен файл у цій базі даних має таку назву: subject01.sad. Спочатку йде слово суб'єкт, потім номер людини, а потім характеристика фотографії. Наприклад, характеристика sad означає сумне обличчя, happy - веселе і т.д.

Функція *get_images* зчитує кожну фотографію, окрім тих, що мають .happy в кінці, і виділяє область, де знаходиться обличчя. Фотографії зі щасливим виразом обличчя будуть використані на наступному кроці для

розпізнавання, це буде контрольна вибірка, тобто ті фотографії, на яких ми будемо перевіряти якість розпізнавання.

Також з кожного імені файлу витягується номер людини на фото і зберігається список міток. Згодом кожна фотографія буде пов'язана з цим номером.

Функція *faceCascade.detectMultiScale()* визначає області на фото, де є людські обличчя. Вона повертає список з параметрами [x,y,w,h] для кожного знайденого обличчя. Ці параметри описують прямокутну область в тому місці, де знайдено обличчя.

Тепер розберемо параметри функції:

– image – вихідне зображення;

– scaleFactor – визначає, на скільки буде збільшуватися вікно пошуку на кожній ітерації. 1.1 означає на 10%, 1.05 - на 5% і т.д. Чим більше це значення, тим швидше працює алгоритм;

– minNeighbors – чим більше це значення, тим більш параноїдальним буде пошук і тим частіше він буде пропускати реальні обличчя, вважаючи, що це помилкове спрацьовування. Оптимальне значення – 3-6;

– minSize - мінімальний розмір обличчя на фото. 30 на 30 зазвичай цілком достатньо.

Що ж, тепер ми можемо створити набір обличь і відповідних міток. Навчимо програму розпізнавати ці обличчя:

```
images, labels = get_images(path)
recognizer.train(images, np.array(labels))
```

Вказуємо шлях до наших фотографій, отримуємо список з фотографіями та підписами. А потім запускаємо нашу навчальну функцію за алгоритмом LBP. У цьому немає нічого надприродного, ми просто передаємо їй значення, отримані після запуску *get_images()*. Все інше програма зробить сама.

Отже, у нас є «розпізнавач» і набір щасливих обличь. Тепер потрібно попросити алгоритм розпізнати ці щасливі обличчя.

```
image_paths = [os.path.join(path, f) for f in os.listdir(path) if f.endswith('.happy')]

for image_path in image_paths:
    gray = Image.open(image_path).convert('L')
    image = np.array(gray, 'uint8')
    faces = faceCascade.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))


    for (x, y, w, h) in faces:
        number_predicted, conf = recognizer.predict(image[y: y + h, x: x + w])

        number_actual = int(os.path.split(image_path)[1].split(".")[0].replace("subject", ""))

        if number_actual == number_predicted:
            print "{} is Correctly Recognized with confidence {}".format(number_actual, conf)
        else:
            print "{} is Incorrect Recognized as {}".format(number_actual, number_predicted)
        cv2.imshow("Recognizing Face", image[y: y + h, x: x + w])
        cv2.waitKey(1000)
```

У циклі ми знову визначаємо розташування обличчя на кожній фотографії з закінченням .happy. Всі параметри та процедури такі ж, як і в попередньому кроці.

Для кожного знайденого обличчя запускаємо функцію *recognizer.predict()*, яка повертає номер-ідентифікатор суб'єкта, який імовірно є на фото, а також параметр довіри. Потім ми порівнюємо повернуте функцією значення з реальним номером об'єкта, і якщо вони збігаються, то розпізнавання вважається успішним.

Все, далі на консолі виводяться результати розпізнавання для кожної фотографії з контрольної вибірки.

```
> face_recognizer.py
1 is Correctly Recognized with confidence 3.60679904352
2 is Correctly Recognized with confidence 2.7062187167
3 is Correctly Recognized with confidence 3.47226623852
4 is Correctly Recognized with confidence 3.44204933749
5 is Correctly Recognized with confidence 3.85364810403
6 is Correctly Recognized with confidence 3.4013632496
7 is Correctly Recognized with confidence 3.67291258578
8 is Correctly Recognized with confidence 3.93510466561
9 is Correctly Recognized with confidence 3.23037308296
10 is Correctly Recognized with confidence 4.83246249165
11 is Correctly Recognized with confidence 3.35613808459
12 is Correctly Recognized with confidence 14.7131008
13 is Correctly Recognized with confidence 3.20571681599
14 is Correctly Recognized with confidence 2.59308289024
15 is Correctly Recognized with confidence 3.85130889801
```

Рисунок 3.10 – Результат виконання програми

3.5 Система розпізнавання обличчя з відеопотоку

Для виконання поставленої задачі необхідно створити датасет персони, яку ми прагнемо знайти у відеопотоці. Але основною проблемою є необхідність отримання бази матеріалу, а саме певної кількості фотографій персони, де буде повністю видно його обличчя. Для початку треба назбирати певний об’єм фото, десь порядку 8 штук для датасету. Можна скористатися й відеофайлов, на якому видно профіль обличчя кандидата на розпізнавання.

Будемо виймати скріншоти з відоепотоку з вибіркою в 3 кадри, для цього напишемо функцію *take_screenshot_from_video()*:

```
def take_screenshot_from_video():
    cap = cv2.VideoCapture("video.mp4")
    count = 0
    if not os.path.exists("dataset_from_video"):
        os.mkdir("dataset_from_video")
    while True:
        ret, frame = cap.read()
        fps = cap.get(cv2.CAP_PROP_FPS)
        multiplier = fps * 3
        if ret:
            frame_id = int(round(cap.get(1)))
            cv2.imshow("frame", frame)
```

```
            k = cv2.waitKey(20)
            if frame_id % multiplier == 0:
                cv2.imwrite(f"dataset_from_video/{count}.jpg", frame)
                print(f"Take a screenshot {count}")
                count += 1
            if k == ord(" "):
                cv2.imwrite(f"dataset_from_video/{count}_extra_scr.jpg", frame)
                print(f"Take an extra screenshot {count}")
                count += 1
            elif k == ord("q"):
                print("Q pressed, closing the app")
                break
        else:
            print("[Error] Can't get the frame...")
            break
    cap.release()
    cv2.destroyAllWindows()
```

Метод *.read()* захоплює, декодує та повертає наступний кадр, таким чином якщо кадр буде прочитаний коректна значення *ret* буде true, якщо ні – false. При правильному причитуванні кадру будемо визивати метод *cv2.imshow()*, де передаємо назву та сам фрейм. Далі в методі *cv2.waitKay()* вказуємо значення в мілісекундаї, що дозволяє відстежувати натискання клавіш й регулювати швидкість програвання відео. За допомогою *cv2.imwrite()* будемо зберігати скріншоти в створену директорію dataset_from_video (рис.3.11).

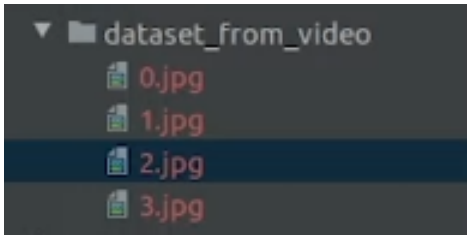


Рисунок 3.11 – Зберігання скріншотів в dataset_from_video

Таким чином ми отримуємо необхідні матеріали для тренування нашої мережі.

Після того як ми отримали всі необхідні зображення обличчя людини, напишемо код, що створить шаблон для розпізнавання певної особи. Для досягнення цього будемо використовувати додаткову бібліотеку face_recognition. Створимо функцію train_model_by_img, в якій опишемо метод створення кодування:

```
def train_model_by_img(name):
    if not os.path.exists("dataset"):
        print("[ERROR] there is no directory 'dataset'")
        sys.exit()
    known_encodings = []
    images = os.listdir("dataset")

    for(i, image) in enumerate(images):
        print(f"[+] processing img {i + 1}/{len(images)}")
        face_img = cv2
        face_recognition.load_image_file(f"dataset/{image}")
        face_enc = face_recognition.face_encodings(face_img)[0]
        if len(known_encodings) == 0:
            known_encodings.append(face_enc)
        else:
            for item in range(0, len(known_encodings)):
                        result  =  face_recognition.compare_faces([face_enc],
known_encodings[item])
                if result[0]:
                    known_encodings.append(face_enc)
                    break
                else:
                    break
    data = {
        "name": name,
        "encodings": known_encodings
    }

    with open(f"{name}_encodings.pickle", "wb") as file:
        file.write(pickle.dumps(data))
    return f"[INFO] File {name}_encodings.pickle successfully created"
```

Для початку перевіряємо наявність директорії dataset. Якщо вона знайдена, то можна починати кодування. У циклі загружаємо зображення за

допомогою методу *face_recognition.load_image_file()* та наступним кроком отримуємо кодування обличчя вже за допомогою методу *face_recognition.face_encodings()* (рис.3.12).

```
[+] processing img 1/7
[-0.06699492  0.05593485 -0.01381714 -0.03522641 -0.04684281  0.06892829
 -0.0919052  -0.02975844  0.1226986  -0.09450018  0.21448058 -0.0267286
 -0.17967293  0.01346489 -0.12635115  0.23381606 -0.17092673 -0.09158731
 -0.10697323 -0.03170604  0.06050514  0.05520367  0.03975895  0.08900684
 -0.14798157 -0.38178921  0.04166758 -0.10291115 -0.04939741 -0.1004282
 -0.01377897  0.1052064  -0.18413119  0.03261381  0.03150034  0.1284419
 -0.03216095 -0.07866904  0.1777885   0.09813731 -0.23536603 -0.02394804
  0.08712308  0.33068323  0.22467186 -0.02833913  0.02766823 -0.05684498
```

Рисунок 3.12 – Вигляд кодування обличчя

Тепер нам необхідно влаштувати перевірку. Кожна наступне кодування повинно зіставляти себе з попередньою. Якщо результат цього зіставлення буде true, то тоді добавляємо нове кодування до списку, таким чином розширюючи точність розпізнавання. За допомогою конструкції *face_recognition.compare_faces([face_enc], known_encodings[item])* у кожній ітерації зіставляємо отримане кодування нового зображення з вже існуючими всередині списку.

В підсумку ми повинні отримати словник, першою парою якого є ім’я людини та список з розпізнаних кодувань обличчя. Дані перетворимо в поток байтів й збережемо в вигляді pickle файлу.

Модуль ‘pickle’ є вбудованим модулем в мові програмування Python і використовується для серіалізації (перетворення у байтовий потік) та десеріалізації (відновлення з байтового потоку) об'єктів Python. Цей модуль дозволяє зберігати об'єкти Python у файлі або передавати їх по мережі, зберігаючи при цьому їх структуру та стан.

Модуль ‘pickle’ дозволяє серіалізувати та десеріалізувати різні типи об'єктів Python, включаючи числа, рядки, списки, словники, класи та навіть складні об'єкти зі взаємозалежностями. Він також підтримує серіалізацію об'єктів з використанням різних протоколів, залежно від потреб користувача.

Проте важливо зазначити, що при використанні модуля ‘pickle’ необхідно бути обережним, оскільки він може виконувати код, що знаходиться в серіалізованих об'єктах. Це може створювати потенційні проблеми з безпекою, якщо серіалізовані об'єкти походять з ненадійного джерела.

Загалом, модуль ‘pickle’ є потужним інструментом для збереження та передачі об'єктів Python та дозволяє легко працювати зі структурованими даними в мові програмування Python.

Після підготувань необхідних даних можна перейти до обробки зображення безпосередньо з IP-камери. Напишемо функцію *detect_person_in_video()*. Спочатку передамо файл pickle, що ми створили раньше:

```
data = pickle.loads(open("Person_name_encodings.pickle", "rb").read())
```

Далі необхідно вказати основний відеопотік, з яким ми будемо працювати. Передаємо IP-адресу камери, та логін й пароль для доступу:

```
cv2.VideoCapture("rtsp://141.***.**.67:5080/user=admin_password=admin_channel=1_stream=0.sdp")
```

У циклі викликаємо метод *.read()*, що захоплює, декодує та повертає наступний кадр. Далі використовуємо метод *face_recognition.face_locations(image, model="cnn")* для отримання масиву координат людських обличь на відео. До речі можна вказати модель: cnn підходить для роботи з відеокартами, а hog – для роботи з процесорами.

Головною задачею тепер є зіставлення кодувань обличь зі створеного нами датасету з обличчями, які ми отримуємо з кожного кадру у даному відеопотоці. І, коли виникає коректне зіставлення, сигналізувати та виділяти обличчя на відео.

```
for face_encoding, face_location in zip(encodings, locations):
                                                    result = face_recognition.compare_faces(data["encodings"], face_encoding)
```

```
            match = None
            if True in result:
                match = data["name"]
                print(f"Match found! {match}")
            else:
                print("ACHTUNG! ALARM!")
```

Тепер необхідно намалювати рамку на відео. Ми вже маємо координати розпізнаного обличчя, тому це не складатиме труднощів.

```
left_top = (face_location[3], face_location[0])
right_bottom = (face_location[1], face_location[2])
color = [0, 255, 0]
cv2.rectangle(image, left_top, right_bottom, color, 4)
```

Тепер виведемо ще й ім’я особи під рамкою. В метод *cv2.putText()* передаємо кадр, ім’я, координати та форматування самого тексту.

```
cv2.putText(
                image,
                match,
                (face_location[3] + 10, face_location[2] + 15),
                cv2.FONT_HERSHEY_SIMPLEX,
                1,
                (255, 255, 255),
                4
            )
```

В кінці ми повинні отримати зображення з рамкою в області обличчя та підписом з ім’ям (рис.3.13). Також в консолі маємо відповідне повідомлення.

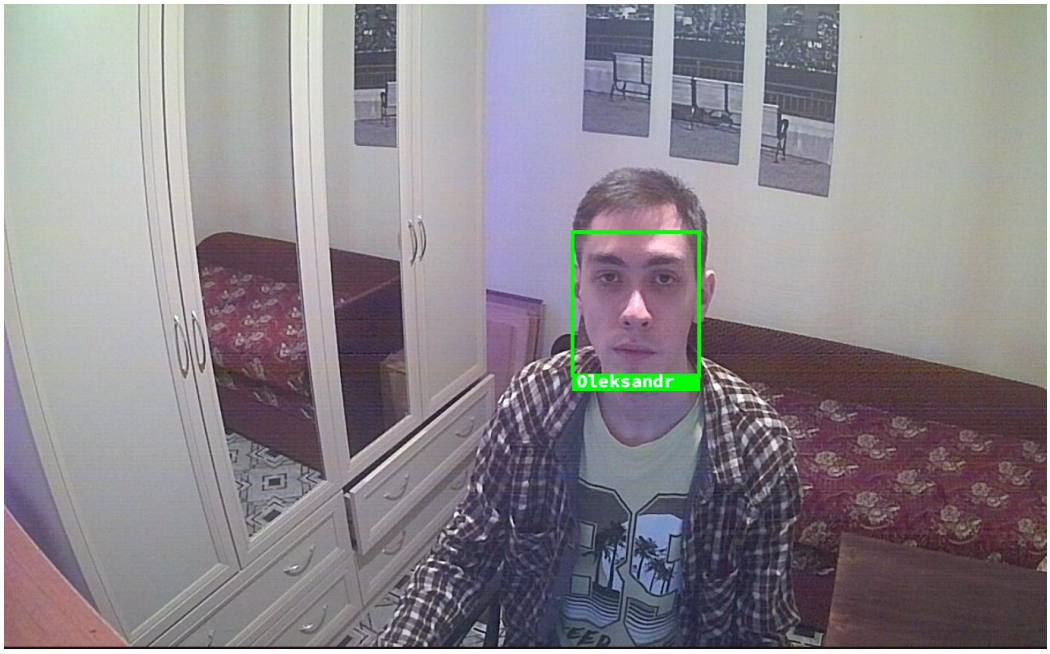


Рисунок 3.13 – Результат роботи програми

Висновки до розділу 3

OpenCV - це потужна бібліотека комп'ютерного зору, яка надає багато засобів для обробки зображень. У ній функції для завантаження та відображення зображень, зміни їх розміру, кадрування областей, конвертації кольорових просторів, фільтрації, виявлення контурів, бінаризації та багато інших операцій.

Завдяки OpenCV ви можете легко виконувати різні завдання з обробки зображень, такі як виявлення обличь, визначення особливостей, розпізнавання об'єктів, аналіз фреймів відео, покадрова обробка та багато іншого. Бібліотека надає зручні інструменти для роботи з зображеннями, дозволяючи вам створювати складні обробки з мінімальними зусиллями.

За допомогою методів витягування особливостей, таких як метод головних компонентів (PCA) або локальні бінарні шаблони (LBP), можна виявляти і описувати унікальні особливості обличь, що дозволяє розпізнавати їх на зображеннях. OpenCV надає функції для виконання цих методів та інших алгоритмів витягування особливостей.

Крім того, OpenCV підтримує використання машинного навчання для розпізнавання обличь. Ви можете навчити класифікатор, використовуючи алгоритми, такі як метод опорних векторів (SVM) або нейронні мережі, і застосовувати його для розпізнавання обличь на нових зображеннях. OpenCV надає інструменти для навчання класифікаторів та застосування їх для розпізнавання обличь.

Також OpenCV підтримує використання попередньо навчених моделей для розпізнавання обличь, таких як моделі, навчені на наборах даних, таких як OpenFace, Dlib та інші. Ви можете використовувати ці моделі для розпізнавання обличь з високою точністю [3].

Загалом, OpenCV надає багато різних методів та інструментів для розпізнавання обличь, що робить його потужним інструментом для реалізації систем розпізнавання обличь у проекті.

ВИСНОВКИ

Метою цієї дипломної роботи було створення автоматизованої системи моніторингу приміщення за допомогою IP-камер та OpenCV, що б забезпечити додатковий захист та запобігання несанкціонованого доступу.

Перш за все було досліджено та проаналізовано використання таких систем в сучасному світі. Описані області та сфери мають величезні рамки використання моніторингу приміщення, але не включають особливої автоматизації та додаткової обробки зображення. Також було проаналізовано апаратне забезпечення, що включає великий спектр видів IP-камер. Крім того, було проведено дослідження та порівняння різних методів розпізнавання обличь, а також розроблення ефективних алгоритмів виявлення та слідкування за обличчями. Відповідно до експериментальних результатів, система демонструє високу точність і надійність при розпізнаванні обличь та виявленні подій.

Наступним кроком було розроблено план приміщення, в якому буде встановлено відеоспостереження. За допомогою прогами IP Video System Design Tool було змодельоване розташування камер, їх робочі видимі зони та налагодження системи моніторингу приміщення. Було успішно налаштовано IP-камери для передачі відеопотоку на сервер, що дозволило отримувати зображення в реальному часі з різних куточків приміщення.

Далі, було використано OpenCV для розробки алгоритмів обробки обличчя у відеопотоці, що дозволить виявляти необхідних людей та запобігати несанкціонований доступу. Система дозволяє автоматично розпізнавати обличчя та зіставляти з вже натренованими моделями людей для забезпечення доступу та відказу для посторонніх. Було реалізовано функціональні можливості, такі як сповіщення про виявлення підозрілого суб’єкту, контроль доступу до приміщень на основі розпізнавання обличь, та візуалізацію даних за допомогою звітів.

В цілому, бакалаврська кваліфікаційна робота підтверджує, що автоматизація моніторингу приміщень за допомогою IP-камер та OpenCV є потужним інструментом для забезпечення безпеки та нагляду. Завдяки використанню сучасних технологій, таких як IP-камери та OpenCV, ми можемо покращити ефективність моніторингу, забезпечити вчасну реакцію на потенційні загрози та зменшити ризики виникнення негативних ситуацій. Результати роботи вносять вагомий внесок у розвиток сучасних систем безпеки та створюють основу для подальших досліджень і вдосконалення даної технології.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Pulli K., Baksheev A., Kornyakov K., Eruhimov V. Real-Time Computer Vision with OpenCV. *Communications of the ACM*. 2012. P. 61–69. DOI: 10.1145/2184319.2184337.

2. Ainampudi K. S., Kadavakollu S., Kothamasu S. P., Vasantha B. An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python. IEEE. 2023. P. 23–52. DOI: 10.1109/ICACCS57279.2023.10113066.

3. Boyko N., Basystiuk O., Shakhovska N. Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. IEEE. 2018. P. 74. DOI: 10.1109/DSMP.2018.8478556.

4. Face detection using OpenCV and Python. URL: https://www.superdatascience.com/blogs/opencv-face-detection (Last accessed: 10.06.2023).

5. Face Recognition Using OpenCV and Python. URL: https://www.superdatascience.com/opencv-face-recognition/ (Last accessed: 10.06.2023).

6. Krishna M., Nabhan T. Y. Study and Analysis of Implementing a Smart Attendance Management System Based on Face Recognition Tecqnique using OpenCV and Machine Learning. IEEE. 2021. P. 412-434. DOI: 10.1109/CSNT51715.2021.9509614.

7. Bideau, P., Learned-Miller, E. It's Moving! A Probabilistic Model for Causal Motion Segmentation in Moving Camera Videos. Lecture Notes in Computer Science(). 2016. ECCV 2016. Vol. 9912. Springer, Cham. P. 433–449. DOI: 10.1007/978-3-319-46484-8_26.

8. Rath S. Segment Anything – A Foundation Model for Image Segmentation. URL: https://learnopencv.com/segment-anything/ (Last accessed: 10.06.2023).

9. G. Hu, F. Yan, C. H. Chan, W. Deng, W. Christmas, J. Kittler & N. M. Robertson. Face Recognition Using a Unified 3D Morphable Model. European Conference on Computer Vision. 2016. DOI: 10.1007/978-3-319-46484-8_5

10. S. Saito, T. Li & H. Li. Real-Time Facial Segmentation and Performance Capture from RGB Input. 2016. European Conference on Computer Vision. URL: https://link.springer.com/chapter/10.1007/978-3-319-46484-8_15

11. OpenCV documentation. URL: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html (Last accessed: 10.06.2023).

12. OpenCV modules. URL: https://docs.opencv.org/4.x/ (Last accessed: 10.06.2023).

13. Object Detection Made Easy with TensorFlow Hub: Step-by-Step Tutorial. URL: https://learnopencv.com/object-detection-tensorflow-hub/ (Last accessed: 10.06.2023).

14. J. Phys. Multi-View Faces Detection Using Viola-Jones Method. Journal of Physics: Conference Series. 2018. P. 1-9. DOI: 10.1088/1742-6596/1114/1/012068.

15. Сучасні IP камери відеоспостереження: основні критерії вибору. URL: https://asisvok.com.ua/blog/item/suchasni-ip-kamery-videosposterezhennia-osnovni-kryterii-vyboru. (Дата звернення: 10.06.2023).

16. PyCharm Features. URL: https://www.jetbrains.com/pycharm/features/. (Last accessed: 10.06.2023).

17. JVSG: Video Surveillance Design Apps. URL: https://www.jvsg.com. (Last accessed: 10.06.2023).

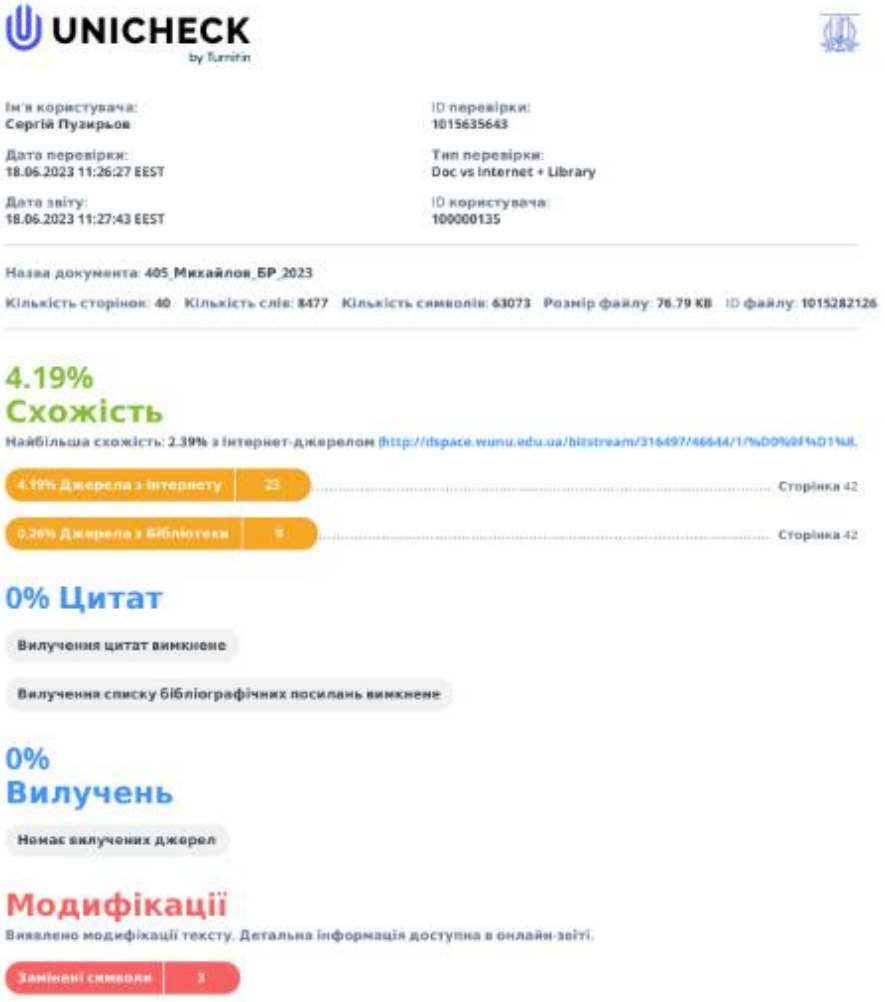
18. Застосування IP-відеоспостереження та ip-камер. URL: https://видеокамеры.com.ua/zastosyvannya_ip_videosposterejennya/. (Дата звернення: 10.06.2023).

ДОДАТОК А

ДОВІДКА

про перевірку на унікальність пояснювальної записки
кваліфікаційної бакалаврської роботи
на тему: «Побудова нейронної мережі для детектування малопомітних рухомих об'єктів»
студента спеціальності 123 «Комп'ютерна інженерія»,
групи 405
Михайлов Олександр Олександрович
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck.
Результат перевірки тексту роботи: схожість складає 4.19%.



Студент

_____ О. О. Михайлов
підпис ініціали, прізвище

Керівник
канд. фіз.-мат. наук, доцент

_____ С. В. Пузирьов
підпис ініціали, прізвище

Дата: «___»_____2023 р.

ДОДАТОК Б

Код основної функції для розпізнавання обличчя

Код основної функції для розпізнавання обличчя main:

```
import face_recognition
from PIL import Image, ImageDraw
import pickle
from cv2 import cv2

def detect_person_in_video():
    data = pickle.loads(open("Person_name_encodings.pickle", "rb").read())
                                                video                          =
cv2.VideoCapture("rtsp://141.***.**.67:5080/user=admin_password=admin_channel=
1_stream=0.sdp")
    while True:
        ret, image = video.read()
        locations = face_recognition.face_locations(image, model="cnn")
        encodings = face_recognition.face_encodings(image, locations)
        for face_encoding, face_location in zip(encodings, locations):
                  result  =  face_recognition.compare_faces(data["encodings"],
face_encoding)
            match = None
            if True in result:
                match = data["name"]
                print(f"Match found! {match}")
            else:
                print("ACHTUNG! ALARM!")
            left_top = (face_location[3], face_location[0])
            right_bottom = (face_location[1], face_location[2])
            color = [0, 255, 0]
            cv2.rectangle(image, left_top, right_bottom, color, 4)
            left_bottom = (face_location[3], face_location[2])
            right_bottom = (face_location[1], face_location[2] + 20)
            cv2.rectangle(image, left_bottom, right_bottom, color, cv2.FILLED)
            cv2.putText(
                image,
                match,
                (face_location[3] + 10, face_location[2] + 15),
                cv2.FONT_HERSHEY_SIMPLEX,
                1,
```

```
                (255, 255, 255),
                4
            )
        cv2.imshow("detect_person_in_video is running", image)
        k = cv2.waitKey(20)
        if k == ord("q"):
            print("Q pressed, closing the app")
            break

def main():
    detect_person_in_video()
if __name__ == '__main__':
    main()
```

ДОДАТОК В

Код функції для тренування моделі

Код функції для тренування моделі training_model:

```
import os
import pickle
import sys
from cv2 import cv2

def train_model_by_img(name):
    if not os.path.exists("dataset"):
        print("[ERROR] there is no directory 'dataset'")
        sys.exit()
    known_encodings = []
    images = os.listdir("dataset")

    for(i, image) in enumerate(images):
        print(f"[+] processing img {i + 1}/{len(images)}")
        face_img = cv2
        face_recognition.load_image_file(f"dataset/{image}")
        face_enc = face_recognition.face_encodings(face_img)[0]
        if len(known_encodings) == 0:
            known_encodings.append(face_enc)
        else:
            for item in range(0, len(known_encodings)):
                        result  =  face_recognition.compare_faces([face_enc],
known_encodings[item])
                if result[0]:
                    known_encodings.append(face_enc)
                    break
                else:
                    break
    data = {
        "name": name,
        "encodings": known_encodings
    }

    with open(f"{name}_encodings.pickle", "wb") as file:
        file.write(pickle.dumps(data))
    return f"[INFO] File {name}_encodings.pickle successfully created"
```

```
def take_screenshot_from_video():
    cap = cv2.VideoCapture("video.mp4")
    count = 0
    if not os.path.exists("dataset_from_video"):
        os.mkdir("dataset_from_video")
    while True:
        ret, frame = cap.read()
        fps = cap.get(cv2.CAP_PROP_FPS)
        multiplier = fps * 3
        if ret:
            frame_id = int(round(cap.get(1)))
            cv2.imshow("frame", frame)
            k = cv2.waitKey(20)
            if frame_id % multiplier == 0:
                cv2.imwrite(f"dataset_from_video/{count}.jpg", frame)
                print(f"Take a screenshot {count}")
                count += 1
            if k == ord(" "):
                cv2.imwrite(f"dataset_from_video/{count}_extra_scr.jpg", frame)
                print(f"Take an extra screenshot {count}")
                count += 1
            elif k == ord("q"):
                print("Q pressed, closing the app")
                break
        else:
            print("[Error] Can't get the frame...")
            break
    cap.release()
    cv2.destroyAllWindows()

def main():
    print(train_model_by_img("Alex"))
if __name__ == '__main__':
    main()
```