Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Розробка системи розумного доступу до виробничих приміщень з використанням технологій комп'ютерного зору
(тема)

Виконав здобувач 2 року навчання,
групи КТРСм-23-2

Водяницький М.А.
(прізвище, ініціали)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та роботехніка
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник проф. Омаров Ш.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТАР _____ Невлюдов І. Ш.
(підпис) (прізвище, ініціали)

Харків 2025

Харківський національний університет радіоелектроніки

Факультет	Автоматики і комп'ютеризованих технологій
Кафедра	Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти	другий (магістерський)
Спеціальність	174 Автоматизація, комп'ютерно-інтегровані технології та робототехніки
Тип програми	освітньо-професійна
Освітня програма	Комп'ютеризовані та робототехнічні системи

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Водяницькому Михайлу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи розумного доступу до виробничих приміщень з використанням технологій комп'ютерного зору
затверджена наказом по університету від 25.11. 2024 р. №1239 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 22.01. 2025 р.
3. Вихідні дані до роботи: 3.1 ESP32-cam; 3.2 Python; 3.3 OpenCV;
3.3 OpenCV; 3.4 Face Recognition;
3.5 Розумний доступ;
3.6 Комп'ютерний зір.
4.Перелік питань, що потрібно опрацювати в роботі:
4.1 Вступ;
4.2 Аналіз предметної області;
4.3 Розробка системи розпізнавання;
4.4 Експериментальна частина;
4.5 Розрахункова частина;
4.6 Охорона праці;

4.7 Висновки;

4.8 Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 17 с. формату А4

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз вимог до дипломнної роботи	05.09.2024-15.09.2024	виконано
2	Аналіз предметної області	16.09.2024-25.09.2024	виконано
3	Вивчення методів фільтрації	26.09.2024-05.10.2024	виконано
4	Вивчення будови та принципу роботи CNN мереж	06.10.2024-15.10.2024	виконано
5	Опис вимог до системи розумного доступу	16.10.2024-20.10.2024	виконано
6	Розробка систем розпізнавання	21.10.2024-20.11.2024	виконано
7	Провдення експериментальних вимірів	21.11.2024-10.12.2024	виконано
8	Аналіз отриманих результатів	11.12.2024-05.01.2025	виконано
9	Оформлення кваліфікаційної роботи	06.01.2025-22.01.2025	виконано
10	Передача роботи на перевірку	23.01.2025	виконано

Дата видачі завдання 5 вересня 2024р.

Здобувач _____ Водяницький М.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Омаров Ш.А.
(підпис) (прізвище, ініціали)

Я, як студент ХНУРЕ, розумію та підтримую політику закладу академічної доброчесності. Я не надавав та не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

05.01.2025  Водяницький М.А.

РЕФЕРАТ

Пояснювальна записка 88 с., 7 табл., 40 рис., 4 дод., 30 джерел.

СИСТЕМА РОЗУМНОГО ДОСТУПУ, КОМП'ЮТЕРНИЙ ЗІР, ВИРОБНИЦТВО, БЕЗПЕКА, РОЗПІЗНАВАННЯ ОБЛИЧЧЯ, ESP32, PYTHON, OPENCV.

Мета кваліфікаційної роботи – підвищення безпеки доступу на підприємстві, мінімізуючи потребу у фізичному контролі шляхом розробки системи розумного доступу до виробничих приміщень на основі технологій комп'ютерного зору.

Об'єкт дослідження –доступ до виробничих приміщень для контролю безпеки та автоматизації процесів.

Предмет дослідження – методи та алгоритми комп'ютерного зору для розпізнавання обличчя та інтеграція цих методів у систему контролю доступу до виробничих приміщень.

У роботі проаналізовано основні функції фільтрації зображення, принцип роботи CNN мереж. На основі проаналізованих системи розумного доступу інших авторів, були розроблені вимоги до власної системи, на основі яких було створено декілька систем розпізнавання на базі OpenCV та dlib. Було проведено експеримент з метою знаходження найбільш точного методу розпізнавання. Отримані результати були піддані багатофакторному аналізу, для встановлення залежності між освітленням і точністю, та між точністю та роздільною здатністю.

ABSTRACT

Explanatory note: 88 pages, 7 tables, 40 figures, 4 app, 30 sources.

INTELLIGENT ACCESS SYSTEM, COMPUTER VISION, MANUFACTURING, SECURITY, FACIAL RECOGNITION, ESP32, PYTHON, OPENCV.

The purpose of the qualification work is to improve access security at the enterprise, minimising the need for physical control by developing a system of smart access to industrial premises based on computer vision technologies.

The object of research is access to production facilities for security control and process automation.

The subject of research is methods and algorithms of computer vision for face recognition and integration of these methods into the access control system for industrial premises.

The paper analyses the main functions of image filtering and the principle of CNN networks. Based on the analysed smart access systems of other authors, the requirements for our own system were developed, on the basis of which several recognition systems based on OpenCV and dlib were created. An experiment was conducted to find the most accurate recognition method. The results were subjected to multivariate analysis to determine the relationship between illumination and accuracy, and between accuracy and resolution.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ФВЧ – фільтр високих частот;

ФНЧ – фільтр низьких частот;

CNN – Convolution Neural Networks;

PCA – principal component analysis.

ВСТУП

У сучасному світі питання безпеки на виробничих підприємствах набуває особливої актуальності. Зростаюча складність виробничих процесів, збільшення кількості персоналу та підвищені вимоги до охорони праці вимагають впровадження нових, більш ефективних та надійних систем контролю доступу. Традиційні методи, такі як фізична охорона та пропускні системи на основі карток, часто є недостатньо гнучкими та можуть бути вразливими до людського фактору. Розробка інтелектуальних систем, здатних автоматично ідентифікувати персонал та контролювати доступ до приміщень, є не лише кроком до підвищення безпеки, але й до оптимізації виробничих процесів, зменшуючи залежність від ручної праці та мінімізуючи ризики помилок.

Технологічний прогрес у сфері комп'ютерного зору відкриває нові можливості для автоматизації та інтелектуалізації систем безпеки. Застосування методів розпізнавання обличчя дозволяє створити системи доступу, які є більш надійними, швидкими та зручними порівняно з традиційними методами. Впровадження таких систем є не лише актуальним, а й необхідним кроком для сучасних підприємств, що прагнуть до інновацій та ефективності.

Мета дослідження – підвищення безпеки доступу на підприємстві, мінімізуючи потребу у фізичному контролі шляхом розробки системи розумного доступу до виробничих приміщень на основі технологій комп’ютерного зору.

Об’єкт дослідження –доступ до виробничих приміщень для контролю безпеки та автоматизації процесів.

Предмет дослідження – методи та алгоритми комп’ютерного зору для розпізнавання обличчя та інтеграція цих методів у систему контролю доступу до виробничих приміщень.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області;
- провести аналіз сучасних методів фільтрації зображення;

- провести аналіз методів розпізнавання обличчя;
- провести аналіз роботи та реалізації CNN мереж;
- провести аналіз вимог до системи розумного доступу;
- розробити схему принципу роботи системи;
- обґрунтувати вибір мови програмування та середовища розробки;
- розробити системи розпізнавання на базі OpenCV;
- розробити систему розпізнавання на базі Face Recognition;
- розробити електричну схему системи безпеки на базі ESP32-cam;
- провести експериментальні дослідження розроблених систем;
- провести аналіз та зробити розрахунки результатів експерименту;

Кваліфікаційна робота виконана згідно ДСТУ 3008 – 15 [1], керуючись навчальним посібником з дипломного проекту [2] та методичними вказівками [3], а результати кваліфікаційної роботи отримали апробацію в науковій статті [4].

Важлива зазначити, що дана кваліфікаційна робота, відповідає ціялям сталого розвитку, а саме цілі 9 «Промисловість, інновації та інфраструктура», виконуючи завдання цілі 9.4, сприяючи розвитку інноваційної екосистеми [5].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Теорія розпізнавання образів

Розвиток мікроелектроніки у двадцять першому сторіччі, у сфері відеокарт, дав великий поштовх у розвитку систем комп'ютерного зору. Комп'ютерній зір (computer vision) – теорія та технологія створення машин, які можуть проводити виявлення, вистежування та визначення об'єктів [6]. Як наукова дисципліна комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень. Головним користувачем даної теорії у наш час є нейронні мережі. Збільшення потужності відеокарт дозволило навчати мережі швидше та на більшій кількості вхідних даних. Завдяки цьому нейронні мережі можуть зчитувати інформацію з зображень за долі секунди, оброблювати її та видавати відповідь. Наприклад можна надіслати мережі фото листку з задачею по фізиці і мережа знайде рішення. Але до буму їх розвитку, інформація з зображення зчитувалась іншими технологіями, які підпорядковуються теорії розпізнавання образів.

Теорія розпізнавання образів – розділ кібернетики, що розвиває теоретичні основи й методи класифікації і ідентифікації предметів, явищ, процесів, сигналів, ситуацій і об'єктів, які характеризуються скінченним набором властивостей і ознак [7]. Розпізнавання образів включається в ширшу наукову дисципліну – теорію машинного навчання, метою якої є розробка методів побудови алгоритмів, що здатні навчатися. Прикладом задачі, що потребує такий підхід, може служити система безпілотного керування автомобілем від компанії Tesla. Ця система повинна зчитувати велику кількість зовнішньої інформації та приймати миттєве рішення: колір світла світлофору на перехресті, дорожні знаки, інші машини, стан дорожнього покриття, погодні явища, стан автомобілю.

Мозок людини здатен оброблювати велику кількість вхідної зовнішньої інформації, не використовуючи для цього велику кількість ресурсів, шляхом

зведення їх до чогось подібного. Тобто наш мозок автоматично проводить класифікації явищ навколишнього світу, на групу схожих явищ.

Приклад: яку літеру алфавіту означає наведена множина символів {a, A, **a**, ***A***, ***a***}?

Розв'язок: зі свого досвіду читання людина знає, що наведені символи позначають літеру «а».

Більш складним прикладом може слугувати фігури на рисунку 1.1. Зрозуміло, що людина буде об'єднувати фігури за кількістю ліній, їх напрямом або їх геометричним положенням. З даних прикладів випливає, що людина об'єднує об'єкти і явища зовнішнього світу в групи за деякими спільними властивостями і присвоює цим групам однакове поняття [8].

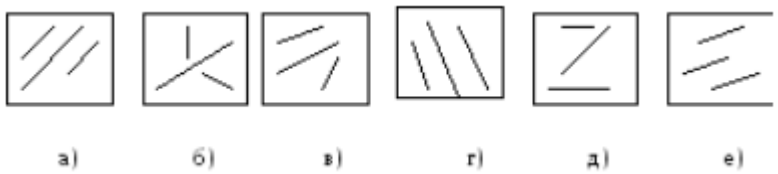

Рисунок 1.1 – Приклади фігур

Задача розпізнавання образів – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних. При постановці задач розпізнавання намагаються користуватися математичною мовою, тому використовують спеціальні визначення: образ, ознака, вектор-реалізації, вирішальне правило [8].

Образ – модель, яка відбиває властивості об'єкта, що розпізнається, на рисунку 1.1 образом є сама фігура.

Ознака – характеристика певної властивості об'єкта, що аналізується, на рисунку 1.1 ознакою є наявність ліній, геометрична форма фігури в якій вони знаходяться.

Вектор-реалізації – структурована послідовність ознак розпізнавання, яка подається у вигляді вектор-рядка або вектор-стовпця. Наприклад обличчя людини може бути перетворене на вектор ознак, де кожна координата відповідає певним

характеристикам обличчя (відстані між очима, форма носа, лінія губ і т.д.). Алгоритми розпізнавання обличчя, такі як FaceNet або Eigenfaces, використовують цей підхід для порівняння обличчя за вектором ознак.

Вирішальне правило – це алгоритм або математична функція, яка визначає, як приймати рішення в задачах класифікації або розпізнавання образів на основі певних вхідних даних. В контексті розпізнавання образів та машинного навчання, вирішальне правило використовується для того, щоб на основі значень ознак або характеристик образу визначити, до якого класу або категорії належить цей образ [7].

Загалом цей розділ кібернетики спеціалізується не тільки на розпізнаванні зображень, а взагалі на будь-якому масиві інформації який потрібно відфільтрувати так проаналізувати.

1.2 Система розпізнавання образів

Початок комп’ютерної ери вивів теорії розпізнавання образів на новий рівень, забезпечивши стрімкий розвиток практичних технологій на базі вже існуючих теоретичних відомостей. Теорія стала системою.

Система розпізнавання образів – це комплекс апаратних і програмних засобів, призначених для ідентифікації або класифікації об'єктів (образів) на основі їх характеристик, здатний моделювати розумові процеси, властиві під час прийняття рішень із метою виявлення аналогій серед досліджуваних об’єктів [7].

Система має велику кількість методів роботи із даними, тому для більшої зручності вони класифікуються за спільними ознаками. Самі системи розпізнавання поділяються на складні та прості. Прості характеризуються єдиною фізичною природою ознак. Наприклад тільки геометричні розміри для ідентифікації об’єкту на екрані, або тільки колір певний звук. Складні системи мають неоднорідні фізичні ознаки – автопілот в автомобілі.

Наступним кроком класифікації є поділення на системи без навчання, з навчання та з самонавчанням (рисунок 1.2). Прості системи загалом не навчається

та будуються за допомогою структурних (лінгвістичних) методів – деревоподібні структури, системи на основі графів та формальних мов. Але прості системи можуть бути частиною складних, тобто комбінованими [3].

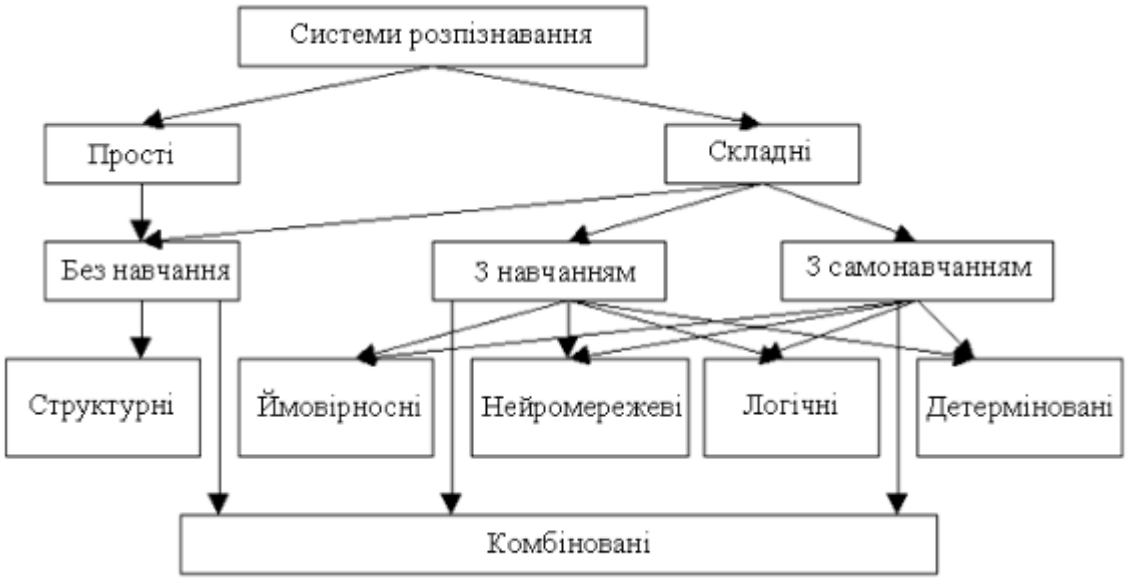

Рисунок 1.2 – Класифікація систем розпізнавання

У складних системах з навчанням (учителем) на етапі навчання з метою побудови вирішальних правил здійснюється розбиття простору ознак на класи розпізнавання. У цих системах первинної інформації достатньо для того, щоб визначити алфавіт класів і побудувати словник ознак, але недостатньо для опису класів на мові ознак. Початкова інформація, дозволяє виділити конкретні об'єкти, що належать різним класам. Метою процесу навчання є визначення вирішального правила шляхом багаторазового пред'явлення системі розпізнавання різних об'єктів із зазначенням їхніх класів. Навчальні системи розпізнавання, що перебувають на етапі формування, працюють під керівництвом "вчителя". "Вчитель" кілька разів показує системі об'єкти з усіх визначених класів і вказує, до якого класу належить кожен з них. Після цього "вчитель" починає перевіряти систему розпізнавання, коригуючи її відповіді, поки кількість помилок не знизиться до бажаного рівня.

На рисунку 1.3 показана загальна схема цього методу навчання. На систему подається масив невідомих об'єктів (W), вони проходять через технічні засоби (ТЗ),

що є вимірювачами ознак розпізнавання, далі алгоритм побудови розподільчих функцій (АРФ), до алгоритму класифікації (А), звідти до системи автоматичного управління розпізнаванням (САУ), на цьому блоці вихідні данні через вчителя (У) порівнюються з навчальними об'єктами (НО). У разі незадовільного результату цикл повторюється. Якщо результат розпізнавання задовільний, результати записуються у опис класів об'єктів розпізнавання (АО). У випадку коли ще на етапі ТЗ вхідні об'єкти співпадають із НО, результати одразу записуються у АО.

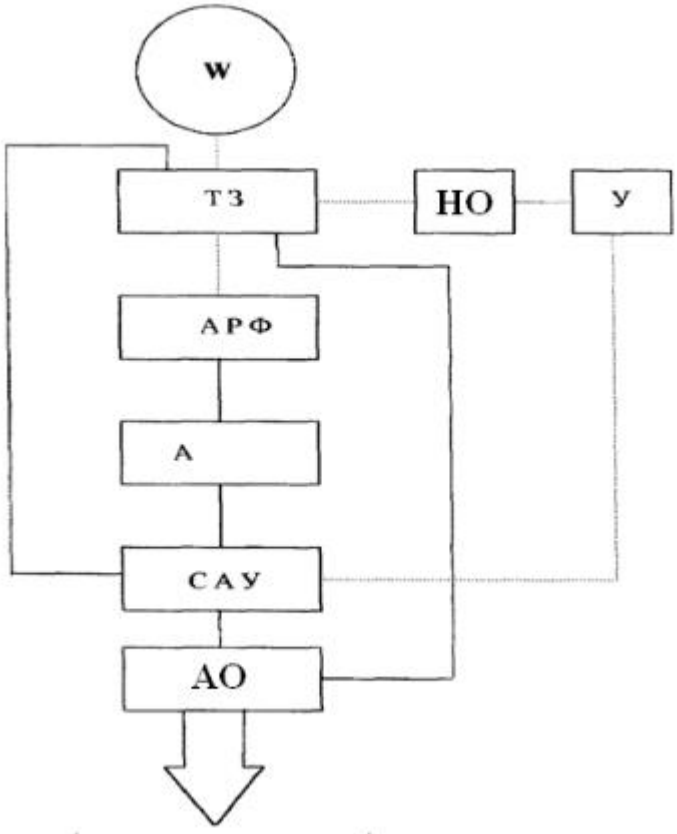

Рисунок 1.3 – Схема навчання з вчителем

У системах з самонавчанням початкової інформації достатньо для визначення словника ознак, але її недостатньо для того, щоб класифікувати об'єкти. На етапі формування системі подають набір об'єктів з визначеними ознаками, проте, через обмежену кількість первинної інформації, система не отримує інструкцій щодо їх належності до класів. Замість цього використовуються правила,

за якими система самостійно під час самонавчання створює класифікацію, яка може відрізнятися від природної.

На відміну від систем без навчання та тих, що навчаються з учителем, системам, що самонавчаються, характерна недостатність інформації не лише для формування описів класів, але й для визначення алфавіту класів. Тобто, відомі лише ознаки розпізнавання. Проте процес навчання організований на основі певного набору правил, відповідно до яких система самостійно виконує класифікацію. Схему такої системи наведено на рисунку 1.4

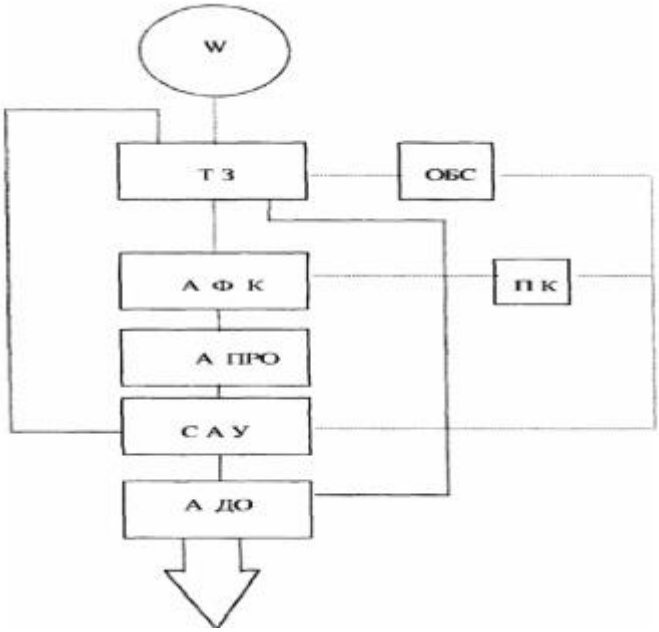

Рисунок 1.4 – Система з самонавчанням

Якщо класифікація базується на особливостях інформації про ознаки розпізнаваних об'єктів, які можуть бути детермінованими, ймовірнісними, логічними або структурними, то в залежності від того, якою мовою ознак описуються ці об'єкти і який алгоритм розпізнавання використовується, системи розпізнавання можна поділити на детерміновані, ймовірнісні, логічні, структурні та комбіновані.

У детермінованих системах для побудови алгоритмів розпізнавання використовуються "геометричні" методи, які базуються на вимірюванні відстаней між об'єктом і еталонами класів. Це передбачає, що в розпізнавальному просторі

мають бути відомі координати еталонів або об'єктів, що належать до відповідних класів.

У ймовірнісних системах застосовуються методи, які базуються на теорії статистичних рішень. Це означає, що алгоритми розпізнавання будуються з урахуванням ймовірнісних зв’язків між ознаками об'єктів та класами, до яких вони належать.

У логічних методи розпізнавання, що спираються на дискретний аналіз і числення висловлень. Ці методи передбачають формування логічних зв'язків у вигляді булевих рівнянь, де змінні – це ознаки об'єктів, а невідомі – класи, до яких ці об'єкти належать.

Структурні системи розпізнавання вимагають використання спеціальних граматик, що породжують мови, де кожне речення описує об'єкти конкретного класу. Застосування цих методів передбачає, що множина речень повинна бути розбита на підмножини, кожна з яких відповідає певному класу, а елементи цих підмножин описують об'єкти, які належать до відповідного класу.

Нейромережеві системи базуються на використанні моделей і методів обчислювальних структур, які в певному сенсі схожі на біологічні нейронні мережі. Їх основною перевагою є висока адаптивність і здатність до апроксимації. У кожному завданні розпізнавання початкові дані є результатом певних спостережень або вимірювань, які називають первинними ознаками, а їх сукупність – вхідним сигналом. Результатом одиничного акту розпізнавання є прийняте рішення, а для вирішення загального завдання розпізнавання формується вирішальне правило або алгоритм, який відображає множину сигналів на множину рішень. Якщо рішення можна поділити на кілька варіантів, то розпізнавання можна розглядати як класифікацію, при цьому вирішальна функція розподіляє множину сигналів на підмножини, що відповідають певним класам. У випадках, коли сигнали можуть бути описані в термінах близькості, класи відокремлюються розділяючими поверхнями, такими як гіперплощини.

У багатьох ситуаціях існує об'єктивна класифікація сигналів, яка може бути відома за наявності додаткової інформації. Проте, іноді об'єктивної класифікації

немає. Об'єктивну класифікацію можна описати через певний шуканий дискретний параметр. Тоді сигнал залежить від шуканого параметра. У загальному випадку, може бути кілька таких параметрів, які можуть бути як дискретними, так і безперервними.

1.3 Фільтрація образів

Неточності та шуми, що виникають під час вимірювання ознак для розпізнавання, зумовлюють необхідність попередньої обробки даних. Це включає фільтрацію для усунення шумів і відновлення правильних значень ознак. Сегментація є окремою групою методів попередньої обробки, яка полягає у відокремленні розпізнаваних об'єктів від фону. Наприклад, у системах відеоспостереження для автоматичної ідентифікації особи попередня обробка зображень з динамічним фоном передбачає пошук і виділення обличчя або його частин.

Фільтрація – математичні методи, що дозволяють виділити на зображенні цікаву для нас область без аналізу усього зображення, це дозволить уникнути обробки зайвого масиву даних.

1.3.1 Фільтр бінаризації по порогу

Бінаризація по порогу [9] – це один з найпростіших та найпоширеніших методів фільтрації зображень, який використовується для сегментації зображення. Основна ідея методу полягає в тому, що всі пікселі зображення діляться на дві категорії — ті, які перевищують певний поріг, і ті, які не перевищують. Пікселі, що перевищують поріг, позначаються як білі, або 1, а ті, що не перевищують, як чорні, або 0. Цей процес дозволяє перевести зображення у бінарне, що спрощує подальший аналіз зображення (рисунок 1.5).

Окрім поділення пікселів на чорні та білі, можна використовувати метод гістограм. Гістограма показує розподіл пікселів за інтенсивністю від 0 до 255. Використовуючи гістограму зображення, можна виділити найбільш релевантні для

нас області інтенсивності пікселів. Таким чином, на зображенні можна виділити лише певний цікавий для нас об’єкт. Ми просто задаємо потрібну область інтенсивності, відрізаючи інші об’єкти. Наприклад деякі області зображення можуть відповідати фону, а інші об'єктам. В такому разі ми обираємо проміжок інтенсивності пікселів, що відповідають фону, та відрізаємо його (рисунок 1.6).

Рисунок 1.5 – Бінаризація по порогу

Рисунок 1.6 – Біноризація за гістограмою в HSV

1.3.2 Фільтр Гаусса та Габора

Класична фільтрація включає методи, які використовуються для видалення або збереження певних частот в сигналі. Основними є ФНЧ Гаусса, ФВЧ Габора та фільтр Фур’є.

Фільтр Фур'є перетворює сигнал з часової області у частотну. Це дозволяє аналізувати та обробляти частоти, присутні у сигналі. Фільтр виділяє певні частоти або прибрати небажані частотні компоненти, такі як шум.

ФНЧ пропускає частоти, які є нижчими за певний встановлений поріг, і пригнічує всі частоти, що вище. Використовується для зменшення шуму або згладжування сигналів. ФВЧ ж виконує зворотну функцію: пропускає високі частоти та пригнічує низькі. Цей тип фільтрації використовується для виділення деталей або контурів.

Найпростішим прикладом фільтрів, що реалізують підкреслення низьких частот та високих частот є фільтр Гаусса [10] та Фільтр Габора [11]. Для кожної точки зображення вибирається вікно та перемножується з фільтром того самого розміру. Результатом такої згортки є нове значення точки. На рисунку 1.7 зображено результат фільтрації методом Габора. Найтемніші пікселі вхідного зображення були посилені.

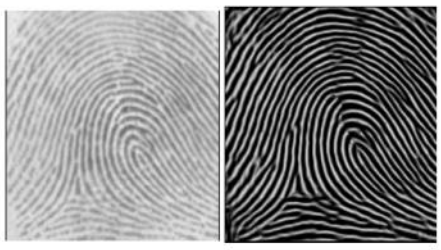

Рисунок 1.7 – Обробка зображення фільтром Габора

На рисунку 1.8 зображено результат фільтрації Гаусом. Гауса використовують для зниження кількості шумів але при ресемплінгу він дає сильне розмиття зображення, що і видно на прикладі.

Рисунок 1.8 – Фільтрація Гаусом

1.3.3 Фільтрація вейвлетами

Фільтрація вейвлетами [12] – це метод обробки сигналів і зображень, що базується на використанні вейвлет-перетворень для аналізу частотного вмісту сигналу або зображення на різних масштабах. Вейвлети – це функції, які використовуються для декомпозиції сигналу на різні частоти і масштаби. На відміну від традиційного перетворення Фур'є, яке дає тільки частотну інформацію, вейвлет-перетворення інформує про зміни частот у часі. У якості вейвлетів найчастіше використовують трьох-мірний Хаара (рисунок 1.9, а), дво-мірний Мейера, вейвлет Мексиканська шляпа (рисунок 1.9, б), вейвлет Добеши (рисунок 1.9, в), вейвлет Морле (рисунок 1.9, г).

Вейвлет фільтрації можна використовують у обробці медичних рентгенівських знімків, видаляючи шум та інші артефакти, не впливаючи на чіткість контурів органів або кісток. У системах ідентифікації особи, вейвлети використовують для пошуку відблиску у оці людині, перевіряючи таким чином, чи реальна людина стоїть перед камерою (рисунок 1.10).

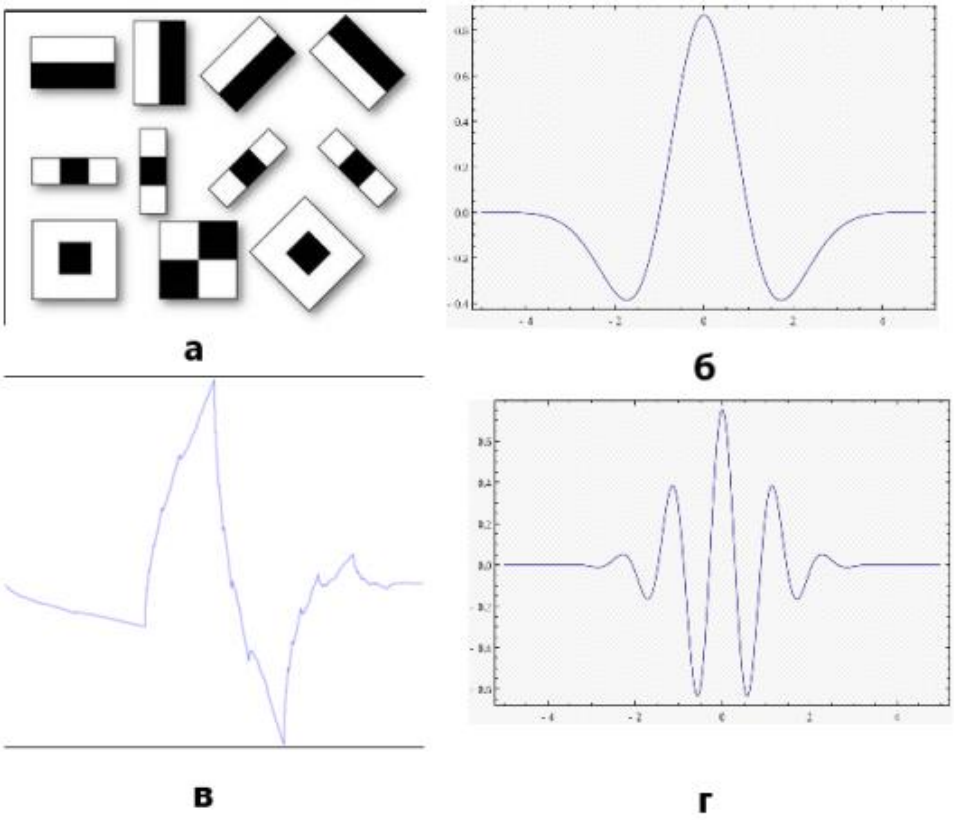

Рисунок 1.9 – Класичні вейвлети

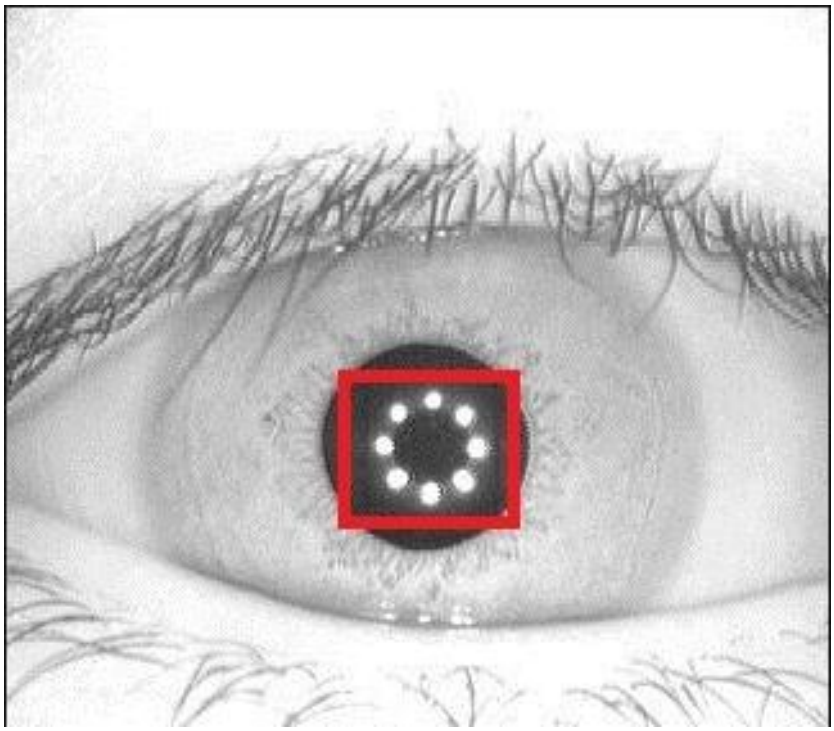

Рисунок 1.10 – Використання вейвлетів для пошуку відблиску

1.3.4 Оператор пошуку контуру Кенні

Окремий клас фільтрів – оператори контурів. Контури дуже корисні, коли ми хочемо перейти від роботи із зображенням до роботи з об'єктами на цьому зображенні. Коли об'єкт досить складний, але добре виділяється, часто єдиним способом роботи з ним є виділення його контурів. Існує ціла низка алгоритмів, що вирішують завдання фільтрації контурів: оператор Кенні, оператор Собеля, оператор Прюітг, оператор Робертса, оператор Лапласа.

Оператор Кенні [13] – це класичний алгоритм для виявлення країв на зображеннях, розроблений Джоном Кенні у 1986 році. Для його роботи спочатку проводиться згладжування зображення за допомогою Гауссового фільтра, щоб зменшити рівень шуму (рисунок 1.11, б). Для визначення місць з різкою зміною інтенсивності пікселів, обчислюється градієнт зображення, використовуючи похідні за напрямками X та Y, визначаючи напрям і величину градієнта. Чим більший градієнт, тим ймовірніше, що це край.

Далі приглушуються всі точки, які не є локальними максимумами градієнта, залишаючи тільки тонкі та чіткі краї. Після встановлюються два пороги – високий та низький поріг. Пікселі з інтенсивністю вище високого порогу вважаються сильними краями. Пікселі між низьким і високим порогами – слабкими краями. Пікселі з інтенсивністю менше низького порогу вважаються фоном. Для побудови фінального контуру, слабкі та сильні краї з’єднуються, але тільки за умови якщо слабкий край межує зі сильним. В такому разі він вважається частиною реального краю, інакше він ігнорується. Це називається гістерезис (рисунок 1.11, в).

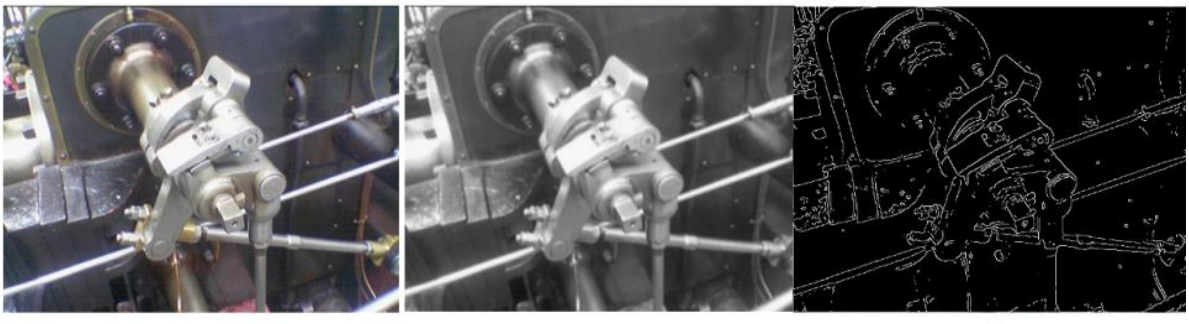

Рисунок 1.11 – Робота оператора Кенні

1.4 Методи розпізнавання обличчя

Розпізнавання обличчя є одним із найпоширеніших методів біометричної автентифікації, що використовує можливості розпізнавання образів і аналізу зображень. Основними завданнями системи розпізнавання обличчя є верифікація та ідентифікація. Верифікація – це перевірка один-до-одного, коли порівнюється обличчя з еталонним зображенням, а ідентифікація – це порівняння один-до-багатьох, де зображення обличчя звіряється з усіма шаблонами у базі даних.

Відбитки пальців, райдужна оболонка ока, унікальні риси обличчя, голос, хода, підпис та інші ознаки можуть бути використані для біометричної ідентифікації людини. На відміну від інших підходів, розпізнавання обличчя не вимагає активної участі людини, що робить його дешевшим за інші види біометрії. Точність ідентифікації обличчя часто може перевищувати 90% у величезних масивах даних з контрольованими позами та освітленням.

1.4.1 Геометричний метод розпізнавання

Геометричний метод розпізнавання базується на аналізі унікальних рис обличчя: відстаней між певними точками очей, носа, рота і форми обличчя. Цей метод використовує метод ключових точок для створення математичної моделі обличчя, яку потім можна порівнювати з іншими моделями [14].

Спочатку система визначає форму обличчя та ключові риси. Потім шляхом вимірювання відстаней між цими точками створюється окремий набір параметрів. Схожі параметри знаходять, порівнюючи цей набір з параметрами інших зразків у базі даних. В основі методу лежить ідея, що пропорції та відстані між окремими рисами обличчя кожної людини є достатньо узгодженими, щоб їх можна було використовувати як основу для ідентифікації. Кожне обличчя має унікальний вектор ознак, що складається з ключових точок і математичних співвідношень між ними. На рисунку 1.12 показаний приклад побудови геометричних ліній між ключовими точками обличчя.

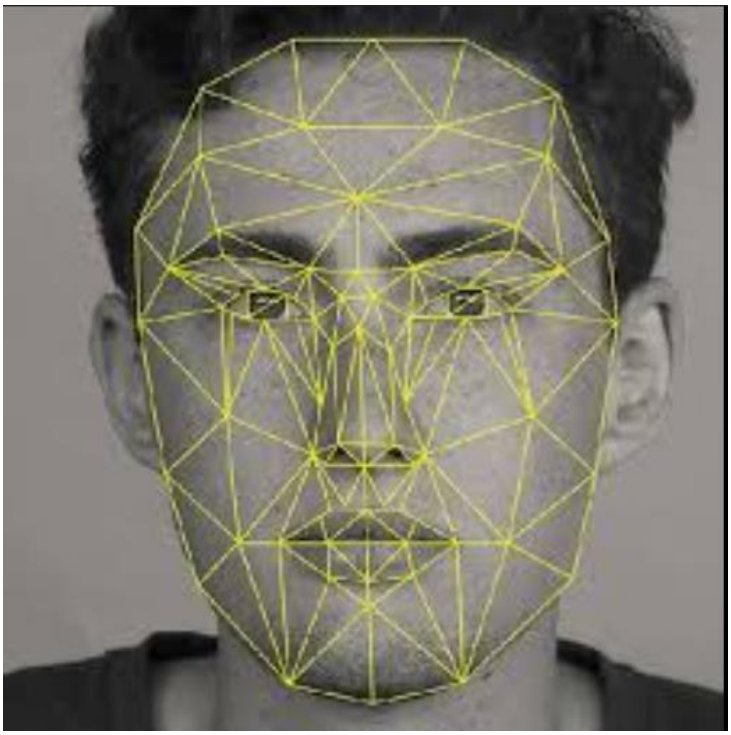

Рисунок 1.12 – Приклад побудови геометричних ліній

1.4.2 Метод головних компонентів

Метод головних компонентів (PCA) – статистичний метод для зменшення розмірності даних, що широко застосовуваний у задачах розпізнавання образів. Основна ідея методу полягає в тому, щоб представити об'єкт в новому просторі з меншою кількістю змінних, які містять максимальну кількість інформації про вихідний об'єкт [14, 15, 16].

Спочатку дані центруються, щоб кожна змінна мала середнє нульове значення. Далі обчислюється коваріаційна матриця, яка показує, як змінні змінюються одна відносно одної. Власні вектори та власні значення коваріаційної матриці обчислюються для визначення головних компонентів. Власні вектори, які відповідають найбільшим власним значенням, утворюють нову базу. Ті вектори, які пояснюють більшість варіацій у даних, і є головними компонентами. Дані проектуються на новий простір, створений головними компонентами, що дозволяє зменшити їх розмірність, залишаючи при цьому максимально можливу кількість інформації (рисунок 1.13).

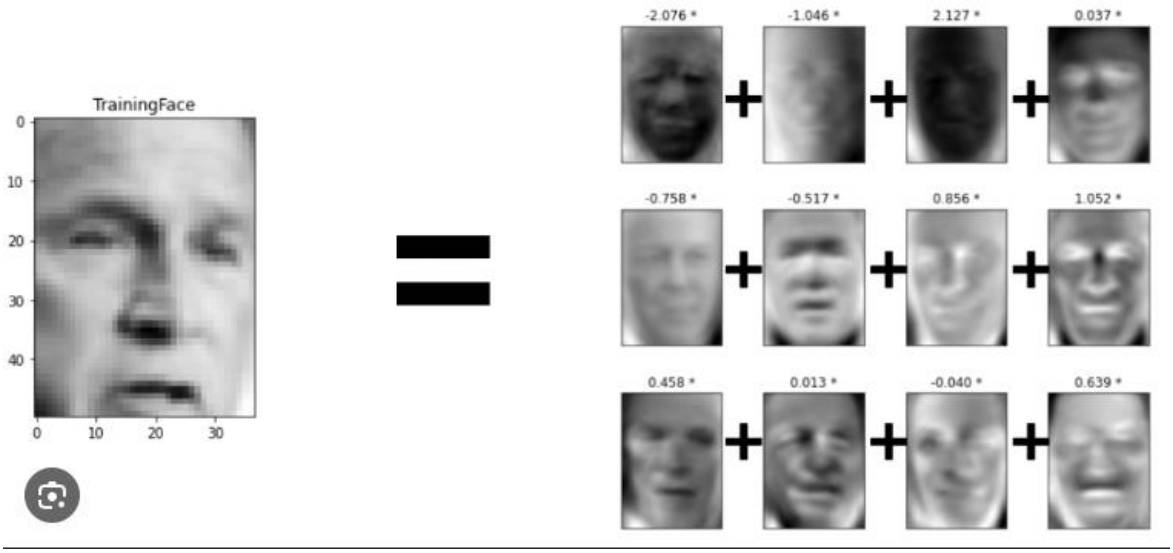

Рисунок 1.13 – Метод головних компонентів

У PCA кожне обличчя представляється як вектор, або як одна довга послідовність з пікселів. Зображення обличчя перетворюється в лінійну комбінацію декількох головних компонентів, що дозволяє ефективно відділити одне обличчя від іншого. Ця техніка зберігає важливу інформацію, істотно скорочуючи кількість даних, що використовуються для обробки. Точність розпізнавання покращується завдяки здатності PCA відфільтровувати шум або незначні коливання. Коли між змінними в даних є лінійні кореляції, PCA працює ефективно.

1.4.3 Використання нейронних мереж у розпізнаванні обличчя

Нейронні мережі, зокрема згорткові нейронні мережі (CNN) та глибокі згорткові нейронні мережі (Deep CNNs), є революційними інструментами для розпізнавання і обробки зображень, завдяки своїй здатності самостійно виділяти важливі ознаки [14, 16, 17].

CNN – було розроблено для вирішення завдань обробки зображень, оскільки вони дозволяють автоматично виділяти важливі елементи, такі як лінії, форми й контури, що значно спрощує і покращує процес розпізнавання. CNN мережа складається с трьох типів шарів: перший шар вхідний, другий шар прихований і третій вихідний шар. Вхідний шар є шаром згортки, в середині якого захований

фільтр, або як його називають ядро, що представляє собою матрицю чисел N*N, ця матриця перемножується із матрицею такого ж розміру, але взяту з вхідного зображення. Далі усі значення нової матриці смутуються і ми отримуємо згорткові признаки, які передаються на інший шар. Тобто матриця з вхідного зображення, яка складається з дев'яти пікселів на вході першого шару, представляється у значенні одного пікселя. Таких операцій згортання на першому шару може бути декілька (рисунок 1.14). У випадку мереж з великою кількістю згорткових шарів, перші шари призначені для виділення загальних ознак.

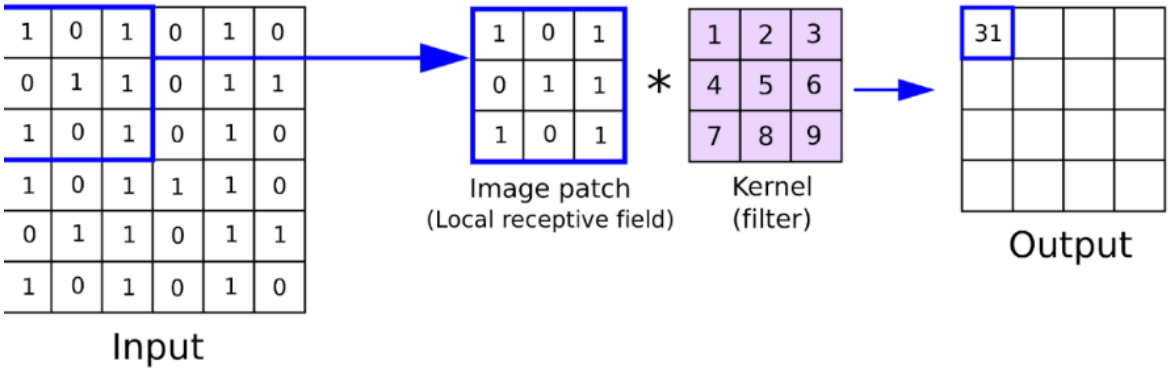

Рисунок 1.14 – Згортковий шар

Після згорткового шару данні потрапляють до прихованого шару, або шару об'єднання. Його головною задачею є зменшення розмірності зображення, для того аби зменшити кількість навчальних параметрів, що знижує обчислювальні витрати. У цьому шарі нейрон отримує матрицю значень з вхідного шару, додає значення зміщення та застосовує порогову функцію, яка називається функцією активації, котра визначає чи активується конкретний нейрон чи ні. Активований нейрон передає дані нейронам наступного шару по каналам зв'язку (рисунок 1.15).

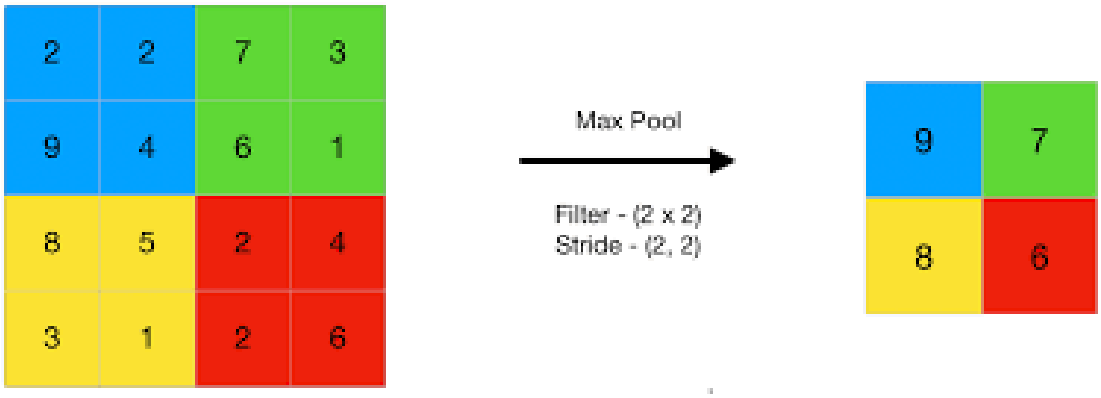

Рисунок 1.15 – Шар об’єднання

На завершальному етапі CNN проводить пряме поширення. Дані обробляються через повнозв’язні шари, де кожен нейрон з’єднаний з усіма нейронами попереднього шару, об’єднуючи таким чином інформацію і створюючи узагальнену картину. Згенерований висновок порівнюється з фактичним висновком для генерації помилок. Ця помилка повертається назад для оновлення фільтрів і значень зміщення.

CNN можуть виділяти ознаки без втручання людини, що робить їх дуже ефективними. Обробляючи локальні області зображення, зменшують обсяг оброблюваної інформації, дозволяючи мережі працювати швидше.

Deep CNNs є розширенням звичайних CNN і мають більшу кількість згорткових шарів, що дозволяє їм виділяти складніші й точніші патерни зображення. Глибокі CNN складаються з десятків і навіть сотень шарів. На перших рівнях мережі виділяються базові ознаки, такі як контури й кути. Глибші шари розглядають більш комплексні форми й патерни.

Найпопулярнішими архітектурами глибоких CNN є ResNet, Inception та VGG, які дозволяють будувати надзвичайно складні моделі, що справляються з великою кількістю ускладнень.

Загальне схема CNN мережі показана на рисунку 1.16.

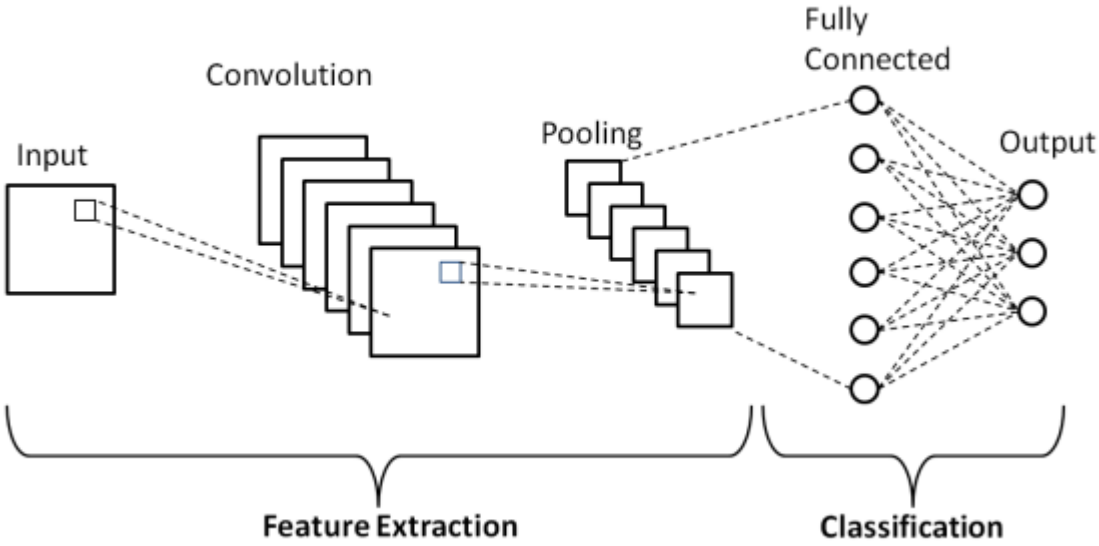

Рисунок 1.16 – Схема методу CNN

1.5 Висновки до першого розділу

У теоретичній частині був проведений аналіз основних понять теорії розпізнавання образів. Було надано визначення системі розпізнавання образів, та надано класифікацію. Було визначено, що системи розпізнавання, в загальному, поділяються на прості та сскладін. Прості характеризуються единою фізичною природою ознак. Складні ж системи мають неоднорідні фізичні ознаки. Прості системи можуть бути лише без навчання і будуються за допомогою лінгвістичних методів. Складні ж системи навчаються або з вчителем, або самостійно. Були розглянуті схеми принципу роботи таких систем.

Був проведений аналіз принципу роботи сучасних методів фільтрації зображення, зокрема фільтрів ФНЧ та ФВЧ. Описані вейвлети, як один з основних методів ідентифікації реальності людини. Детально розібран принцип роботи CNN мереж.

2 РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ

2.1 Опис вимог до системи розумного доступу

У ході розробки системи розумного доступу до виробничого приміщення, необхідно розробити декілька функціональних програм, які використовують комп'ютерний зір та пропонують різні методи розпізнавання обличчя людини, з метою знаходження найбільш оптимального варіанту. Вони повинні відповідати наступним функціональним вимогам:

– програма повинна мати функціонал пошуку обличчя на зображенні/камері;

– програма повинна мати функцію розпізнавання обличчя;

– програма повинна виводити інформацію про результат зчитування даних обличчя;

– у програмі повинна бути реалізована база даних обличь;

– програма повинна мати функціонал додавання/видалення обличь до бази даних;

– програма повинна дозволяти користувачу використовувати усі функції без реєстрації;

– код програми повинен мати можливість працювати з будь-яким апаратним рішенням з додавання мінімальних змін.

У результаті аналізу функціональних вимог та огляду пропозицій конкурентів, сформовано наступні не функціональні вимогу:

– програма повинна бути написана на мові програмування Python;

– програма повинна працювати на будь-якій версії Windows;

– програма повинна працювати без необхідності постійного з'єднання з мережею інтернет.

2.2 Загальна схема принципу роботи та апаратні рішення

На рисунку 2.1 зображена схема принципу роботи, що розроблена на основі функціональних вимог, за якою повинна працювати майбутня програма. Згідно схеми співробітник виробництва буде підходити до терміналу з камерою. Вбудована IP-камера буде передавати зображення або відео на сервер для подальшої обробки. З метою підвищення безпеки необхідно використовувати RTSP-протокол. Програма знаходить на зображенні обличчя та порівнює його з наявною інформацією у базі даних. Якщо обличчя ідентифіковане, програма надсилає сигнал на відкриття дверей. Для співробітника цей процес є швидким та автоматичним. Якщо ж, обличчя не належить співробітнику виробництва, або належить співробітнику з недостатнім рівнем допуску до приміщення, програма надсилає запит до адміністратора, аби той прийняв рішення в ручному режимі.

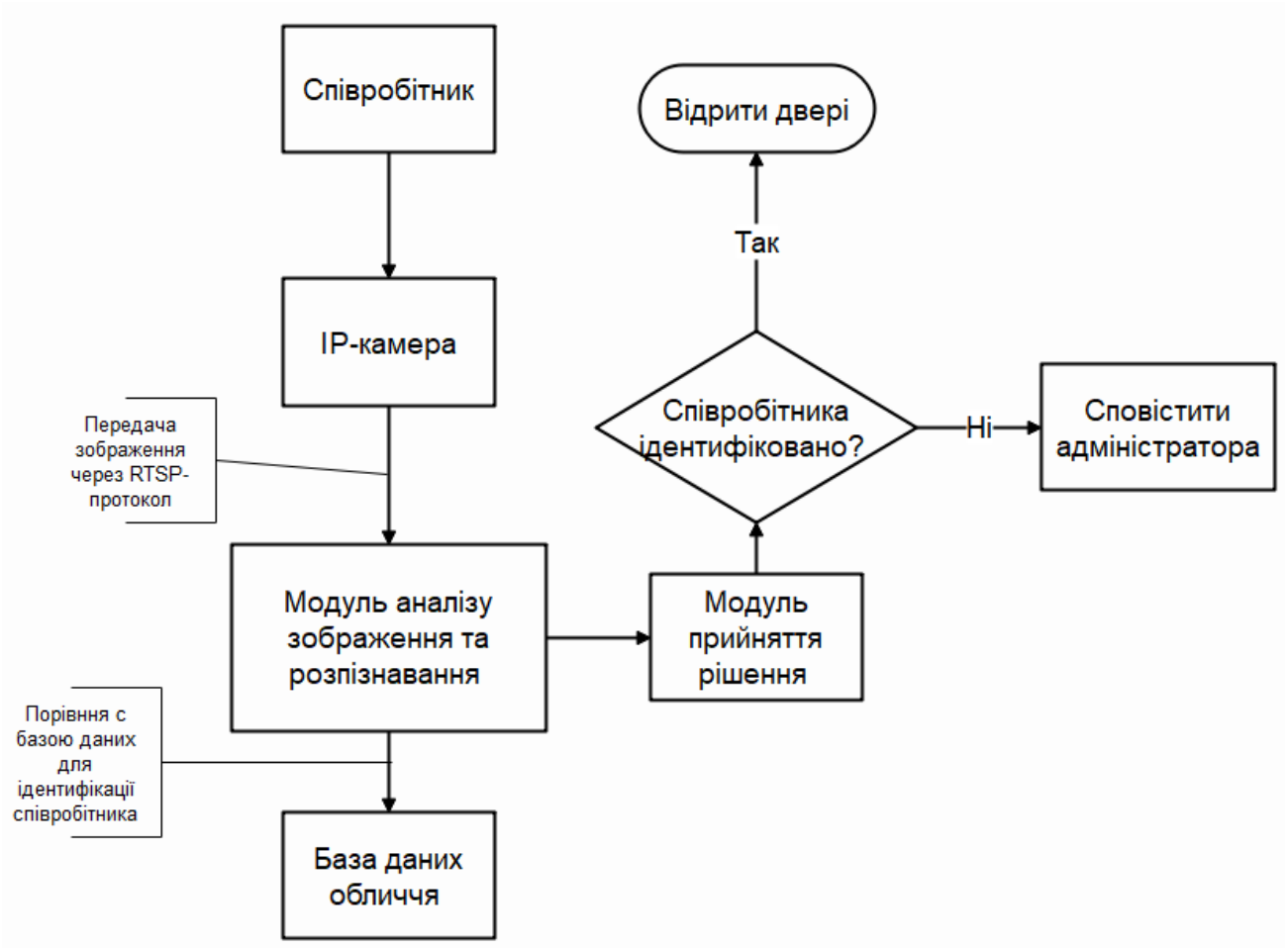

Рисунок 2.1 – Загальна схема принципу роботи програми

Розглянемо можливі апаратні рішення. Загально, реалізацію системи розпізнавання можна поділити на два етапи – етап проектування та етап промислової реалізації.

На етапі проектування нашою метою є створення прототипу, на якому замовнику буде продемонстровано ключові особливості створюваної системи, та буде продемонстрований базовий функціонал. Так як замовник може вносити велику кількість правок та побажань, через що система може кардинально змінюватися, необхідна якомога гнучка та бюджетна апаратна частина, для цього етапу.

Базовим варіантом буде використання Raspberry Pi з модулем камери. Raspberry можна використовувати як сервер з базою даних, сховище зображень або записів відео з співробітниками за останній період. Також перевагою буде можливість підключення інших не Raspberry модулів через мережевий MQTT-протокол, для обміну даними. Одним з таких пристроїв може бути ESP32-CAM. Це модуль на базі мікроконтролера ESP32, який поєднує в собі потужні обчислювальні можливості та вбудовану камеру. Розроблений для програм, пов'язаних із відео- та фотозйомкою, а також передачі даних через Wi-Fi або Bluetooth, це компактне та доступне рішення для створення проектів на основі комп'ютерного зору та Інтернету речей (IoT). Raspberry Pi може виступати в ролі серверу, що приймає та оброблює інформацію одразу від декількох ESP.

Сервером, що приймає інформацію від ESP32, може будь-який стаціонарний комп’ютер, не обов’язково Raspberry Pi, адже базовим функціоналом ESP32 є створення WebServer клієнту з трансляцією зображення, до якого може підключитися будь-який пристрій, через Wi-Fi. Також сама ESP дозволяє створювати невеличкі проекти с використанням елементів Arduino.

На етапі промислової реалізації, можна використовувати готові IP-камери, такі як Hikvision, Dahua, або Xiaomi. Вони мають низку переваг та особливостей, особливо для промислових та комерційних додатків. Готові IP-камери спроектовані для цілодобової роботи. Вони стійкі до перегріву, зовнішніх факторів, багато камер мають захист за стандартом IP65/IP67 та мають сертифікати для промислового

використання. На відміну від бюджетних DIY-рішень, вони перевірені на стабільність виробником, що знижує ймовірність збоїв. Підтримують роздільну здатність від Full HD до 4K, що важливо для точного розпізнавання обличчя або об'єктів. Деякі моделі мають реалізацію нічного режиму, інфрачервоне підсвічування забезпечує чітке зображення в умовах слабкого освітлення або повної темряви.

2.3 Вибір мови програмування та середовища розробки

Робота над проектом передбачає використання бібліотеки OpenCV – це бібліотека з відкритим вихідним кодом для задач комп'ютерного зору і машинного навчання. Її розробила 2000 року компанія Intel і відтоді вона стала однією з найпопулярніших бібліотек у цій галузі. OpenCV надає інструменти для аналізу зображень, опрацювання відео та розроблення додатків, пов'язаних зі штучним інтелектом і комп'ютерним зором. OpenCV підтримує безліч мов програмування. Основною э C++ чиє використання гарантує високу продуктивність та зручний API. Але C++ потребує розвинених навиків оптимізації коду, не є варіантом для швидкого створення готового продукту і потребує доволі потужної системи.

Мова Python підходить для швидкого прототипування та експериментів завдяки лаконічному синтаксису. Ідеально поєднується з бібліотеками машинного навчання TensorFlow та PyTorch. OpenCV надає інтерфейс через бібліотеку cv2. Python – це високорівнева мова програмування загального призначення. Її філософія проектування підкреслює читабельність коду з використанням значних відступів. Python динамічно перевіряє типи та збирає сміття. Підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об'єктно-орієнтоване та функціональне програмування.

Для роботи на мові Python необхідно обрати середовище розробки. IDE по типу PyCharm і VisualStudio гарно підходять для написання великих проектів. Серед інтерактивних середовищ можна виділити Python REPL, що доступна через команду python або python3 у терміналі – зручно для тестування окремих рядків

коду або невеликих скриптів. Jupyter Notebook дозволяє писати Python-код і отримувати візуальний висновок у вигляді графіків, таблиці тощо. Для написання проекту буде використовуватися IDE PyCharm (рисунок 2.2) [18].

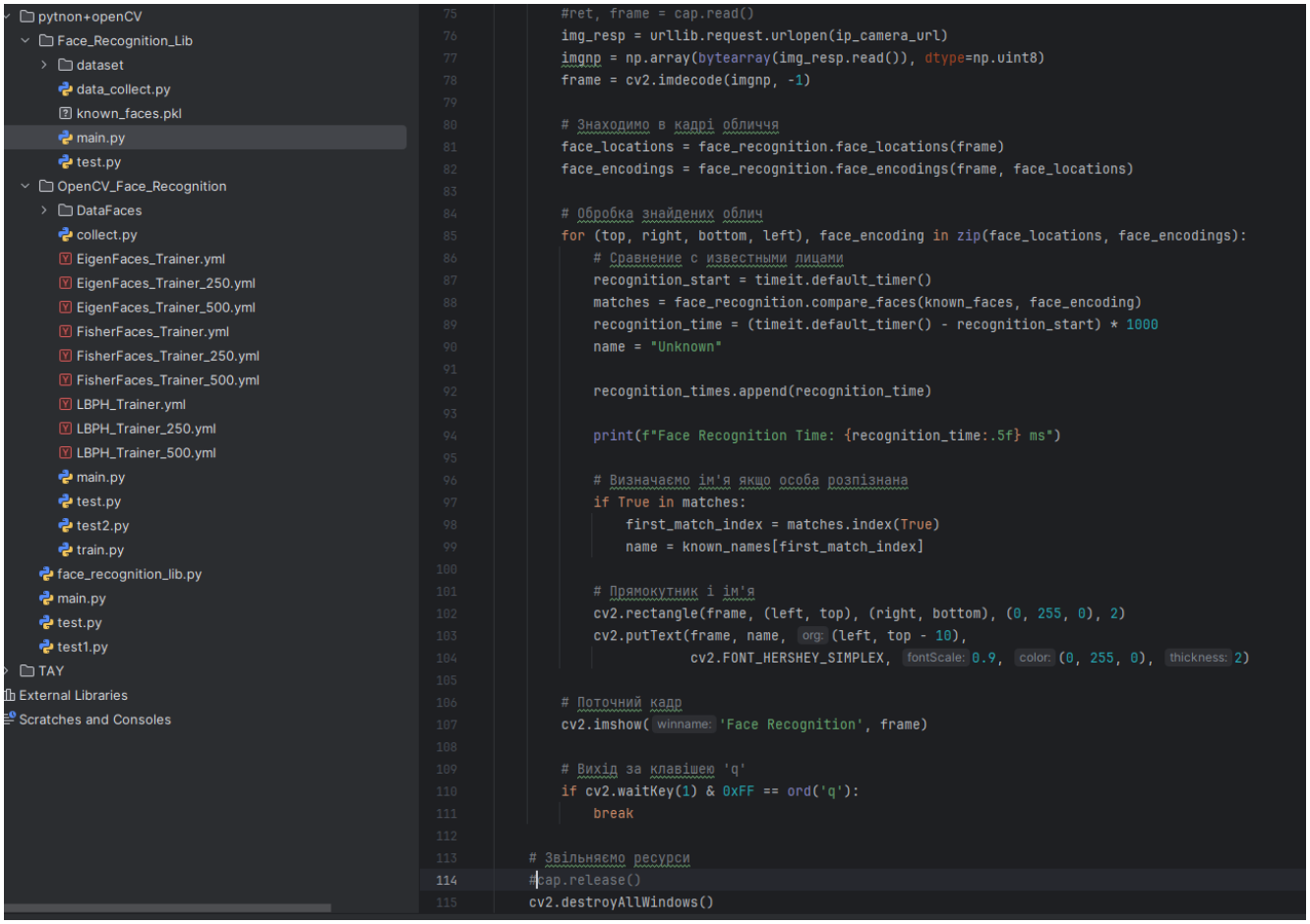

Рисунок 2.2 – Вид IDE PyCharm на прикладі коду Python

Для ESP32-CAM також необхідне середовище розробки. VSCode надає можливість встановити розширення для написання та компіляції коду у ESP32, також можна використовувати MicroPython. Другий варіант, використання власної IDE. Загалом було розроблено дві IDE – Arduino Legacy IDE та Arduino IDE.

Для роботи буде використовуватися Arduino IDE. Початкова альфа-версія нової IDE Arduino була випущена 18 жовтня 2019 року під назвою Arduino Pro IDE. Бета-версія вийшла 1 березня 2021 року, перейменована на IDE 2.0. 14 вересня 2022 року Arduino IDE 2.0 була офіційно випущена як стабільна версія. Система все ще використовує Arduino CLI, але покращення включають більш професійне

середовище розробки та підтримку автозавершення. Однією з важливих функцій Arduino IDE 2.0 є можливість налагодження (рисунок 2.3) [19].

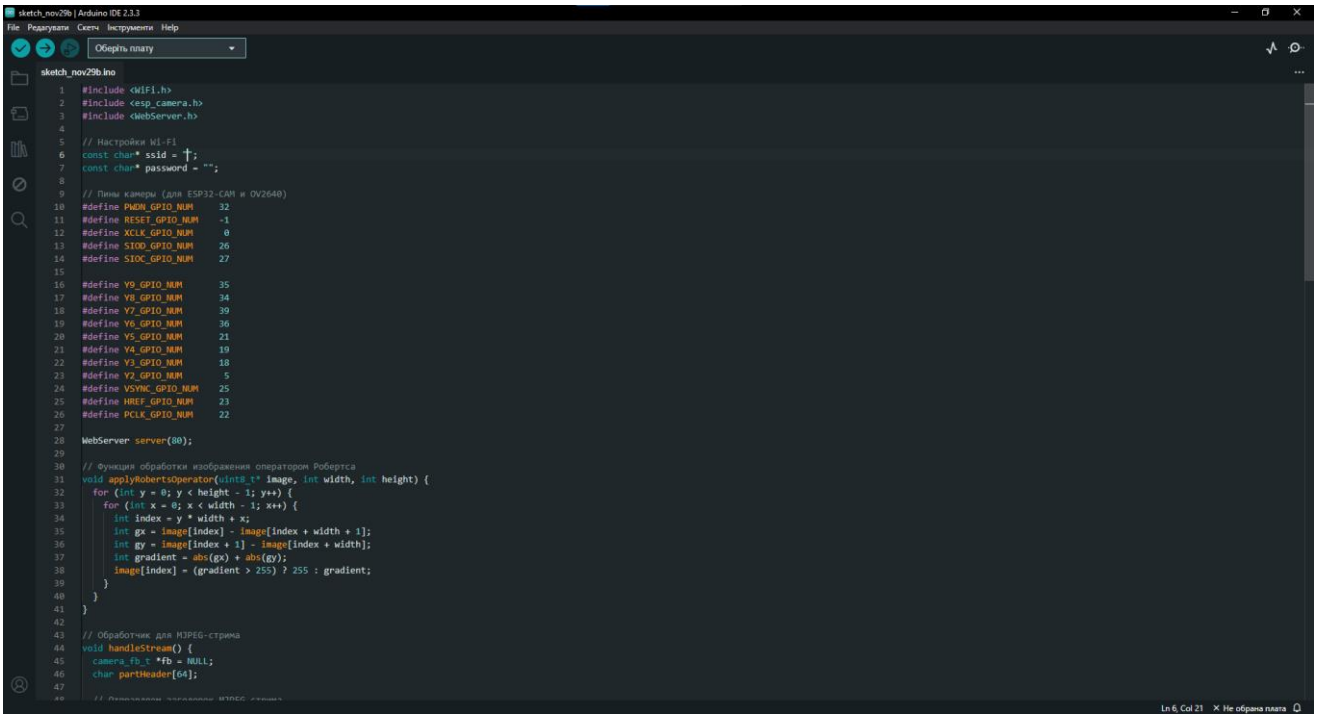

Рисунок 2.3 – Arduino IDE на прикладі коду

2.4 Розпізнавання обличчя бібліотекою ESP-WHO

ESP32 – це серія мікроконтролерів типу «система на кристалі», що мають інтегровані контролери Wi-Fi і Bluetooth, низьке енергоспоживання і невисоку ціну. У серії ESP32 використовується мікропроцесор Tensilica Xtensa LX6 в двоядерних та одноядерних варіаціях та включає вбудовані антенні перемикачі, радіочастотний балун, підсилювач потужності, приймач з низьким рівнем шумів, фільтри та модулі керування живленням. ESP32 створений та розроблений компанією Espressif Systems, китайською компанією, розташованою у Шанхаї, а виробляється компанією TSMC. Він є наступником мікроконтролера ESP8266.

Компанія Espressif Systems пропонує свій офіційний фреймворк для роботи з ESP32 – Espressif IoT Development Framework. Він включає велику кількість бібліотек, серед яких бібліотека Arduino-esp32, вона дозволяє запрограмувати чіп у

середовищі Arduino IDE. Загалом бібліотека не має чіткої направленості і пропонує базовий функціонал інших, більш вузько спеціалізованих бібліотек, таких як наприклад ESP-WHO [20]. ESP-WHO – це платформа розробки обробки зображень на основі чипів Espressif. Містить приклади розробки, які можна використовувати у практичних додатках. Розглянемо один з таких прикладів.

Проект складається з 4-ох файлів, перший CameraWebServer.ino, він містить об’яву двох загальних змінних, ssid для назви Wi-Fi мережі підключення та password для паролю від мережі. Далі функція setup() – це початкова точка налаштування програми на ESP32, вона виконує підготовку пристрою до роботи. В ній виконується ініціалізація послідовного порту зі швидкістю 115200 для виводу помилок та налагоджувальних повідомлень в консоль. Далі проводиться налаштування параметрів через створення структури camera_config_t, де задаються піни підключення, частота тактового сигналу камери через xclk_freq_hz та формат зображення через PIXFORMAT_JPEG. Далі необхідно визначити режим роботи камери, створивши запит на перевірку доступності PSRAM, якщо пам’ять доступно максимальна якість буде UXGA – 1600x1200:

```
if(psramFound()){
   config.frame_size = FRAMESIZE_UXGA;
   config.jpeg_quality = 10;
   config.fb_count = 2;
} else {
   config.frame_size = FRAMESIZE_SVGA;
   config.jpeg_quality = 12;
   config.fb_count = 1;
}
```

Далі проводиться ініціалізація камери з перевіркою на помилку, у разі якої в послідовний порт виводиться помилка:

```
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}
```

Проводимо підключення до Wi-Fi, ініціалізація запуску функції startCameraServer(), та виводимо у послідовний порт локальний IP-адрес по якому можно підключитися до камери:

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
startCameraServer();
Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect");
```

У функції setup() ми бачимо виклик функції startCameraServer(), яка знаходиться у файлі app_httpd.cpp. У функції ми проводимо ініціалізацію HTTP-сервера зі стандартними параметрами та задаємо та налаштовуємо маршрути які сервер буде оброблювати:

```
httpd_config_t config = HTTPD_DEFAULT_CONFIG();
httpd_uri_t index_uri = {
    .uri       = "/",
```

```
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
};
```

Кожен маршрут включає в себе handler – покажчик на функцію-обробник для цього URI. Загалом 5-ть маршрутів, кожен вказує на свою функцію. Наприклад функція index_handler представляє собою обробник HTTP-запиту, пов'язаний з URI/, її завдання відправити клієнту HTML-сторінку, стиснуту з використанням gzip, яка є інтерфейсом користувача для керування камерою:

```
static esp_err_t index_handler(httpd_req_t *req) {
    httpd_resp_set_type(req, "text/html");
    httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
    sensor_t * s = esp_camera_sensor_get();
    if (s->id.PID == OV3660_PID) {
        return httpd_resp_send(req, (const char *)index_ov3660_html_gz, index_ov3660_html_gz_len);
    }
    return httpd_resp_send(req, (const char *)index_ov2640_html_gz, index_ov2640_html_gz_len);
}
```

Нижче проводиться конфігурація алгоритму розпізнавання обличь та ініціалізація бази даних обличь:

```
mtmn_config.type = FAST;
mtmn_config.min_face = 80;
...
face_id_init(&id_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);
```

У проекті використовується метод розпізнавання обличь MTMN – це легка модель розпізнавання обличчя людини, яка побудована на новій мобільній архітектурі під назвою MobileNetV2 і багатозадачних каскадних згорткових мережах і спеціально розроблена для вбудованих пристроїв [21].

MTMN складається з трьох основних частин:

– мережа пропозицій (P-Net): пропонує обмежувальні рамки-кандидати та надсилає їх до R-Net;

– уточнення мережі (R-Net): екранує обмежувальні рамки з P-Net;

– вихідна мережа (O-Net): виводить остаточні результати, тобто точну обмежувальну рамку, коефіцієнт достовірності та орієнтир із 5 точок.

На рисунку 2.4 показана діаграма робочого процесу MTNM.

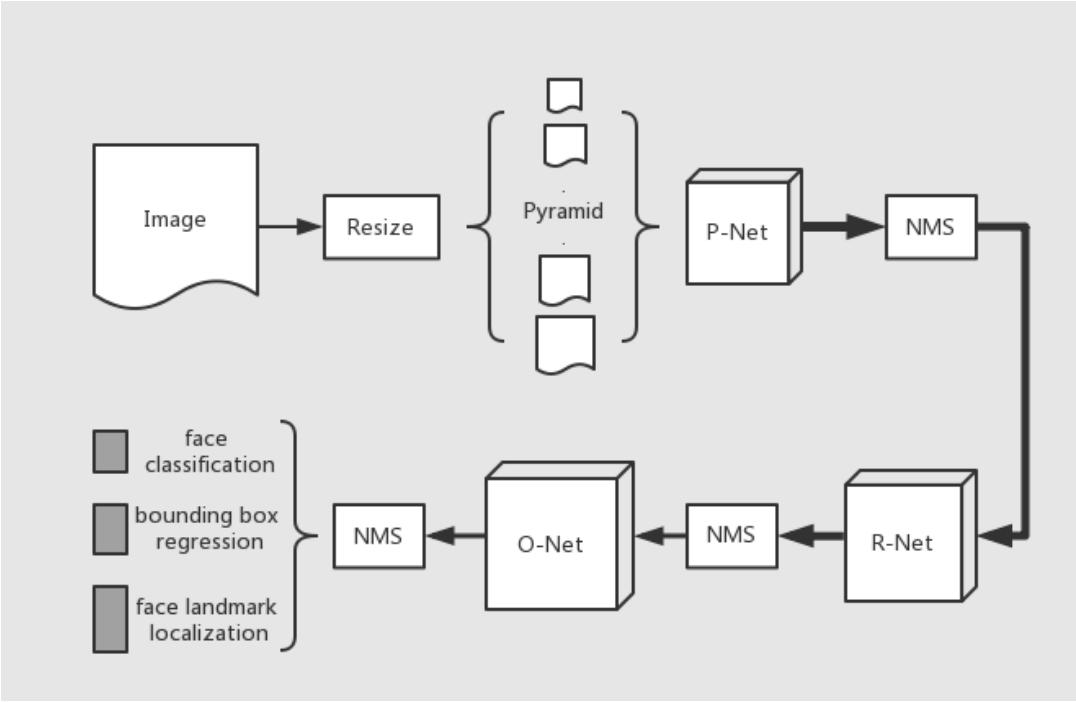

Рисунок 2.4 – Діаграма роботи MTNM

У випадку якщо обличчя було знайдено, є можливість увімкнути код розпізнавання:

```
static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
```

```
    dl_matrix3du_t *aligned_face = NULL;
    int matched_id = 0;
    aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
    if(!aligned_face){
        Serial.println("Could not allocate face recognition buffer");
        return matched_id;
    }
    if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
        if (is_enrolling == 1){
            int8_t left_sample_face = enroll_face(&id_list, aligned_face);
            if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
                Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
            }
            Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail,
ENROLL_CONFIRM_TIMES - left_sample_face);
            rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]",
id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
            if (left_sample_face == 0){
                is_enrolling = 0;
                Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
            }
        } else {
            matched_id = recognize_face(&id_list, aligned_face);
            if (matched_id >= 0) {
                Serial.printf("Match Face ID: %u\n", matched_id);
                rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u",
matched_id);
            } else {
                Serial.println("No Match Found");
                rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
```

```
            matched_id = -1;
          }
        }
    } else {
        Serial.println("Face Not Aligned");
        //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
    }
    dl_matrix3du_free(aligned_face);
    return matched_id;
}
```

Код виконує розпізнавання обличчя на зображенні. Спочатку створюється буфер для зберігання вирівняної особи, позначений як aligned_face. Якщо буфер не вдалося створити, повертається ідентифікатор, що дорівнює нулю. Потім відбувається вирівнювання обличчя за допомогою функції align_face, яка використовує виявлені рамки. Якщо вирівнювання пройшло успішно і система знаходиться в режимі реєстрації is_enrolling, відбувається додавання особи до бази даних за допомогою функції enroll_face. Якщо це останній зразок реєстрації, відображається повідомлення про завершення реєстрації. Якщо система не в режимі реєстрації, спроба розпізнавання обличчя за допомогою функції recognize_face. Якщо його знайдено, виводиться його ідентифікатор. Якщо особа не знайдена, виводиться повідомлення про тривогу. Наприкінці звільняється пам'ять, зайнята буфером aligned_face.

На рисунку 2.5 показаний результат роботи пошуку обличчя, його розпізнавання із занесення у базу даних та вивід рамки із ID користувача.

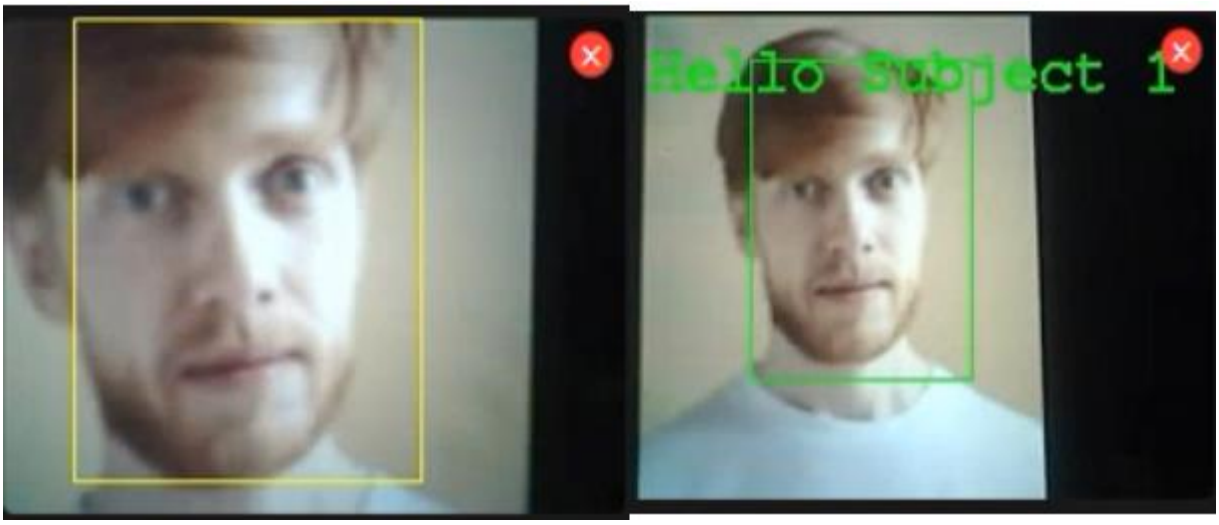

Рисунок 2.5 – Результат роботи програми

2.5 Побудова системи розпізнавання за допомогою OpenCV

Принцип роботи системи розпізнавання на основі OpenCV полягатиме в тому, що основна логіка виконується на сторонньому сервері, а не на чіпі камери. IP-камера буде виконувати лише функції передачі зображення через власний Web-сервер, де у подальшому зображення буде зчитуватися програмою. Можна використовувати стандартний Web-сервер, але його недоліком є те, що окрім трансляції зображення, ресурси процесору витрачаються на підтримку іншого функціоналу, що використовуватися не буде. Тож доцільнішим буде написання коду, що надає функціонал лише трансляції зображення. Для цього будемо використовувати бібліотеку esp32cam [22]. Ця бібліотека, користувацька надбудова над ESP-WHO, що надає більш комфортний та простий доступ до функціоналу трансляції зображення.

Розглянемо основні частини коду. Створюється власна функція в якій проходить розрахунок тривалості потокової передачу одного кадру, також викликається метод streamMjpeg для початку потокової передачі відео, що належить класу esp32cam::Camera, в якому в якості аргумента передається клієнт:

```
void serveMjpeg() {
```

```
  Serial.println("MJPEG streaming begin");
  auto startTime = millis();
  int nFrames = esp32cam::Camera.streamMjpeg(server.client());
  auto duration = millis() - startTime;
  Serial.printf("MJPEG streaming end: %dfrm %0.2ffps\n", nFrames, 1000.0 * nFrames
/ duration);}
```

Проводимо налаштування камери. Використовуємо поле імен esp32cam, що дозволяє не вказувати префікс esp32cam:: для усіх наступних викликів с цього поля імен. У змінній res проводимо пошук відповідної роздільності для камери із заданими параметрами. Створюємо об’єкт конфігурації Config cfg, куди записуємо піни камери, роздільну здатність та якість картинки. Проводимо ініціалізацію камери із заданими параметрами через begin(cfg). Налаштовуємо сервер на видачу запитів через /stream.mjpeg та запускаємо його:

```
{
    using namespace esp32cam;
    auto res = esp32cam::Resolution::find(800, 600);
    esp32cam::Config cfg;
    cfg.setPins(esp32cam::pins::AiThinker);
    cfg.setResolution(res);
    cfg.setJpeg(80);
    bool ok = esp32cam::Camera.begin(cfg);
    if (!ok) {
      Serial.println("camera initialize failure");
      delay(5000);
      ESP.restart();
    }Serial.println("camera initialize success");
  }
  server.on("/stream.mjpeg", serveMjpeg);
```

```
server.begin();}
```

На рисунку 2.6 показаний результат роботи.

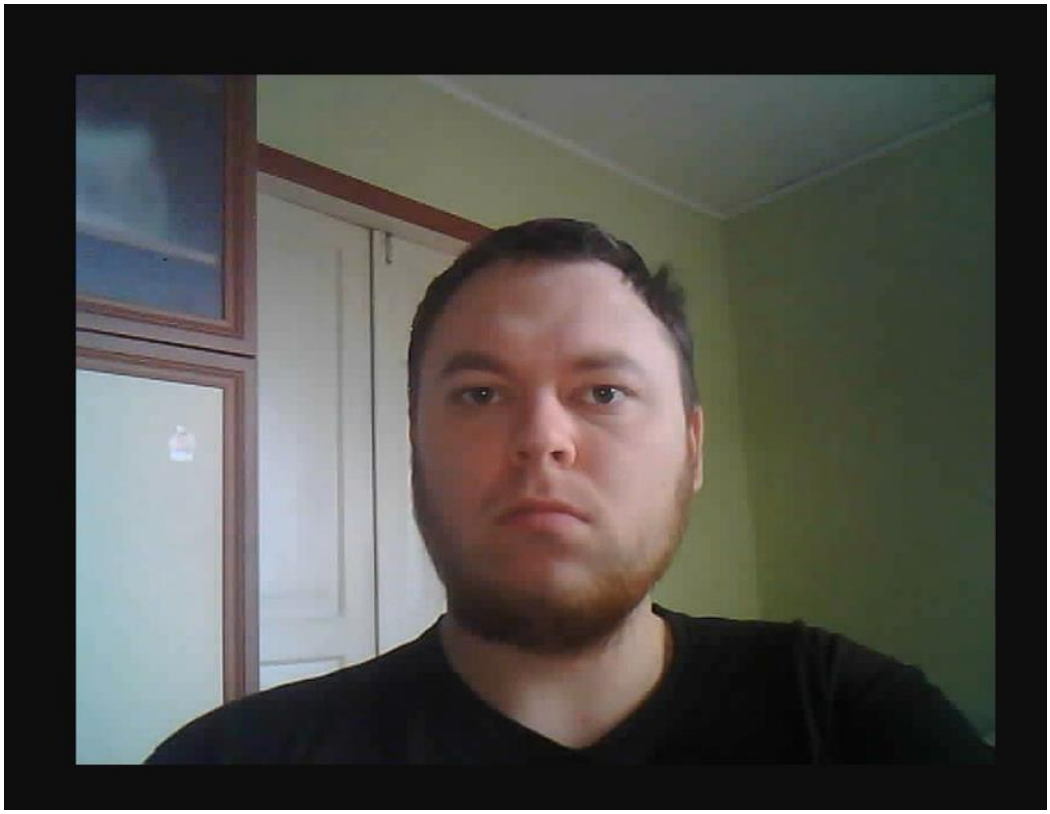

Рисунок 2.6 – Результат роботи

Наступним кроком пишемо виконавчий код серверної частини, за допомогою Python та OpenCV. В OpenCV нас цікавитиме каскадний класифікатор, а саме метод Haar, для знаходження обличчя, та метод Local Binary Patterns Histograms (LBPH) для розпізнавання, а також методи захоплення та фільтрації зображення [23]. Програма буде складатися з трьох файлів: collect.py, train.py та main.py.

Файл collect.py буде проводити збір матеріалів для подальшого навчання алгоритму LBPH. Для цього потрібно провести захоплення зображення та виділити на ньому обличчя. Якість роботи алгоритму залежить від кількості матеріалу для навчання, тому ми будемо створювати велику кількість зображень з назвою User.id.n, де id – це номер користувача, а n порядковий номер зображення для цього користувача.

Створюємо змінну user_id та вводи id користувача при запуску програми, далі проводимо ініціалізацію змінної p_camera_url куди записується адреса трансляції, у змінній cap проходить захоплення трансляції, змінна face_cascade записує метод знаходження обличчя, що використовується, створюємо вікно трансляції та оголошуємо змінні count та max_count:

```
user_id: int = int(input("Enter user ID: "))
ip_camera_url = "http://192.168.1.9/stream.mjpeg"
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades)
cap = cv2.VideoCapture(ip_camera_url)
cv2.namedWindow("live transmission", cv2.WINDOW_AUTOSIZE)
count: int = 0
max_count: int = 200
```

Далі створюємо цикл while, що працює за умови count < max_count. Вводячи значення max_count, ми регулюємо кількість ітерацій циклу, а отже і кількість зображень для навчання. Кожну ітерацію циклу ми зчитуємо значення кадру через cap.read(), далі переводимо кадр у градації сірого, та знаходимо на отриманому зображенні обличчя методом face_cascade.detectMultiScale. Створюємо цикл for, що приймає значення координат прямокутника за віссю, що оточує обличчя, а також його ширину та висоту. Через функцію cv2.imwrite, область зображення що дорівнює обличчю зберігається з заданим user_id. Функція cv2.rectangle обводить знайдене обличчя рамкою, а функція cv2.imshow оновлює вікно трансляції за останнім кадром, завдяки чому за знаходження обличчя можливо спостерігати в реальному часі. Код циклу наведений нижче:

```
while count < max_count:
    ret, frame = cap.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

```
    faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5)
    for (x, y, w, h) in faces:
        count += 1
        cv2.imwrite('DataFaces/User.' + str(user_id) + "." + str(count) + ".jpg", gray_frame[y:y + h, x:x + w])
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    if count >= max_count:
        print("Dataset collection done.....")
        break
    cv2.imshow('live transmission', frame)
```

У файлі train.py проводиться навчання моделі LBPH. На початку коду приймаються змінні recognizer для запису об’єкту розпізнавача обличь на основі методу LBPH та path, що встановлює шлях до директорії де зберігаються зображення отримані після роботи collect.py. Далі створюється функція getImageID, що приймає path та повертає два списки – faces та ids. У циклі for кожні зображення відкривається та записується у змінну faceNP перетворюючи зображення на масив NumPy, записуємо у faces. Витягуємо ідентифікатор користувача з імені файлу та записуємо у список ids. Код функції нижче:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
path = "DataFaces"
def getImageID(path):
    imagePath = [os.path.join(path, f) for f in os.listdir(path)]
    faces=[]
    ids=[]
    for imagePaths in imagePath:
        faceImage = Image.open(imagePaths).convert('L')
        faceNP = np.array(faceImage)
```

```
        Id= (os.path.split(imagePaths)[-1].split(".")[1])
        Id=int(Id)
        faces.append(faceNP)
        ids.append(Id)
        cv2.waitKey(1)
    return ids, faces
```

Викликаємо функцію та записуємо результати у змінні IDs та facedata. Використовуємо метод train() для навчання розпізнаванню на даних facedata. За допомогою методу write(), зберігаємо модель у файл формату yml. Код наведений нижче:

```
IDs, facedata = getImageID(path)
recognizer.train(facedata, np.array(IDs))
recognizer.write("Trainer.yml")
print("Training Completed.....")
```

У файлі main.py використовується код створення трансляції та знаходження обличчя на зображенні, як і у collect.py, але у циклі for викликається метод predict(), що повертає serial ідентифікатор користувача, якщо він розпізнаний, та коефіцієнт conf, що показує рівень впевненості. Якщо коефіцієнт вище якогось заданого нами порогу, тоді програма вважатиме обличчя розпізнаним, якщо коефіцієнт нижче – виводиться текст Unknown, що сповіщає про те що користувач не ідентифікований. Код нижче циклу нижче:

```
for (x, y, w, h) in faces:
        serial, conf = recognizer.predict(gray[y:y + h, x:x + w])
        if conf > 50:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 1)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (50, 50, 255), 2)
```

```
            cv2.rectangle(frame, (x, y - 40), (x + w, y), (50, 50, 255), -1)
            cv2.putText(frame, name_list[serial], (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
        else:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 1)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (50, 50, 255), 2)
            cv2.rectangle(frame, (x, y - 40), (x + w, y), (50, 50, 255), -1)
            cv2.putText(frame, "Unknown", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
```

На рисунках 2.7 - 2.9 продемонстровано робота програми.

Повний код файлів наведений у додатку Б.

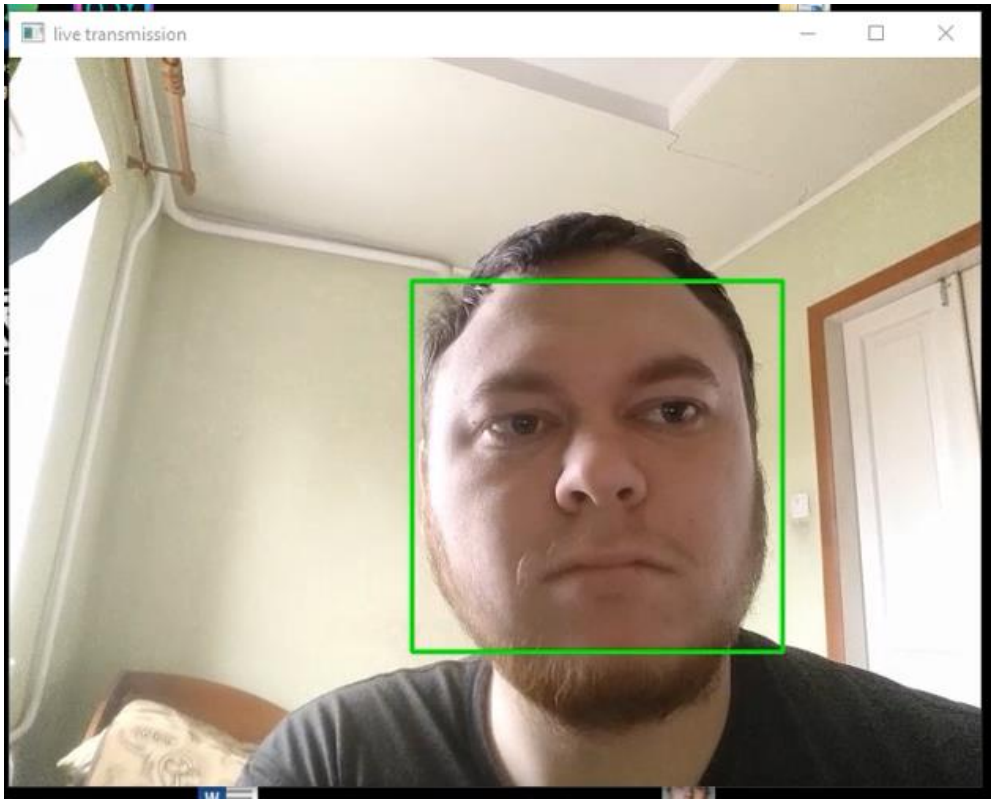

Рисунок 2.7 – Збір даних для подальшого навчання

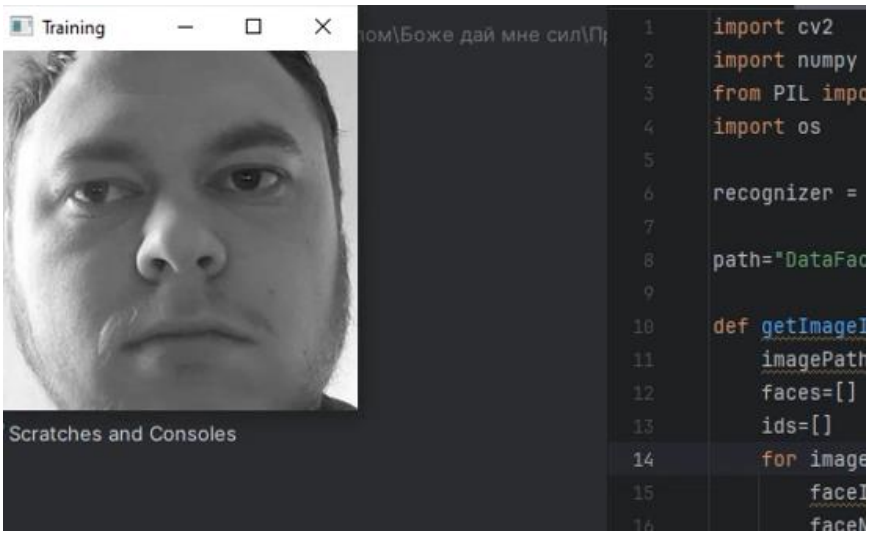

Рисунок 2.8 – Процес тренування алгоритму

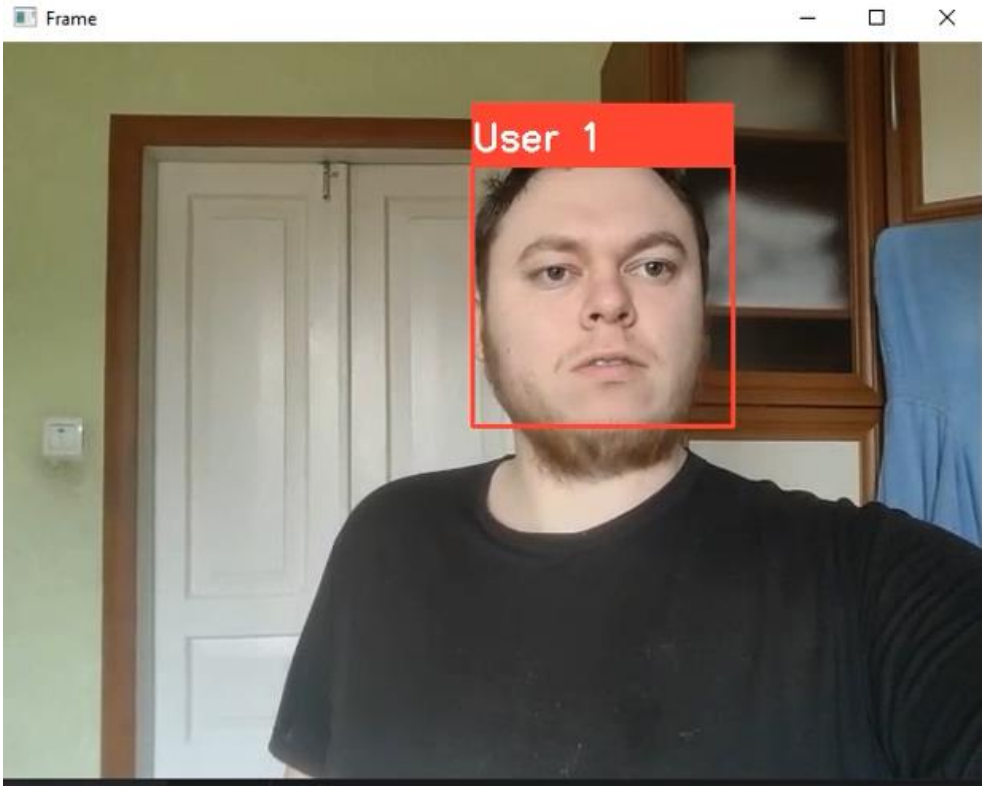

Рисунок 2.9– Результат розпізнавання

2.6 Система розпізнавання на базі Face Recognition

Написання логіки розпізнавання лише з використанням OpenCV не потребує великої обчислювальна потужності системи. Використання бібліотеки Face Recognition накладає більші обмеження, але пропонує більш точний результат [24].

Ця бібліотека побудована на базу dlib та використовує глибокі згорткові нейронні мережі для обробки зображень та метод HOG (гістограма спрямованих градієнтів) для знаходження обличь. Для роботи із бібліотекою необхідно додатково встановити dlib та CMake.

Як і в випадку реалізації на OpenCV, програма складатиметься з декількох файлів. У файлі data_collect.py описана логіка збору зображень обличь для подальшого навчання. Розглянемо основні компоненти.

На відміну від моделі LBPH з OpenCV, для навчання згорткові мережі із бібліотеки Face Recognition, потрібна набагато менша кількість вхідних зображень. Створюємо змінні, що відповідатимуть за ім’я користувача та бажану кількість зображень. За допомогою f-рядку створюємо шлях для папки. Функцією os.makedirs створюється директорія для збереження зображень для навчання:

```
username = input("Enter username: ")
max_count = int(input("Enter the number of photos to collect: "))
user_dir = f"dataset/{username}"
os.makedirs(user_dir, exist_ok=True)
```

Використовуючи функцію face_locations() знаходимо обличчя в кадрі та повертаємо список координат прямокутників, що охоплюють знайдене обличчя. Перевіряємо за допомогою if чи знайдене обличчя, якщо так, беремо перше знайдене та вирізаємо по координатам top, right, bottom, left. Отримане зображення зберігається в папці користувача user_dir з унікальним ім'ям, що включає ім'я користувача та лічильник count. Код наведений нижче:

```
face_locations = face_recognition.face_locations(frame)
    if face_locations:
        top, right, bottom, left = face_locations[0]
        face_img = frame[top:bottom, left:right]
        filename = os.path.join(user_dir, f"{username}_{count}.jpg")
```

```
cv2.imwrite(filename, face_img)
cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
cv2.putText(frame, f"Photos: {count + 1}/{max_count}",
            (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
count += 1
```

Таким чином ми отримаємо директорію, яка складається з папки dataset в якій створюється папка з іменем користувача, в яку зберігаються зображення обличчя, для подальшого навчання. На рисунку 2.10 показана робота цього коду.

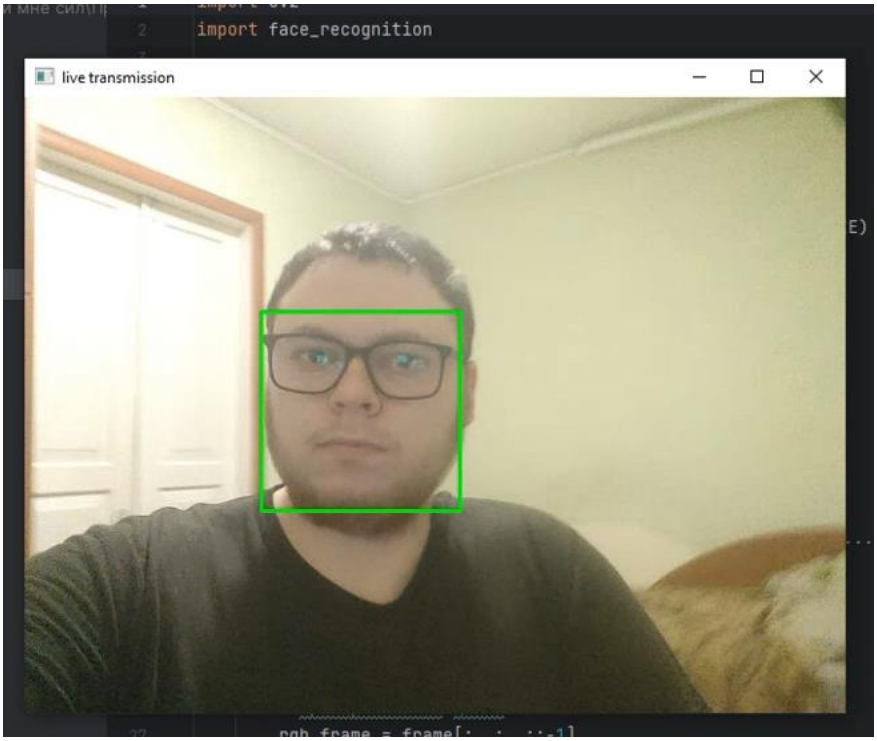

Рисунок 2.10 – Робота функції face_locations()

У файлі main.py описуємо велику кількість логіки програми, тому для більшої зручності буде використано функції. Вони дозволять, у разі необхідності, легко масштабувати програму до повноцінного додатку. При написанні великих проектів головною проблемо є повторення однакового коду, який виконує одні й ті самі задачі, аде в різних частинах програми. Використання функцій дозволить просто викликати їх в необхідний нам момент, безкінечну кількість разів. Такий

підхід є більш об’єктно орієнтовним. На рисунку 2.11 зображено блок схему логіки роботи алгоритму main.py, написаного з використанням функцій Python.

Алгоритм працює наступним чином. Після початку роботи викликається головна функція main(), яка викликає функцію prepare_face_recognition(), що перевіряє чи створений файл кодування обличчя. Якщо файлу немає, він видалений або це перший запуск алгоритму, викликається функція encode_known_faces(). Вона проводить навчання системи кодуючи обличчя у файл з розширенням pkl, використовуючи зображення з папки dataset. Після алгоритм починає роботу з початку, якщо файл в перший раз було створено правильно, викликається функція recognize_faces(). В ній описана логіка порівняння зображення з кадру із закодованими обличчями. Для цього всередині recognize_faces() викликається функція load_encoded_faces(), що повертає закодовані обличчя, та робить ще одну перевірку на їх наявність. У разі відсутності вона викликає encode_known_faces() для повторення процесу спочатку.

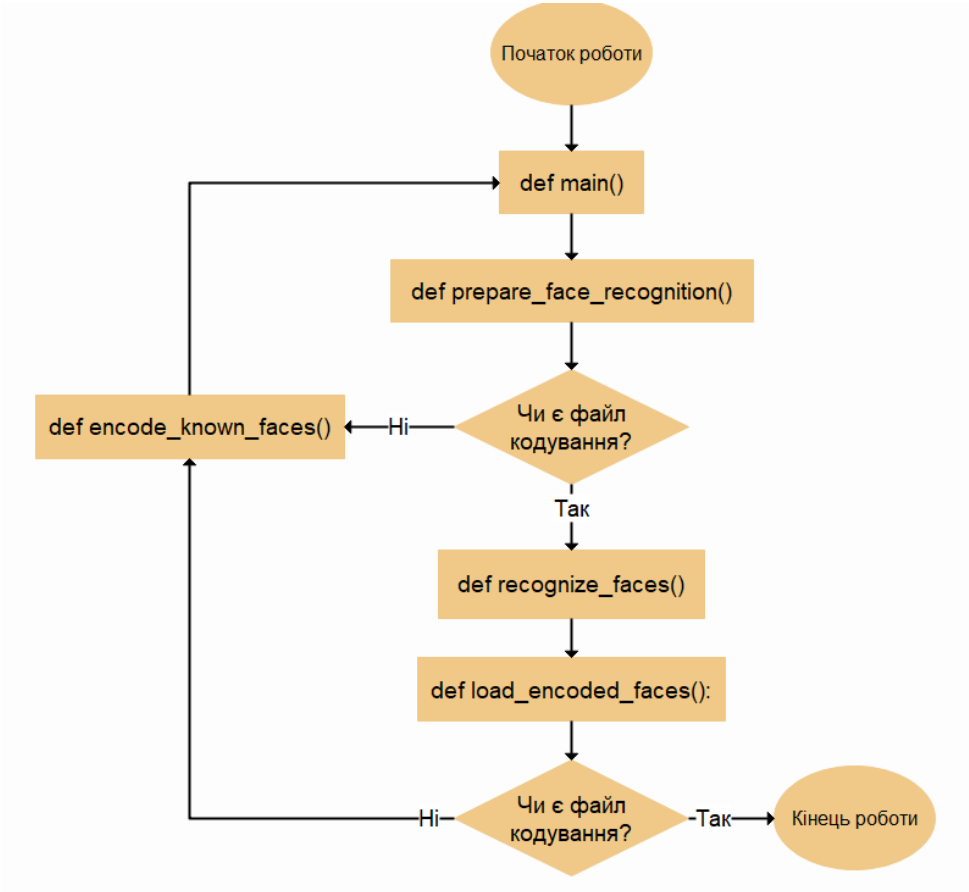

Рисунок 2.11 – Схема роботи алгоритму програми файлу main.py

Розглянемо головній функції більш детально.

Функція encode_known_faces відповідає за створення та збереження закодованих обличь, вона зберігає дані у файл known_faces.pkl для подальшого використання. У функції створюються два цикли for для проходу по папкам користувачів та файлам всередині. У середині циклу функцією load_image_file завантажується зображення, а функцією face_encodings виконує кодування створюючи 128-мірний ембедінг особи, повертає список ембеддингів, по одному для кожної виявленої особи у зображенні. Конструкцією if перевіряється чи пройшло кодування, та виводиться повідомлення з номером файлу який був чи не був закодований. У конструкції with за допомогою функції pickle.dump, зберігаються дані у файл known_faces.pkl у бінарному форматі. Код функції наведений нижче:

```
def encode_known_faces():
    known_faces = []
    known_names = []
    # Цикл проходу по папкам користувачів
    for username in os.listdir("dataset"):
        user_dir = os.path.join("dataset", username)
        # Цикл проходу по файлам в папці користувача
        for filename in os.listdir(user_dir):
            image_path = os.path.join(user_dir, filename)
            try:
                # Завантаження і кодування
                image = face_recognition.load_image_file(image_path)
                encodings = face_recognition.face_encodings(image)
                # Якщо кодування успішне завантажуємо
                if encodings:
                    known_faces.append(encodings[0])
```

```
                known_names.append(username)
                print(f"Face successfully encoded: {image_path}")
            else:
                print(f"Could not find face in image: {image_path}")
        # Помилка на файлі №…
        except Exception as e:
            print(f"Processing error {image_path}: {e}")
# Збереження закодованих обличь
with open('known_faces.pkl', 'wb') as f:
    pickle.dump({
        'encodings': known_faces,
        'names': known_names
    }, f)
print(f"Encoded persons: {len(known_faces)}")
return known_faces, known_names
```

Функція recognize_faces() виконує розпізнавання осіб у реальному часі, використовуючи веб-камеру. Спочатку виконується завантаження закодованих ембеддингів та їх імен із файлу чи бази даних, та виконується перевірка. Якщо список known_faces порожній, виводиться повідомлення, що не обличь для розпізнавання, і виконання функції припиняється. Виконується захоплення камери так запускається безкінечний цикл while. У циклі захоплюється кадр та функціями face_locations та face_encodings знаходиться обличчя в кадрі та кодується у 128-бітний формат. У циклі for функція compare_faces() порівнює поточний ембеддинг із кожним відомим обличчям та повертає список булевих значень. Якщо особа збіглася, прописує True. Якщо збігу немає, змінна name залишається Unknown. Якщо є збіг, знаходить індекс першої збіглої особи та отримує ім'я користувача, що відповідає ембеддингу. Далі функція малює прямокутник навколо знайденого обличчя та виводить значення name та показує поточний кадр з накладеним прямокутником. Код функції нижче:

```
def recognize_faces():
    # Загурзка закодованих значень через виклик функції
    known_faces, known_names = load_encoded_faces()
    # Перевірка наявності відомих обличь
    if not known_faces:
        print("No known faces to recognize!")
        return
    # Захоплення камери
    cap = cv2.VideoCapture(0)
    while True:
        # Захоплюємо кадр
        ret, frame = cap.read()
        # Знаходимо в кадрі обличчя
        face_locations = face_recognition.face_locations(frame)
        face_encodings = face_recognition.face_encodings(frame, face_locations)
        # Обробка знайдених облич
        for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
            # Сравнение с известными лицами
            matches = face_recognition.compare_faces(known_faces, face_encoding)
            name = "Unknown"
            # Визначаємо ім'я якщо особа розпізнана
            if True in matches:
                first_match_index = matches.index(True)
                name = known_names[first_match_index]
            # Прямокутник і ім'я
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
            cv2.putText(frame, name, (left, top - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
        # Поточний кадр
        cv2.imshow('Face Recognition', frame)
```

```
    # Вихід за клавішею 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Звільняємо ресурси
cap.release()
cv2.destroyAllWindows()
```

На рисунках 2.12 можна побачити результат роботи алгоритму.

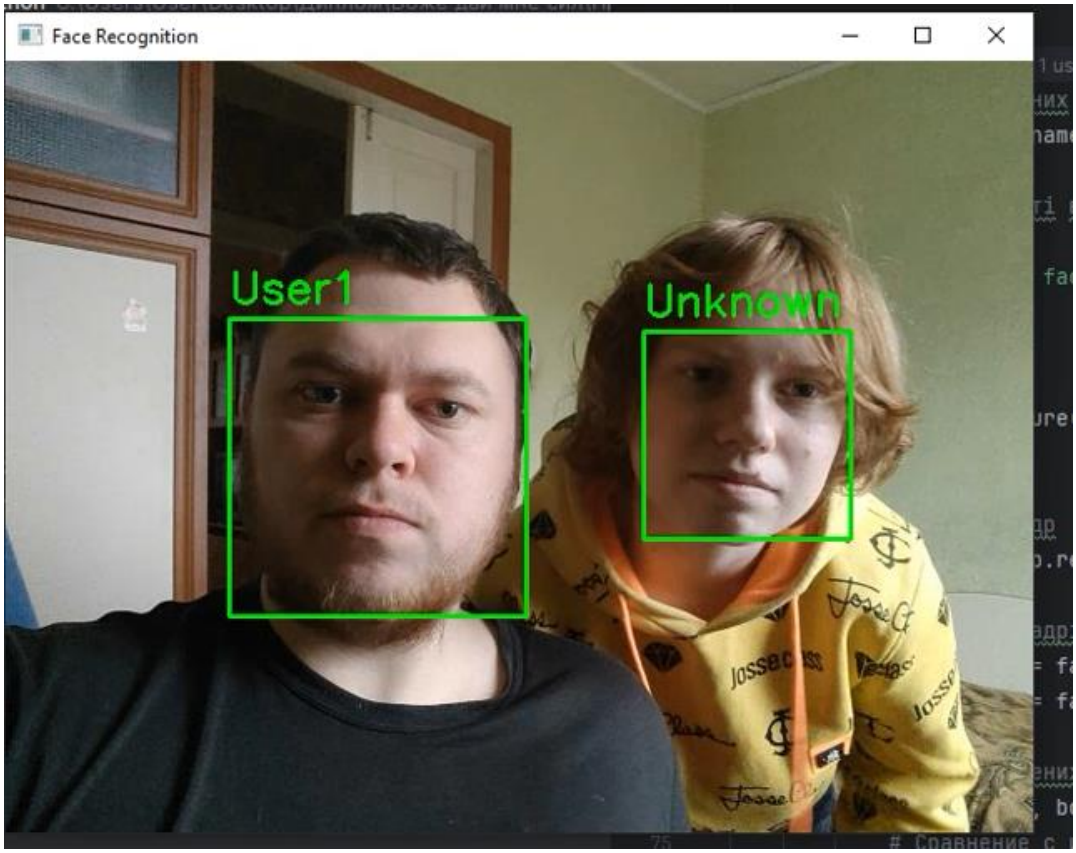

Рисунок 2.12 – Ідентифікація користувача

Повний код файлів наведений у Додатку В

2.7 Електрична схема системи доступу до виробничих приміщень

Згідно визначених раніше функціональних вимог до програми, необхідно реалізувати функції надання доступу до промислового приміщення. Однією з можливих системи фізичного захисту, можуть бути двері із магнітним замком. Така система може бути реалізована за допомогою Arduino. На рисунку 2.13 продемонстрований макет реалізації замку на ESP32.

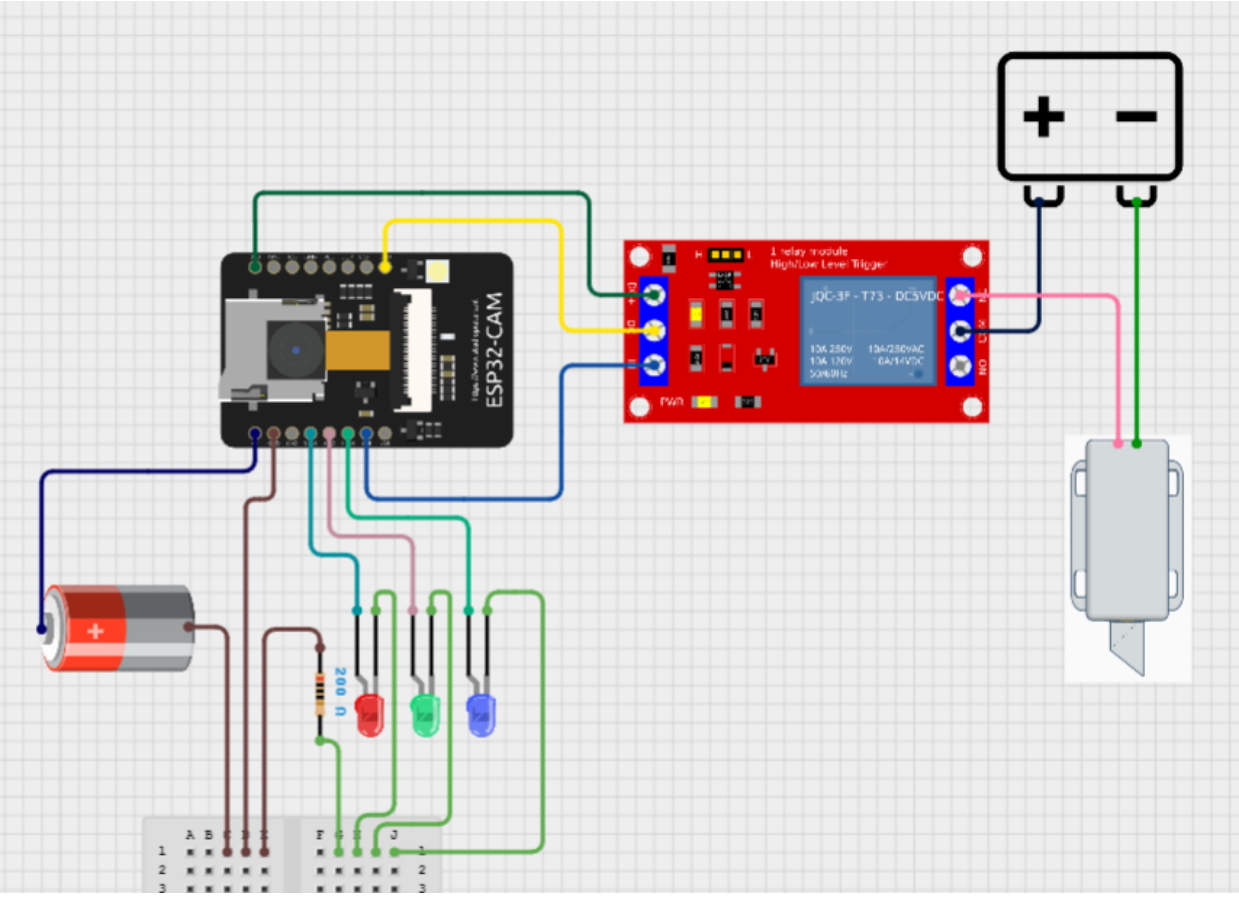

Рисунок 2.13 – Макет підключення електричних компонентів до ESP32

Для роботи плати необхідно надавати живлення не менше 5 В. Через роз'єм GPIO2 виконується керування оптоізольованим релейний модулем на 3 В. До реле підключений електромагнітний замок на 12 В. Замок під'єднаний до постійно замкнутого контакту реле, через що на нього постійно подається живлення. При поданні на реле сигналу з GPIO2 контакт розмикається і замок втрачає живлення. До контактів GPIO13, GPIO14 та GPIO15 під'єднані світлодіоди. Синій світлодіод

горить коли камера не фіксує обличчя в кадрі. Червоний світлодіод сигналізує про те що обличчя розпізнане не було. Зелений світлодіод сигналізує, що обличчя працівника було розпізнане і йому надано пропуск. Розглянемо реалізацію цього функціоналу у коді.

У файл проекту в Arduino IDE необхідно провести ініціалізацію пінів, що будемо використовувати. За замовчування, на кожний з них окрім синього, подається сигнал LOW. Командою Serial.readStringUntil('\n') ми читаємо значення рядку, що отримали через послідовний порт. Загалом в нас є 4 команди. Команда GREEN_ON відкриває замок. GREEN_OFF зачиняє. RED_ON сигналізує про не надання доступу. BLUE_ON повертає стан за замовчуванням. Код програми наведений нижче:

```
#define relayPin 2   // Реле підключене до GPIO2
#define redLED 13    // Червоний світлодіод до GPIO13
#define blueLED 14   // Синій світлодіод до GPIO14
#define greenLED 15  // Зелений світлодіод до GPIO15
void setup() {
  Serial.begin(115200);
  pinMode(relayPin, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  digitalWrite(relayPin, LOW);
  digitalWrite(redLED, LOW);
  digitalWrite(blueLED, HIGH);
  digitalWrite(greenLED, LOW);
}
void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
```

```
    command.trim();  // Видалення зайвих пробілів
    if (command == "GREEN_ON") {
      digitalWrite(greenLED, HIGH);
      digitalWrite(relayPin, HIGH);
      digitalWrite(blueLED, LOW);
    } else if (command == "GREEN_OFF") {
      digitalWrite(greenLED, LOW);
      digitalWrite(relayPin, LOW);
      digitalWrite(blueLED, HIGH);
    } else if (command == "BLUE_ON") {
      digitalWrite(blueLED, HIGH);
      digitalWrite(redLED, LOW);
      digitalWrite(greenLED, LOW);
    } else if (command == "RED_ON") {
      digitalWrite(redLED, HIGH);
      digitalWrite(blueLED, LOW);
      digitalWrite(greenLED, LOW);
    } else {
      Serial.println("Unknown command");
    }
  }
}
```

У середовищі програмування Python, реалізуємо написання функцій в середині окремого класу. Функції, що надсилають команди по послідовному порту, будуть описані в окремому файлі в середині класу ESP32Control. Це дозволить нам викликати не кожну функцію окремо, а одразу весь клас. Код реалізації класу наведений нижче:

```
import serial
```

```
import time
class ESP32Control:
    def __init__(self, port, baud_rate=115200):
        self.port = port
        self.baud_rate = baud_rate
        self.esp = None
    def connect(self):
        """Підключення до ESP32 через вказаний порт."""
        try:
            self.esp = serial.Serial(self.port, self.baud_rate, timeout=1)
            time.sleep(2)
            print(f"Connected to ESP32 on {self.port}")
        except Exception as e:
            print(f"Failed to connect to ESP32: {e}")
    def send_command(self, command):
        """Відправка команди на ESP32."""
        if self.esp and self.esp.is_open:
            self.esp.write((command + '\n').encode())
            response = self.esp.readline().decode().strip()
            print(f"ESP32 Response: {response}")
        else:
            print("ESP32 is not connected!")
    def green_on(self):
        """Ввімкнути зелений світлодіод і реле."""
        self.send_command("GREEN_ON")
    def green_off(self):
        """Вимкнути зелений світлодіод і реле."""
        self.send_command("GREEN_OFF")
    def blue_on(self):
        """Ввімкнути синій."""
```

```
        self.send_command("BLUE_ON")
    def red_on(self):
        """Ввімкнути червоний."""
        self.send_command("RED_ON")
    def close(self):
        """Закрити підключення до ESP32."""
        if self.esp:
            self.esp.close()
            print("Disconnected from ESP32.")
```

Розглянемо короткий приклад використання цього функціоналу. Викликаємо клас ESP32Control з файлу esp32_control, та прописуємо значення port, що за замовчування не вказано. Це значення повинне дорівнювати тому порту до якого підключена ESP32. Далі виконується стандартний код пошуку функцією face_locations() і разу знаходження викликається відповідна функція green_on, що відкриває замок. Код прикладу наведений нижче:

```
from esp32_control import ESP32Control
esp = ESP32Control(port="COM3") # Підключення до ESP32
esp.connect()
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    face_locations = face_recognition.face_locations(frame)
    if face_locations:  # Якщо обличчя знайдено
        print("Face detected!")
        esp.green_on()  # Викликаємо функцію включення світлодіоду
        time.sleep(5)   # Чекаємо 5 секунд
```

```
        esp.green_off()  # Виклик функції вимкнення
    else:
        esp.blue_on()  # Функція ввімкнення синього
    cv2.imshow("Video", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
esp.close()
```

2.8 Висновки до другого розділу

У другому розділі була проведена розробка трьох систем розпізнавання обличчя. Першочергово були описані вимоги до системи грунтуючись на існуючих рішеннях конкурентів. На базі цих вимог, була розроблена загальна схема принципу роботи майбутньої системи. Був проведений аналіз можливих апаратних рішень та обране середовище розробки. Для роботи із ESP32-cam було обране середовище Arduino IDE. Для написання коду на мові програмування Python використовувався PyCharm.

Усі програми створювалися за принципом, що модуль ESP32-cam лише передає відео-сигнал працюючи як IP-камера, а уся логіка розрахунків відбувається на сервері, в якості якого виступав персональний комп’ютер із процесором Ryzen 3100, під керуванням системи Windows 10. Було використано два основні підходи в вирішенні питання розпізнавання – із використанням OpenCV та із використанням бібліотеки dlib.

Для подальшого виміру та порівнянь, була також створена система, що виконує розпізнавання обличчя не використовуючи сервер, а проводячи усі розрахунки безпосередньо на модулі ESP32-cam.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1 Опис експерименту

Метою експерименту є знаходження найбільш ефективного методу пошуку та розпізнавання обличчя за різних умов освітлення. Виміри будуть проходити в приміщенні за трьох різних умов: денне освітлення, на півсутінки, штучне освітлення від LED-лампи у теплому спектрі. Також буде змінюватися роздільна якість відео, з метою знаходження меж ефективної роботи алгоритму.

Дослідження буде поділене на дві частини. Під час першої буде вимірюватися здатність знаходження обличчя у кадрі, трьома описаними вище методами – бібліотека ESP-WHO, алгоритм OpenCV, бібліотека Face Recognition. Під час другої частини буде вимірятися здатність цих системи розпізнавати обличчя в заданих умовах. В OpenCV, окрім моделі розпізнавання LBPH, будуть також досліджені моделі Eigenfaces та Fisherfaces [23].

Результати вимір будуть заноситися в таблиці, у яких буде вказаний метод що вимірюється, умови освітлення, час витрачений на виконання, точність результату, що буде виражена у відсоткового значенні. Точність буде вимірюватися лише у другій частині.

3.2 Обладнання експерименту

Для проведення дослідження буде використано модуль ESP32-Cam з WiFi модулем, що був встановлений на плату адаптер CH340, з модулем камери OV2640 на 2 Мп (рисунок 3.1). Також, для проведення контрольних вимірів, з метою перевірки максимуму можливостей описаних методів, буде використовуватися фронтальна камера телефону OnePlus на 16 Мп. Для передачі IP-сигналу буде використовуватися програмне забезпечення DroidCam [25]. Для виміру рівня

освітлення буде використовуватися програмне забезпечення для смартфону – світломір [26].

Усі дослідження проходили за трьох рівнів освітлення:

– природне освітлення на рівні 300 люкс;

– природне освітлення на рівні 170 люкс;

– штучне освітлення у вигляді настеленої LED-лампи на висоті 1 метру від камери на рівні 65 люкс;

Відстань від камери до обличчя у всіх дослідженнях, дорівнювалась 30 см.

Рисунок 3.1 – Модуль ESP32-Cam на платі CH340

3.3 Дослідження функції знаходження

Першочергово було проведено вимір роботи стандартного прикладу на базі ESP-WHO. Через те що усі вирахування відбуваються безпосередньо на модулі,

програмно встановлено обмеження на максимальну роздільну здатність – вона не перевищує значення 400 на 296 пікселів. З цієї причини заміри для цього методу відбуваються окремо, адже в інших реалізація в нас є можливість встановити більшу роздільну здатність. Для виміру була обрана найбільш та найменш можлива роздільна здатність. Вимірювався час на знаходження та кількість кадрів за секунду, що був в цей момент. При вимірюванні в умовах освітлення приміщення LED-лампою, алгоритм не зміг провести знаходження обличчя. До таблиці 3.1 записані результати на основі середнього значення за десять вимірів.

Таблиця 3.1 – Результати вимірювання часу знаходження

Параметри виміру		Результати виміру	
Роздільна здатність	Рівень освітлення, Лк	Час знаходження, мс	Кількість кадрів за секунду, кадр/с
400x296	300	660 мс	2,1 кадр/с
160x120	300	120 мс	8,3 кадр/с
400x296	170	672 мс	1,5 кадр/с
160x120	170	140 мс	7,7 кадр/с
400x296	65	-	-
160x120	65	-	-

Наступним кроком було виміряно роботу алгоритмів CascadeClassifier від OpenCV та Facelocation з бібліотеки dlib. Так як модуль ESP32-Cam, у цьому сценарії використання, не виконую функцій розрахунку знаходження обличчя, а лише передає картинку, знімаються обмеження на роздільну здатність картинки. Було взято три варіанти, та вимірювався час на знаходження в кадрі та точність з якою це відбувалось, чи захопив алгоритм саме обличчя чи інший предмет у кадрі. Кількість кадрів відео-потоку була сталою і дорівнювала 25. Під час вимірів, різниця у детекції між світлом у 300 люксів та світлом 170-200 люксів, що дорівнювало значенню сутінків, була не значною, тож до таблиці були занесені значення лише при двох параметрах світла. Результати занесені до таблиці 3.2

Таблиця 3.2 – Результати вимірювання часу знаходження для алгоритмів

Параметри виміру			Результати виміру	
Алгоритм роботи	Роздільна здатність	Рівень освітлення, Лк	Час знаходження, мс	Точність, %
CascadeClassifier	320x240	300	18 мс	75%
CascadeClassifier	800x600	300	47 мс	80%
CascadeClassifier	1280x1024	300	72 мс	85%
CascadeClassifier	320x240	65	20 мс	8%
CascadeClassifier	800x600	65	51 мс	41%
CascadeClassifier	1280x1024	65	79 мс	24%
Facelocation	320x240	300	53 мс	98%
Facelocation	800x600	300	345 мс	98%
Facelocation	1280x1024	300	925 мс	98%
Facelocation	320x240	65	55 мс	65%
Facelocation	800x600	65	360 мс	73%
Facelocation	1280x1024	65	975 мс	81%

З метою виміру можливостей роботи реалізованих алгоритмів, був проведений контрольній вимір з використанням фронтальної камери смартфону OnePlus на 16 Мп. Цей вимір буде симулювати роботу професійної IP-камери, що частіше за все використовують на виробництвах. Роздільна здатність була сталою і дорівнювала 720x480. Результати занесені до таблиці 3.3.

Таблиця 3.3 – Результати виміру симуляції

Параметри виміру		Результати виміру	
Алгоритм роботи	Рівень освітлення, Лк	Час знаходження, мс	Точність розпізнавання, %
CascadeClassifier	300	52 мс	86%
Facelocation	300	220 мс	98%
CascadeClassifie	65	54 мс	64%
Facelocation	65	225 мс	98%

3.4 Дослідження функції розпізнавання

Було проведено вимір функції розпізнавання обличчя на базі ESP-WHO. Вимірювався час розпізнавання в кадрі, кількість кадрів та точність. Після внесення обличчя до бази, камера наводилась на обличчя та рахувалося скільки разів вона розпізнала обличчя з першого разу. Записані результати на основі середнього значення за десять вимірів. Результати записані до таблиці 3.4.

Таблиця 3.4 – Результати виміру системи розпізнавання

Параметри виміру		Результати виміру		
Роздільна здатність	Рівень освітлення, Лк	Час розпізнавання, мс	Кількість кадрів за секунду, кадр/с	Точність розпізнавання, %
400x296	300	1258 мс	0,8 кадр/с	53 %
160x120	300	1031 мс	1,0 кадр/с	68%
400x296	170	1436 мс	0,6 кадр/с	44 %
160x120	170	1131 мс	0,9 кадр/с	54 %
400x296	65	-	-	-
160x120	65	-	-	-

Для OpenCV проводилися виміри точності розпізнавання одразу для трьох методів: LBPH, Eigenfaces та Fisherfaces. Кожна модель має свої унікальні порогові значення для розпізнавання, тож для початку будуть виміряні ці значення за різних умов. Порогові значення відображають вірогідність того, що модель розпізнала обличчя, і чим ці пороги менші тим більша ця вірогідність. Вони залежать від кількості наданого для навчання вхідного матеріалу, тому будуть виміряні значення при різній кількості матеріалу для навчання. Результати занесені до таблиці 3.5.

Таблиця 3.5 – Результати вимірювання порогових значень

Параметри виміру		Результати виміру
Алгоритм роботи	Кількість фото для навчання	Порогове значення
LBPH	50	54
LBPH	250	51
LBPH	500	48
Eigenfaces	50	10687
Eigenfaces	250	5100
Eigenfaces	500	6200
Fisherfaces	50	49
Fisherfaces	250	47
Fisherfaces	500	5

Далі було виміряно точність розпізнавання для кожного методу при умові використання моделі навченої з найменшим пороговим значенням. Також до таблиці було додано результат виміру методу FaceRecognition із бібліотеки dlib. Виміри проводилися при сталій роздільній здатності у 800 на 600. Результати занесені до таблиці 3.6.

Таблиця 3.6 – Результати виміру алгоритмів розпізнавання

Параметри виміру		Результати виміру	
Алгоритм роботи	Рівень освітлення, Лк	Час розпізнавання, мс	Точність розпізнавання, %
LBPH	300	44 мс	70%
Eigenfaces	300	125 мс	15%
Fisherfaces	300	105 мс	65%
FaceRecognition	300	0,07 мс	98%
LBPH	65	51 мс	0%
Eigenfaces	65	130 мс	0%
Fisherfaces	65	110 мс	0%
FaceRecognition	65	0.08 мс	95%

Також був проведений контрольний вимір з використання камери смартфону OnePlus на 16 Мп, при одному рівні освітлення, адже згідно вимірів із таблиці 3.3, різниця в рівнях освітлення близька до статистичної похибки.

Таблиця 3.7 – Результати контрольного виміру

Параметри виміру	Результати виміру	
Алгоритм роботи	Час розпізнавання, мс	Точність розпізнавання, %
LBPH	51 мс	83%
Eigenfaces	118 мс	64%
Fisherfaces	99 мс	86%
FaceRecognition	0,1 мс	98%

На рисунках 3.2 - 3.5 показаний процес виміру параметрів.

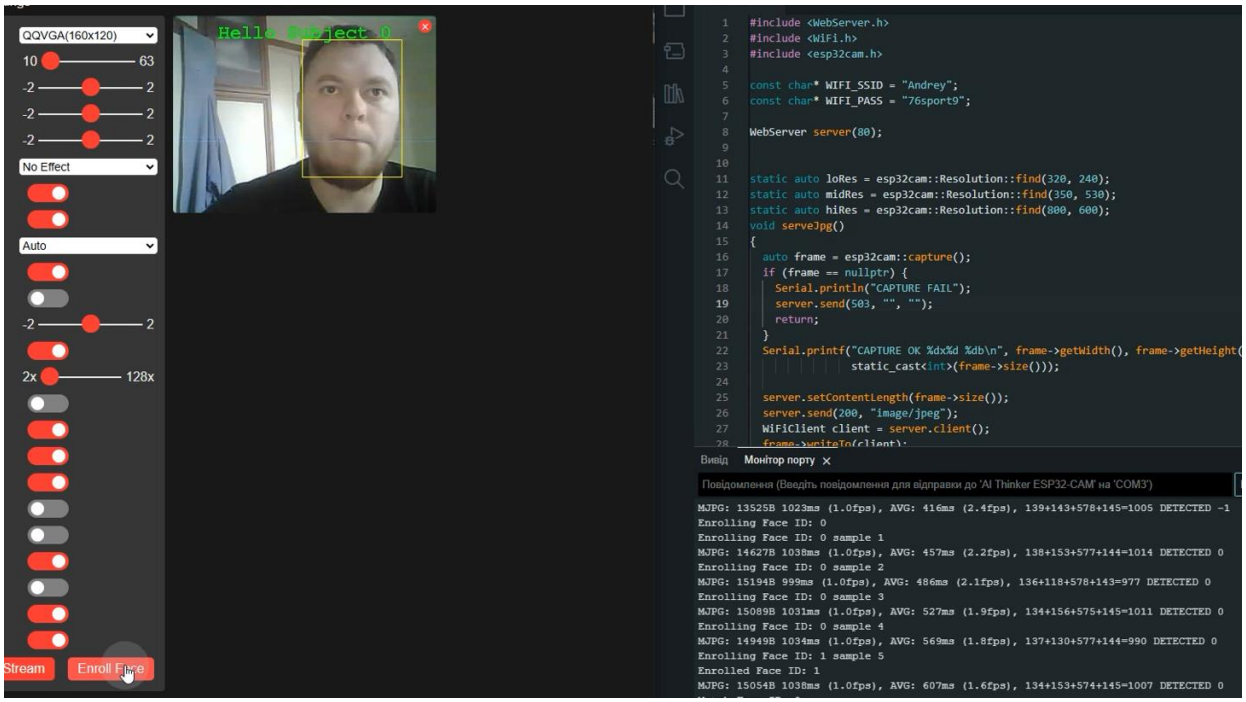

Рисунок 3.2 – Тестування алгоримту розпізнавання ESP-WHO

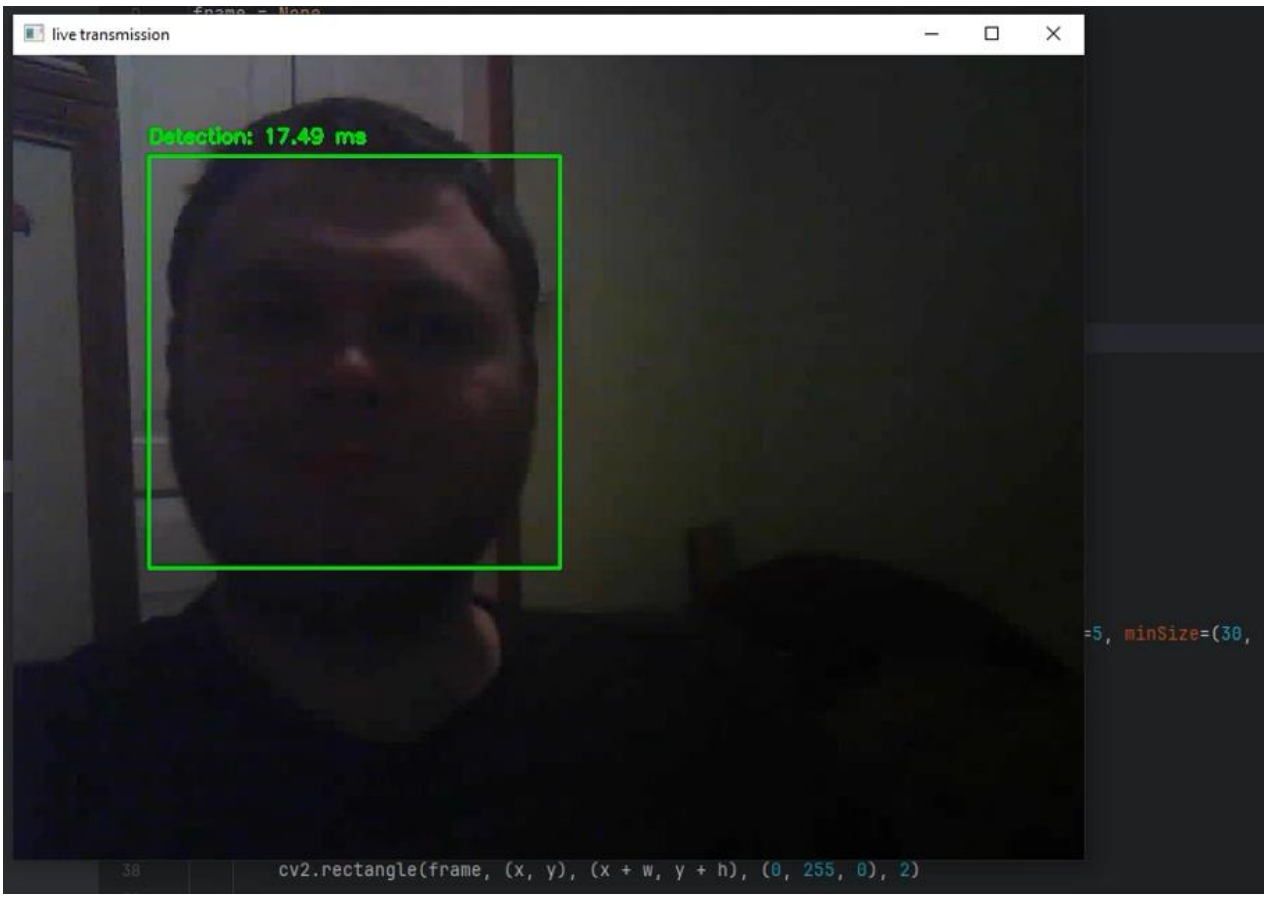

Рисунок 3.3 – Тестування алгоритму детекції обличчя CascadeClassifier

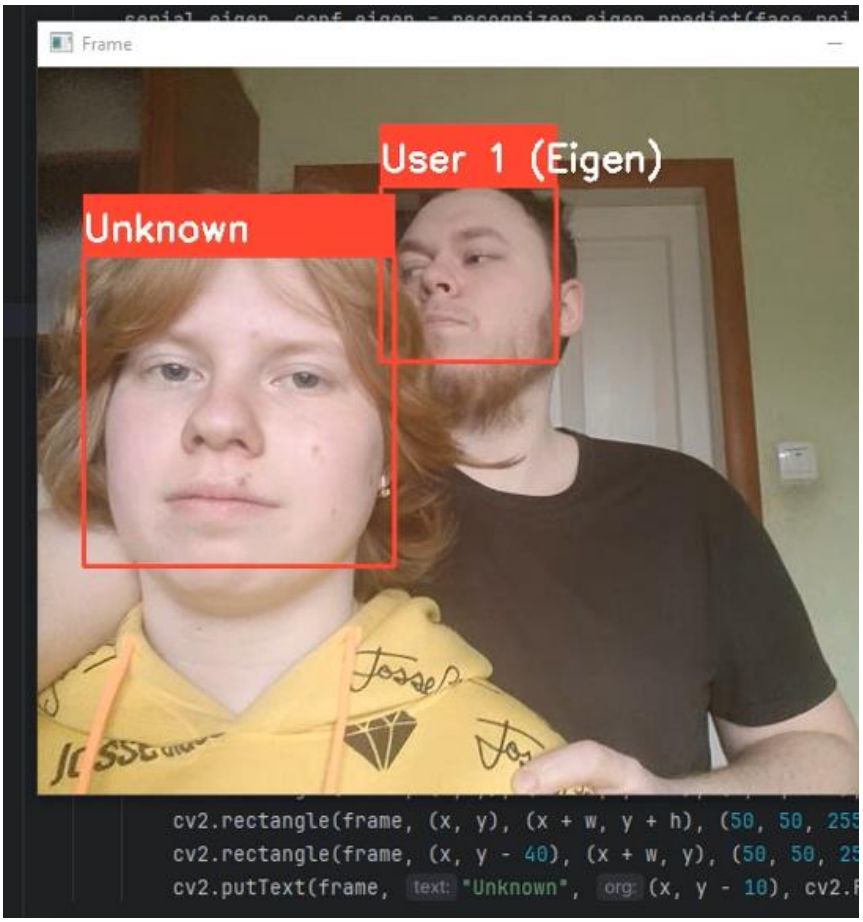

Рисунок 3.4 – Тестування алгоритму розпізнавання Fisherfaces

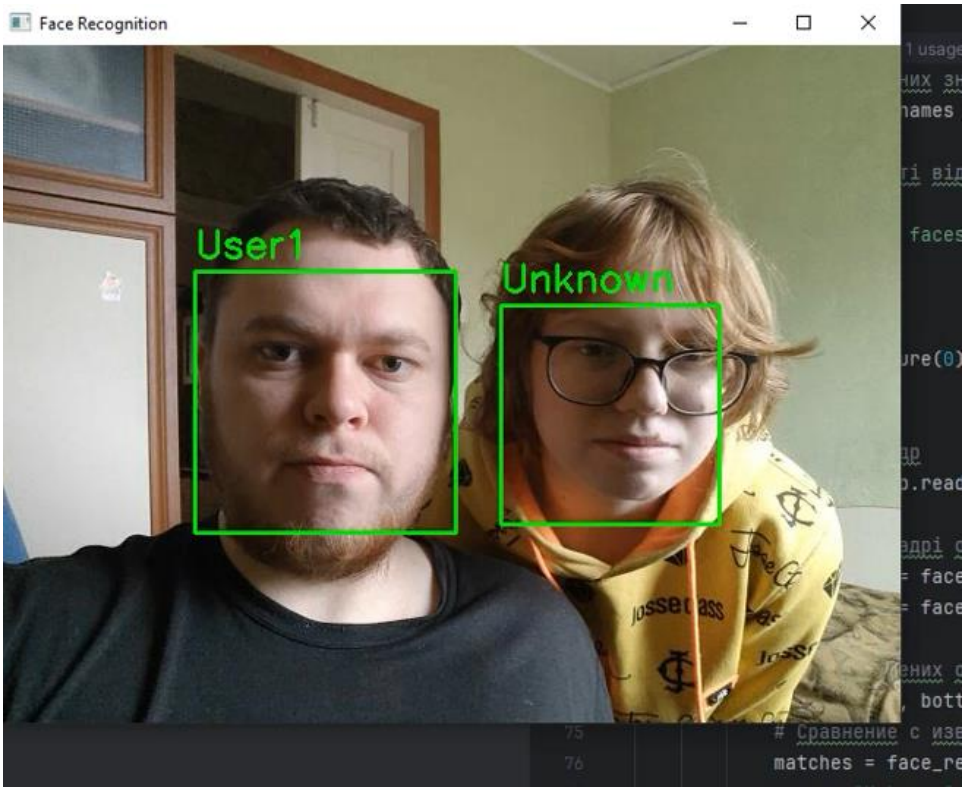

Рисунок 3.5 – Тестування системи розпізнавання на базі FaceRecognition

3.5 Висновки до третього розідлу

На основі проведених вимірів, використання модулю ESP32-Cam, в якості платформи для проведення розрахунків пошуку і розпізнавання обличчя, не є оптимальним варіантом. Стандартна бібліотека ESP-WHO надає не оптимальний алгоритм роботи. Під час вимірювань модуль сильно нагрівався, спостерігалися проблеми кольоропередачі камери, починалися проблеми ініціалізації пінів камери, система падала з причин неправильного використання пам'яті. Запуск алгоритмів пошуку значуще понижував кількість кадрів, що понижує безпеку системи. Обмеження використання розпізнавання лише на малих величинах роздільної здатності підсилює залежність камери модулю від гарного освітлення. Виміри продемонстрували не здатність камери працювати в умовах малого освітлення, що робить модуль не оптимальним варіантом для роботи в умовах промислового приміщення.

Виміри методів пошуку та розпізнавання обличчя продемонстрували доволі значущий недолік алгоритму CascadeClassifie від OpenCV. Він доволі часто знаходить обличчя на неживих об'єктах, як от стіна чи елемент одягу. Але в той же час алгоритм поєднує в собі швидкість роботи і малі вимоги до потужності системи. Алгоритм легко налаштовується і може бути частиною більшої системи. Окрім обличчя він може шукати окрему його частину або частину тіла, що в теорії дозволяє підвищити безпеку системи, шляхом написання умови, що кадр на розпізнавання буде передаватися лише після того, як в знайденому обличчі буде ідентифіковано око чи ніс. До недоліків можна віднести сильну залежність від якості освітлення, залежність падіння якості пошуку від кількості темних не освітлених місць на обличчі, не спроможність знайти обличчя якщо користувач не дивиться прямо в кадр або дивиться в сторону.

Алгоритм Facelocation пропонує більшу терпимість до рівня освітлення та дозволяє користувачу не дивитися строго в камеру. Алгоритм може ідентифікувати обличчя при повороті голови чи легкому відхиленні в бік. Також він краще

ідентифікує в головному уборі чи окулярах. Натомість він потребує більшу потужність системи та має більший час пошуку ніж CascadeClassifie.

Виміри методів розпізнавання показали абсолютну не ефективність алгоритму Eigenfaces. Він найбільш залежний він рівня освітлення та положення голови. Має підвищенні вимоги до якості матеріалу навчання. Алгоритм навчається найдовше з усіх і система під його керування завантажується також найдовше.

Метод Fisherfaces, згідно вимірів порогових значень, має найбільшу ефективність навчання серед усіх алгоритмів OpenCV. Але також потребує потужної системи і доволі сильно навантажує її. Контрольні виміри показали найбільшу точність розпізнавання серед алгоритмів OpenCV. Fisherfaces все ще сильно залежить від якості освітлення, але частіше ідентифікує обличчя при змінені положення голови.

LBPH працює найшвидше із усіх OpenCV алгоритмів, його можна навчати лише на даних одного користувача. Метод найкраще працює в умовах різного освітлення, але пропонує середню точність. Доволі залежний від положення голови та елементів одягу.

Найкращим методом розпізнавання обличчя, за результатами вимірів, є алгоритм FaceRecognition. Найбільша точність за різних умов освітлення, найбільша швидкість. Є єдиним алгоритмом, ідентифікуючим користувача в умовах освітлення 65 люкс при використанні ESP32-Cam зі збереженням точності. До мінусів можна віднести велику залежність від потужності системи.

Таким чином, згідно результатів, можна сказати що найкращим методом для забезпечення безпеки на промисловому підприємстві є FaceRecognition.

4 РОЗРАХУНКОВА ЧАСТИНА

4.1 Теоретичні положення

Для аналізу результатів використано багатовимірний статистичний аналіз з застосуванням кореляційного, регресійного та графічного методів дослідження [27]. Багатовимірний статистичний аналіз вивчає оброблення багатомірних статистичних даних з метою виявлення взаємозв'язків, їх характеру та структури. Суть багатофакторного аналізу полягає в одночасному розгляданні кількох пов'язаних змінних, причому кожна з них вважається однаково важливою. Метою дослідження було виявлення взаємозв'язків між змінними та закономірностей у багато-параметричній системі.

Кореляційний аналіз використовується для оцінки сили та напрямку зв'язку між двома змінними [28]. Його мета – визначити, наскільки сильно зміни однієї змінної пов'язані зі змінами іншої. Міцність зв'язку між двома величинами можна виразити за допомогою коефіцієнта кореляції. Це число r з інтервалу -1, 1. Якщо r близьке до -1, то кореляційний зв'язок між величинами є оберненим, а якщо r близьке до 1 – прямим. Чим ближче r до нуля, тим кореляційний зв'язок слабший. Якщо говорити більш докладно, то міцність лінійного кореляційного зв'язку оцінюється так:

- r >= 0,8 – сильний кореляційний зв’язок;
- 0,4 <= r < 0,8 – кореляційний зв’язок наявний;
- r < 0,4 – кореляційний зв’язок відсутній.

Для побудови кореляційної матриці необхідно розрахувати коефіцієнт кореляції r, для чого буде використовуватися формула кореляції Пірсона. Це найпоширеніший спосіб вимірювання сили лінійного зв'язку між двома змінними X і Y.

Коефіцієнт рахується за формулою:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \times \sum(Y_i - \bar{Y})^2}}, \quad (4.1)$$

де X_i, Y_i – значення змінних X і Y;

$\bar{X}$ $\bar{Y}$ – середнє значення X і Y.

Для встановлення значимості коефіцієнту кореляції, використовуються критерії Ст'юдента. Це статистичний метод, який дозволяє оцінити, чи є виявлений коефіцієнт кореляції значущим, тобто чи існує лінійна залежність між змінними в генеральній сукупності. Для розрахунку використовують дві гіпотези: нульова гіпотеза H_0 та альтернативна гіпотеза H_1.

Формула нульової гіпотези:

$$t = \frac{|r| \cdot \sqrt{n-2}}{\sqrt{1-r^2}}, \quad (4.2)$$

де r – вибірковий коефіцієнт кореляції Пірсона;

n – кількість спостережень у вибірці.

Формула визначення ступеню свободи:

$$df = n - 2. \quad (4.3)$$

Використовуючи таблиці розподілу Ст'юдента (рисунок 4.1) визначається критичне значення $t_{кр}$ на основі обраного рівня значущості α та ступеня свободи. Якщо:

– $|t| < t_{кр}$ то нульова гіпотеза приймається, коефіцієнт кореляції не є значимим, між досліджуваними ознаками не існує лінійного кореляційного зв'язку;

– $|t| > t_{кр}$ то нульова гіпотеза відхиляється, приймається конкуруюча гіпотеза, коефіцієнт кореляції є значимим, між досліджуваними ознаками існує лінійний кореляційний зв'язок.

Критерій Ст'юдента *t*

Число ступенів свободи f	Рівень значимості α			
	0,10	0,05	0,01	0,001
1	6,31	12,70	63,70	637,00
2	2,92	4,30	9,92	31,60
3	2,35	3,18	5,84	12,90
4	2,13	2,78	4,60	8,61
5	2,01	2,57	4,03	6,86
6	1,94	2,45	3,71	5,96
7	1,89	2,36	3,50	5,40
8	1,86	2,31	3,36	5,04
9	1,83	2,26	3,25	4,78
10	1,81	2,23	3,17	4,59
11	1,80	2,20	3,11	4,44
12	1,78	2,18	3,05	4,32
13	1,77	2,16	3,01	4,22
14	1,76	2,14	2,98	4,14
15	1,75	2,13	2,95	4,07
16	1,75	2,12	2,92	4,01
17	1,74	2,11	2,90	3,96
18	1,73	2,10	2,88	3,92
19	1,73	2,09	2,86	3,88
20	1,73	2,09	2,85	3,85
21	1,72	2,08	2,83	3,82
22	1,72	2,07	2,82	3,79
23	1,71	2,07	2,81	3,77
24	1,71	2,06	2,80	3,74
25	1,71	2,06	2,79	3,72
26	1,71	2,06	2,78	3,71
27	1,71	2,05	2,77	3,69
28	1,70	2,05	2,76	3,66
29	1,70	2,05	2,76	3,66
30	1,70	2,04	2,75	3,65
40	1,68	2,02	2,70	3,55
60	1,67	2,00	2,66	3,46
120	1,66	1,98	2,62	3,37
∞	1,64	1,96	2,58	3,29

Рисунок 4.1 – Таблиця розподілу Ст'юдента

Якщо кореляційний зв'язок виявився суттєвим, то доцільно скористатися методами регресійного аналізу, основне завдання якого полягає у визначенні

характеру зв'язку і побудові його математичної моделі. На основі моделі можна передбачити ту або іншу подію, спрогнозувати, як будуть розвиватися певні процеси у разі змінення характеристик об'єкта дослідження. Лінійна регресія є найпростішою формою регресійного аналізу, яка шукає лінійний зв'язок між змінними [29]. Вона визначається рівнянням прямої:

$$y = \beta_0 + \beta_1 x + \epsilon, \tag{4.4}$$

де y – залежна змінна, відгук;

x – незалежна змінна, фактор;

β_0 – вільний член, перетин з віссю y;

β_1 – коефіцієнт нахилу, що відображає змінну y при зміні x на одиницю;

ϵ – випадкова похибка, шум.

Формула розрахунку вільного члена β_0:

$$\beta_0 = \bar{y} - \beta_1 \bar{x}, \tag{4.5}$$

де $\bar{y}$ – середнє значення всіх залежних змінних;

$\bar{x}$ – середнє значення всіх незалежних змінних.

Формула розрахунку коефіцієнту нахилу:

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}. \tag{4.6}$$

Формула розрахунку прогнозованого значення

$$\hat{y} = \beta_0 + \beta_1 x. \tag{4.7}$$

4.2 Проведення розрахунків

Усі розрахунки проводилися з використанням бібліотек візуалізації та аналізу даних pandas, scipy, seaborn, matplotlib.pyplot для мови програмування Python. Розрахунки проводилися для результатів вимірів часу знаходження для алгоритмів з таблиці 3.2 та результатів виміру алгоритмів розпізнавання з таблиці 3.6.

Вхідні дані для таблиці 3.2:

```
# Додавання даних
data = {
   'Алгоритм': ['CascadeClassifier', 'CascadeClassifier', 'CascadeClassifier', 'CascadeClassifier',
                'CascadeClassifier', 'CascadeClassifier', 'Facelocation', 'Facelocation',
                'Facelocation', 'Facelocation', 'Facelocation', 'Facelocation'],
   'Роздільна здатність': [320, 800, 1280, 320, 800, 1280,
                           320, 800, 1280, 320, 800, 1280],
   'Освітленість': [300, 300, 300, 65, 65, 65, 300, 300, 300, 65, 65, 65],
   'Точність': [75, 80, 85, 8, 41, 24, 98, 98, 98, 65, 73, 81]
}
```

Код розрахунку кореляційної матриці:

```
# Кореляційна матриця (лише для числових даних)
correlation_matrix = df[['Освітленість', 'Роздільна здатність', 'Точність']].corr()
print("\nКореляційна матриця (тільки числові дані):")
print(correlation_matrix)
# Графік кореляцій
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', square=True)
plt.title('Кореляційна матриця (тільки числові дані)')
plt.show()
```

На рисунку 4.2 показана розрахована кореляційна матриця. Згідно матриці ми бачимо сильну кореляційну залежність точність та освітлення і доволі слабку кореляцію між точністю та роздільною здатністю.

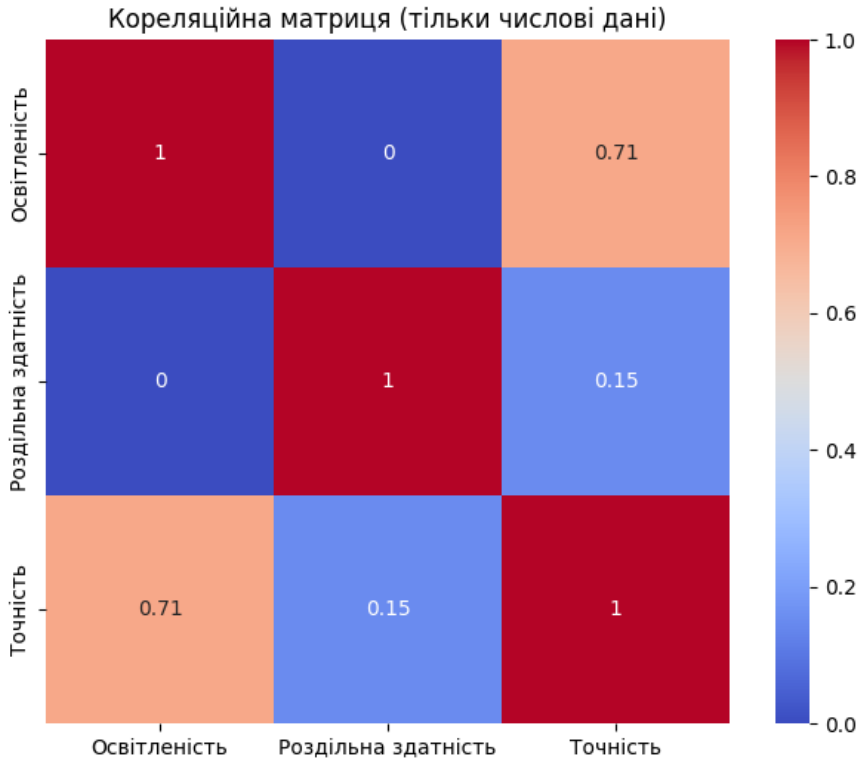

Рисунок 4.2 – Кореляційна матриця для таблиці 3.2

Розрахунок кореляції Пірсона і значення t:

```
# Розрахунок кореляції Пірсона та t -значення
pearson_corr, p_value = stats.pearsonr(df['Освітленість'], df['Точність'])
print(f"\nКореляція Пірсона: {pearson_corr}")
print(f" t -значення: {p_value}")
```

Кореляція Пірсона дорівнює 0,7114570487658045.

t –значення дорівнює 0,009466748654414252

Код лінійної регресії:

```
# Прогнозування та візуалізація лінійної регресії
predicted = pipeline.predict(X)
plt.figure(figsize=(10, 6))
plt.scatter(range(len(y)), y, color='blue', label='Реальні дані')
plt.plot(range(len(y)), predicted, color='red', label='Прогнозовані дані')
plt.title('Прогнозування точності')
plt.xlabel('Індекс спостереження')
plt.ylabel('Точність розпізнавання')
plt.legend()
plt.show()14252
```

На рисунку 4.3 показаний графік лінійної регресії.

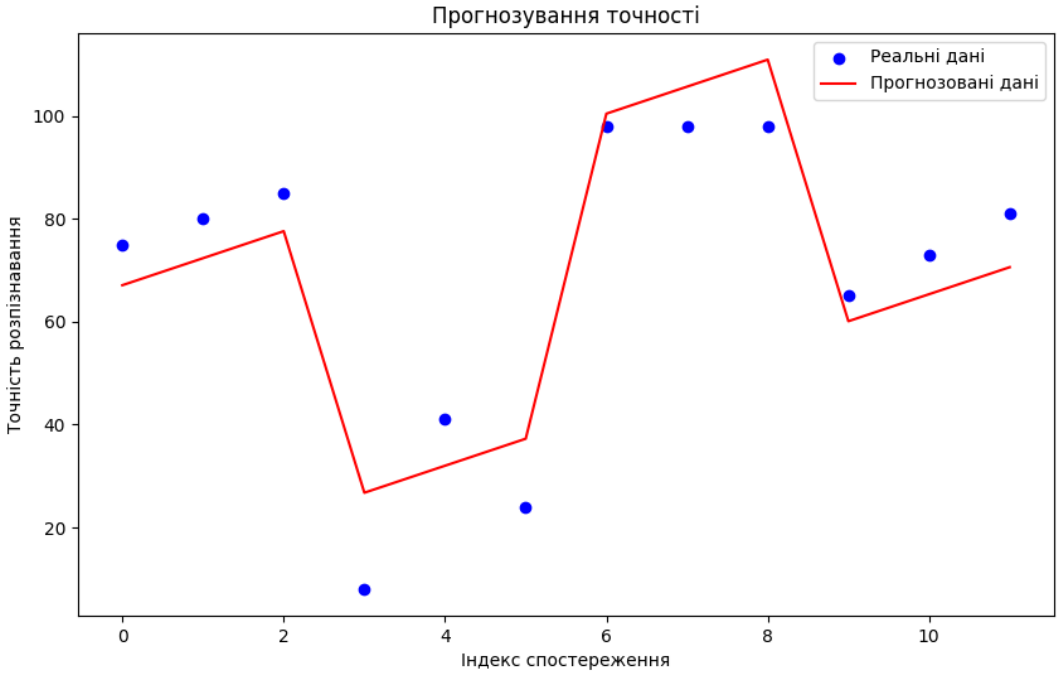

Рисунок 4.3 – Графік лінійної регресії

На рисунку 4.4 показаний загальний графік точності розпізнавання.

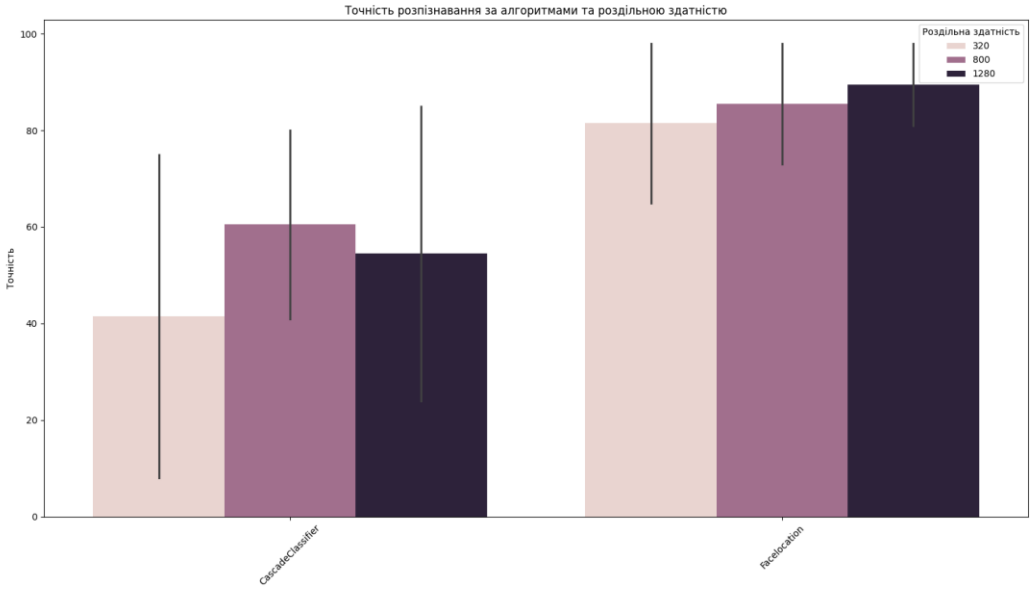

Рисунок 4.4 – Загальний графік точності розпізнавання за алгоритмами та роздільною здатністю

Вхідні данні для таблиці 3.6:

```
# Додавання даних
data = {
    'Алгоритм': ['LBPH', 'Eigenfaces', 'Fisherfaces', 'FaceRecognition',
                 'LBPH', 'Eigenfaces', 'Fisherfaces', 'FaceRecognition'],
    'Освітленість': [300, 300, 300, 300, 65, 65, 65, 65],
    'Час_розпізнавання': [44, 125, 105, 0.07, 51, 130, 110, 0.08],
    'Точність': [70, 15, 55, 98, 0, 0, 0, 95]
}
```

За схожим принципом було проведено розрахунки для значень таблиці 3.6.

На рисунку 4.5 показана отримана кореляційна матриця, згідно якої ми бачимо сильну кореляцію між освітленістю та точністю.

На рисунку 4.6 показана отримана лінійна регресія.

Кореляція Пірсона для значень таблиці 3.6 дорівнює 0,4448314708926203. t-значення дорівнює 0,26943585928751446.

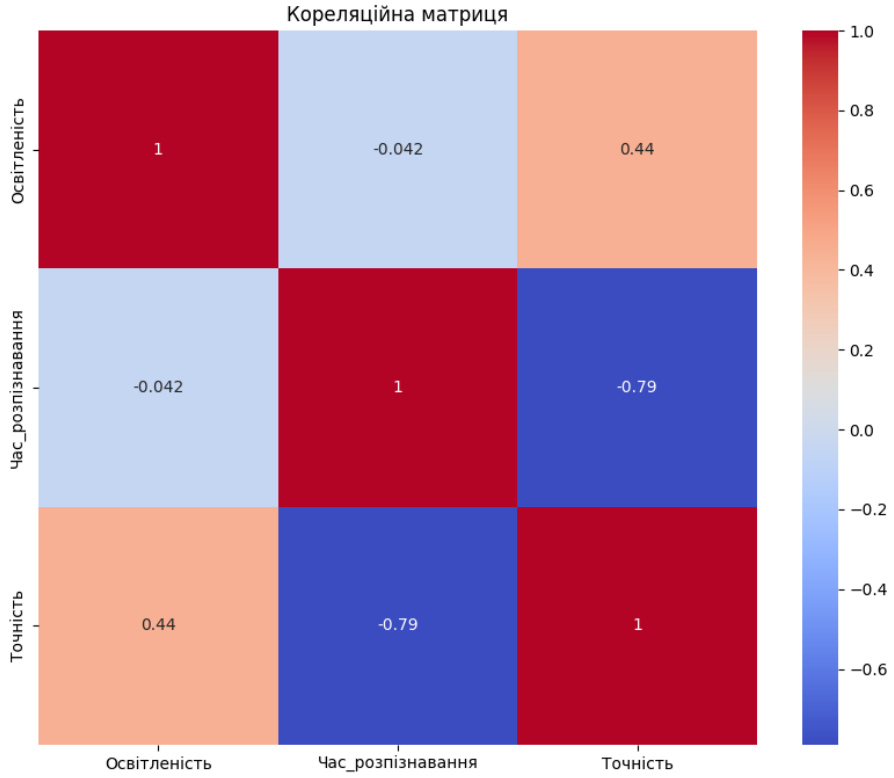

Рисунок 4.5 – Кореляційна матриця значень таблиці 3.6

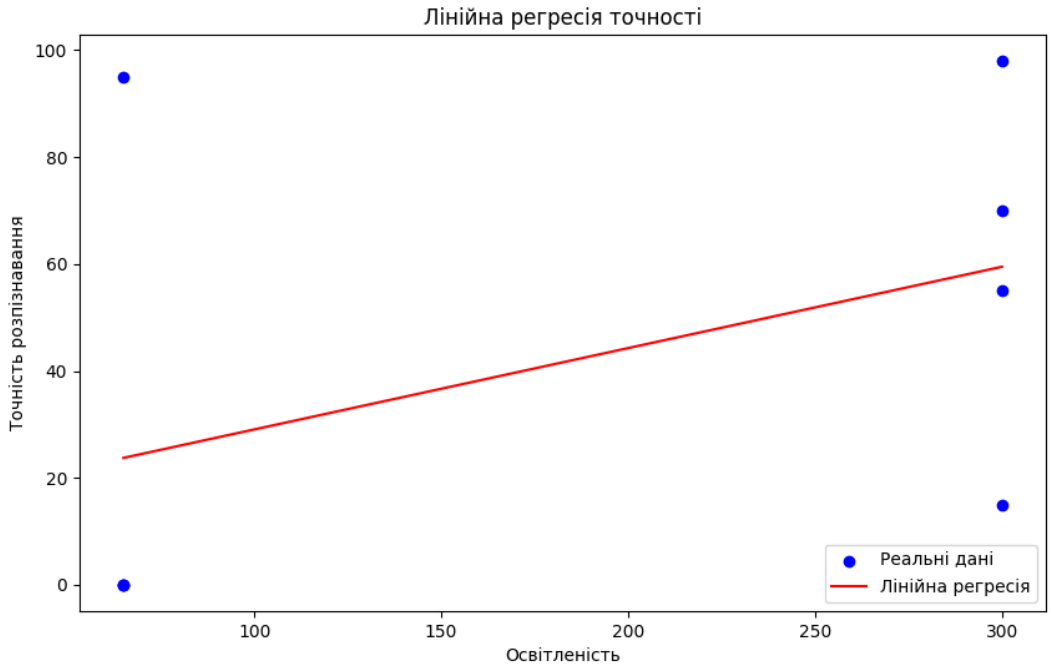

Рисунок 4.6 – Лінійна регресія значень таблиці 3.6

4.3 Висновки до розрахункової частини

Розрахункова частина роботи підтвердила ефективність розробленої системи для розпізнавання обличчя. Проедені розрахунки показали високу точність систем на базі OpenCV та dlib, і незадовільні результати для системи, що використовую лише модуль ESP32-cam. Згідно результатів модуль має проблеми з продуктивністю та не спроможний гарантувати однакову точність в різних умовах освітлення.

Для систем на базі OpenCV, розрахунки показали високу кореляцію між освітленням і точністю. В гарних умовах така система може запропонувати задовільну точність у поєднанні з швидкістю та простотою реалізації. Але при погіршенній освітлення чи зміні кута нахилу голови точність значно падає і є доволі складно прогнозованою.

Система на базі бібліотеки dlib, показала найкращі результати за результатами розрахукнів. Вона спроможна забезпечити велику точність розпізнавання навіть в найгірших умовах освітлення, але є доволі повільною і залежною від потужності системи.

Таким чином, розрахункова частина підтверджує доцільність використання системи на базі dlib, для подальшої розробки у сфері забезпечення безпеки до виробничого приміщення.

5 ОХОРОНА ПРАЦІ

Охорона праці є невід'ємною частиною будь-якої діяльності, пов'язаної з електронікою, особливо при роботі з електропристроями та складанням мікросхем. Нехтування правилами безпеки може призвести до пошкодження обладнання та отримання травм різного ступеня тяжкості. З метою уникнення подібних проблем, перед початком роботи необхідно ознайомитися з правилами безпеки та неухильно їх дотримуватися [30].

Перед початком будь-якої роботи з електропристроями, необхідно провести ретельну перевірку всього обладнання. Це включає в себе візуальний огляд паяльника, мультиметра, блоку живлення та інших пристроїв на наявність пошкоджень. Особливу увагу слід приділити перевірці цілісності ізоляції. Її пошкодження може призвести до короткого замикання або ураження електричним струмом.

Мікросхеми дуже чутливі до статичної електрики. Її накопичення може привести до ураження струмом працівника чи знищення мікросхеми. Для запобігання цьому дуже важливо використовувати антистатичні килимки на робочих місця, та уникати одягу що виробляє статичну електрику шляхом тертя. Важливо використовувати антистатичного браслета працівниками, що займаються збіркою електронних компонентів.

Мікросхеми є крихкими компонентами, тому з ними потрібно поводитися обережно. Не можна згинати виводи мікросхем чи прикладайте надмірних зусиль під час встановлення мікросхем в панельки або на плату.

Під час роботи з паяльними засобами існує ризик виникнення пожежі. З метою уникнення подібного інциденту, на кожному робочому місці повинен бути ручний вогнегасник. Рекомендовано використовувати паяльники з функцією авто-вимкнення на випадок якщо працівник забув це зробити. Кожне приміщення повинне бути обладнане датчиками детекції диму та відкритого вогню. Необхідні автоматичні системи пожежогасіння.

В кожному приміщенні повинна бути аптечка першої допомоги, а працівники повинні проходити щоквартальні курси з надання першої допомоги.

Правильне освітлення робочого місця є важливим для запобігання напрузі очей та втомі під час роботи за персональним комп’ютером чи малими деталями мікросхем. Робоче місце повинне бути забезпечене достатнім освітленням та мати можливість регулювання яскравості та напряму освітлення.

Не менш важливим аспектом, як особистої, так і загальної безпеки, є використання ліцензійного програмного забезпечення. Його використання дозволить уникнути проблем із авторськими правами та забезпечить безпеку комп’ютера від злому системи та викрадення персональних даних.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено аналіз предметної області, розглянуто основні методи фільтрації зображення та сучасні методи розпізнавання обличчя. Було досягнуто поставленої мети – розроблена система розумного доступу до виробничого приміщення, із використання технологій комп'ютерного зору, що мінімізує потребу у фізичному контролі та підвищує рівень безпеки на об'єкті.

З ціллю досягнення мети у другому розділі, з використанням OpenCV та бібліотеки dlib для мови програмування Python, було розроблено декілька систем розпізнавання обличчя людини, що працюють на відрізних один від одного принципах. Були описані вимоги до системи, на основі аналізу пропозицій конкурентів, на базі яких була розроблена схема загального принципу роботи.

Проведені експериментальні дослідження, та подальший аналіз результатів, продемонстрував підвищенні вимоги до освітлення, в таких систем. Була зафіксована пряма залежність між освітленням та точністю розпізнавання. Також експериментальні дослідження показали не спроможність модулю ESP32-cam вирішувати задачі забезпечення безпеки на виробництві. Погана камера, мала потужність, значні вимоги до навичок програміста, роблять системи на його основі повільними та не ефективними.

На основі проведених досліджень, можна сказати що, розроблена система із CNN-мережою на базі бібліотеки dlib, забезпечує найвищий рівень точності, а отже і безпеки. Розроблена система не потребує попереднього глибокого та довгого навчання, на відміну від Eigenfaces та Fisherfaces.

Отримані результати кваліфікаційної роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме 9.4 [5].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008-15. Документація. Звіти у сфері науки та техніки. структура та правила оформлення. Введ. 2015-06-22. К. Держстандарт України, 2017. 29 с.

2. Невлюдов, І.Ш. Дипломне проектування для студентів усіх форм навчання спеціальностей 151 «Автоматизація та комп'ютерно-інтегровані технології» [Текст]: навч. посіб. / І.Ш. Невлюдов, А.О. Андрусевич, О.В. Токарєва, Г.В. Пономарьова. – Київ-58, пр. Космонавта Комарова, 1, 2016. – 320с.

3. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

4. Водяницький М. А. Розробка системи розумного доступу до виробничого приміщення з використанням технологій комп'ютерного зору/ М. А. Водяницький // Автоматизація та Приладобудування = Automation and Development of ElectronicDevices (ADED'2024) : збірник студентських наукових статей. – Харків : ХНУРЕ, 2024. – Вип. 1. – С. 147-151.

5 Ціль 9. Промисловість, інновації та інфраструктура // Сайт Diia business URL: https://business.diia.gov.ua/entrepreneur-handbook/item/cil_9_promislovist_innovaciyi_ta_infrastruktura

6. What is computer vision? // Сайт IBM.com URL: https://www.ibm.com/topics/computer-vision.

7. Основи теорії розпізнавання образів, навчальний посібник у двох частинах, А.С. Довбиш, І.В. Шелехов // Сайт НТУ Національний транспортний університет URL: http://kist.ntu.edu.ua/textPhD/tro2.pdf.

8. Теорія розпізнавання образів, навчальний посібник, В.В. Гавриленко, Г.Ф. Іванченко, Є.Г. Шевченко // Сайт studfile URL: https://studfile.net/preview/5607072/.

9. What is Thresholding in Image Processing? A Guide. // Сайт Roboflow Blog URL: https://blog.roboflow.com/image-thresholding/.

10. Computer Vision (Part 25) – Gaussian Filter | by Coursesteach // Сайт Medium URL: https://medium.com/@Coursesteach/computer-vision-part-25-gaussian-filter-c81ea05a4630.

11. Through The Eyes of Gabor Filter // Сайт Medium URL: https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97.

12. Wavelet // Сайт Wikipedia URL: https://en.wikipedia.org/wiki/Wavelet.

13. Canny Edge Detection // Сайт OpenCV Documentation URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

14. Face Recognition Techniques - A Review // Сайт Ajay Kumar Garg Engineering College URL: https://www.akgec.ac.in/wp-content/uploads/2019/06/2-Face-Recognition-Techniques-Bhaskar-Gupta.pdf.

15. Face Recognition for Beginners // Сайт Medium URL: https://towardsdatascience.com/face-recognition-for-beginners-a7a9bd5eb5c2.

16. Face Recognition: From Traditional to Deep Learning Methods // Сайт Cornell University URL: https://arxiv.org/pdf/1811.00116.

17. Convolution Neural Network (CNN) // Сайт Analytics Vidhya URL: https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-networkcnn/.

18. PyCharm // Сайт Wiki URL: https://en.wikipedia.org/wiki/PyCharm.

19. Arduino // Сайт Wiki URL: https://en.wikipedia.org/wiki/Arduino.

20. ESP-WHO // Сайт GitHub URL: https://github.com/espressif/esp-who.

21. M5Stack Camera // Сайт GitHub URL: https://github.com/m5stack/M5Stack-Camera/blob/master/face_qr/components/esp-face/face_detection/README.md.

22. Yoursunny esp32cam // Сайт GitHub URL: https://github.com/yoursunny/esp32cam.

23. Face Recognition with OpenCV // Сайт OpenCV Documentation URL: https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html.

24. face-recognition // Сайт PyPl URL: https://pypi.org/project/face-recognition/.

25. Додатки в Google Play – DroidCam Webcam (Classic) // Сайт Google Play URL: https://play.google.com/store/apps/details?id=com.dev47apps.droidcam&hl=ru.

26. Світломір – Додатки в Google Play // Сайт Google Play URL: https://play.google.com/store/apps/details?id=com.lux.light.meter&hl=uk.

27. Multivariate statistics // Сайт Wiki URL: https://en.wikipedia.org/wiki /Multivariate_statistics

28. Кореляційний аналіз // Сайт Wiki ТНТУ URL: https://wiki.tntu.edu.ua /Кореляційний_аналіз.

29. Основи кореляційного та регресійного аналізу // Сайт Табличний процесор MS Excel https://pchilka-litsei.in.ua/excel-book/basis_analysis.html.

30. Методичні вказівки до виконання розділу "Охорона праці" у випускних роботах ОКР "бакалавр" усіх форм навчання / упоряд.: В. А. Айвазов. Т. Є. Стиценко., Н. Л. Березуцька ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2018. – 28 с. – 1,81.