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

Бакалаврська дипломна робота на тему:

«СПЕЦІАЛІЗОВАНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХОПЛЕННЯ І СТЕЖЕННЯ ЗА ОБЛИЧЧЯМ КОРИСТУВАЧА У РЕЖИМІ РЕАЛЬНОГО ЧАСУ»

Виконав: студент 4 курсу, групи 2СП-20б
спеціальності123 — «Комп'ютерна інженерія»
освітня програма — «Системне програмування»

_____ Федоренко М. С.

Керівник: ст. викл. каф. ОТ

_____ Кисюк Д. В.

«16» 06 2024 р.

Рецензент: к.ф-м.н., доцент кафедри МБІС

_____ Шиян А. А.

«19» 06 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«20» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти І-й (бакалаврський)
Галузь знань —12 — Інформаційні технології
Спеціальність —123 — «Комп'ютерна інженерія»
Освітня програма — «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О.Д. Азаров
«06» 02 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Федоренку Максиму Сергійовичу

1 Тема роботи «Спеціалізоване програмне забезпечення для захоплення і стеження за обличчям користувача у режимі реального часу», керівник роботи Кисюк Дмитро Васильович, ст. викл. каф. ОТ ,затверджені наказом вищого навчального закладу від 20.03. 2024 р. № 67.

2 Строк подання студентом роботи 15.06.2024 р.

3 Вихідні дані до роботи: частота надходження кадрів відеопотоку не менше 15 кадрів/сек, розмір зображення — не менше 240 х 240 пікселів, модель кольорів для представлення зображення — RGB, кількість градацій яскравості зображення — 256.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз методів та засобів розпізнавання облич, аналіз задачі розпізнавання облич у відеопотоках, розробка системи реєстрації на основі аналізу відеозображень, розробка методики проведення експеременту та

та тестування, аналіз отриманних результатів і рекомендації щодо їхнього розвитку, висновки, список використаних джерел, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): методи виділення рухомих об'єктів, послідовність виділення рухомих об'єктів, типи згорткових нейронних мереж, схема програми відстеження пересування людини.

6 Консультанти розділів роботи див. табл. 1

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Кисюк Д. В., стр. викл. каф. ОТ		

Д

ат

а видачі завдання 08.03.2024 р.

8 Календарний план (див. табл. 2).

Таблиця 2 — Календарний план

№	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	08.03.2024	Викон.
2	Огляд та аналіз сучасних підсистем забезпечення захоплення і стеження за обличчям користувача	08.03.2024 – 15.03.2024	Викон.
3	Проведення порівняльного аналізу аналогів підсистем захоплення і стеження за обличчям користувача	16.03.2024 – 23.04.2024	Викон.
4	Розробка структури програмного засобу, вибір платформи, інструментів розробки та проектування	24.03.2024 – 21.04.2024	Викон.
5	Програмна реалізація функціональних модулів: виявлення облич, стеження за обличчям та збереження результатів	22.04.2024 – 05.05.2024	Викон.
6	Підготовка матеріалів для опису проектування програмного засобу	06.05.2024 – 12.05.2024	Викон.
7	Оформлення пояснювальної записки та ілюстративного матеріалу	13.05.2024 – 18.05.2024	Викон.
8	Перевірка якості виконання дипломної роботи	19.05.2024 – 20.05.2024	Викон

Студент _____ Федоренко М. С.

Керівник роботи _____ Кисюк Д. В.

АНОТАЦІЯ

УДК 004.93

Федоренко М. С. Спеціалізоване програмне забезпечення для захоплення і стеження за обличчям користувача у режимі реального часу. Бакалаврська дипломна робота зі спеціальності 123 — Комп’ютерна інженерія, освітня програма — Системне програмування. Вінниця: ВНТУ, 2024. 100 с.

На укр. мові. Бібліогр.:26 назв; рис.: 3.

У бакалаврській дипломній роботі розроблено спеціалізоване програмне забезпечення для захоплення і стеження за обличчям користувача у режимі реального часу. У роботі зроблено аналіз методів виявлення облич у відеопотоці, запропоновано здійснювати процес виділення облич із використанням методу каскадних класифікаторів Хаара, а розпізнавання типу облич із застосуванням згорткової нейронної мережі. В роботі розроблено послідовність обробки відео зображення для виявлення та стеження за обличчям користувача та програмне забезпечення, яке реалізує цей процес у режимі реального часу.

Ключові слова: каскадні класифікатори Хаара, відеоспостереження, стеження за обличчям, згорткова нейронна мережа.

ANNOTATION

Fedorenko M. S. Specialized software for capturing and tracking the user's face in real time. Bachelor's thesis in specialty 123 -- Computer engineering, educational program -- System programming. Vinnytsia: VNTU, 2024. 100 c.

In Ukrainian. Bibliogr.: 26 titles; Figures: 3.

In the bachelor's thesis, specialized software for capturing and tracking the user's face in real time was developed. The work analyzes methods of face detection in a video stream, proposes to perform face extraction using the method of cascade Haar classifiers, and face type recognition using a convolutional neural network. The paper develops a video image processing sequence for detecting and tracking the user's face and software that implements this process in real time.

Keywords: cascade Haar classifiers, video surveillance, face tracking, convolutional neural network.

ЗМІСТ

ВСТУП

Актуальність даного дослідження зумовлена зростаючою потребою у високоякісних системах для захоплення і стеження за обличчям користувача в режимі реального часу. Сучасні технології ідентифікації та верифікації особистості стають все більш важливими у різних сферах, включаючи безпеку, охорону здоров'я, розваги та маркетинг. Зокрема, в системах безпеки обличчяві технології використовуються для контролю доступу, моніторингу територій та виявлення підозрілої поведінки.

Метою роботи є розробка системи автоматичного розпізнавання облич, яка забезпечує високу точність і швидкість обробки зображень і відео в режимі реального часу.

Завдання:

— проаналізувати сучасні методи і алгоритми розпізнавання облич;

— вибрати найефективніші методи для детектування та розпізнавання облич;

— розробити програмне забезпечення для захоплення та стеження за обличчям користувача;

— провести тестування і порівняння ефективності вибраних методів;

— оцінити практичну цінність і можливість застосування розробленої системи у різних сферах.

Розпізнавання облич — це практичне застосування теорії розпізнавання образів, яке полягає в автоматичному виявленні облич на зображеннях та, за потреби, ідентифікації особи. Останнім часом інтерес до цієї технології значно зріс через кілька причин, таких як зростаюча стурбованість щодо безпеки, необхідність ідентифікації особистості у цифровому середовищі та попит на аналіз облич у мультимедійних даних та комп'ютерних розвагах.

За останні роки в цій сфері було досягнуто значного прогресу завдяки вдосконаленню методів моделювання та аналізу облич. Було розроблено системи для виявлення та відстеження облич, що дозволило вирішувати складні завдання комп'ютерного зору та розпізнавання образів.

Основне завдання дослідження полягає в автоматичному розпізнаванні облич на зображеннях та відео. Перевірка обличчя включає порівняння "один до одного", коли зображення обличчя запиту зіставляється із зображенням обличчя шаблону. Ідентифікація обличчя передбачає порівняння "один до багатьох", коли зображення обличчя запиту зіставляється з усіма зображеннями у базі даних для визначення особи.

Об'єктом дослідження є відеоаналітика, що має велике значення та поширення у сучасному світі. Відеоаналітика — це сучасна технологія, яка включає методи комп'ютерного зору для автоматизованого збору інформації з послідовності кадрів, отриманих у реальному часі або з відеозаписів.

Предметом дослідження є основне завдання відеоаналітики — детектування та розпізнавання облич у відеопотоках. Цей метод ефективний при різних умовах освітлення, але не завжди точно визначає марковані точки.

Практична цінність цієї роботи полягає в розробці системи реєстрації на основі аналізу відеозображень та вирішенні цього завдання за допомогою різних методів комп'ютерного зору. Перша частина роботи присвячена огляду методів, їх порівнянню та вибору найефективнішого методу для детектування та розпізнавання облич. Далі проведено аналіз завдання та визначено умови для реалізації системи. Наступним етапом описано основні етапи роботи системи, включаючи детектування та розпізнавання облич у кадрах відеопотоку з використанням різних методів. Після цього було проведено дослідження ефективності методів та порівняння результатів. Потім обрано інструменти розробки та описано програмну реалізацію системи детектування та розпізнавання облич. На завершення проведено тестування ефективності системи та складено результати й висновки.

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ ОБЛИЧ

Виділення облич користувачів, які змінюють своє положення у межах сцени отриманого відео зображення, ґрунтується на виявленні окремих пікселів або груп пікселів, що за вибраними параметрами відрізняються на сусідніх кадрах відеопотоку. Розгляду та аналізу існуючих методів виявлення та стеження за обличчям користувачів у режимі реального часу присвячений даний розділ дипломної роботи.

1.1 Методи виявлення облич (Face Detection)

Метод Віоли-Джонса спосіб, що дозволяє демонструвати предмети у відображеннях у теперішньому періоді. Запропонований у 2001 р. Paul Viola і Michael Jones. [3] Метод розроблявся для виявлення облич, але згодом з'ясувалося, що він включає ще можливість розрізняти різноманітні класи зображень.

Навчання класифікаторів повільне, проте результати виявлення обличчя швидкі. Під час роботи детектора можливість неправильного виявлення обличчя низька. Відмінні результати розпізнавання виходять за умови, що кут нахилу буде приблизно до 30 градусів. Якщо ж кут нахилу буде більше 30 градусів, то відсоток виявлень низький. За кута нахилу понад 30 градусів відсоток виявлень різко падає [11].

Переваги методу Face Detection:

— найпоширеніший метод виявлення облич;

— використання каскадного класифікатора значно підвищує швидкість;

— висока точність виявлення повернених облич на кут до 30 градусів (якщо кут більший, ефективність цього методу низька).

Недоліки методу Face Detection:

— тривалй час навчання і необхідність аналізу великої кількості тестових зображень (великий розмір навчальної вибірки);

— у процесі розпізнавання на положення обличчя є обмеження.

Метод AdaBoost (Adaptive Boosting) — це метод посилення класифікаторів методом об'єднання їх у комітет. Уперше був запропонований Йіавом Фройндом і Робертом Шапіре в 1999 році. [7]. Вживається в поєднанні з кількома методами систематизації для поліпшення їхньої продуктивності. Метод є адаптивним, що означає, що кожен подальший комітет класифікаторів базується на відповідності об'єктам, невірно класифікованим минулими комітетами.

Переваги методу Adaptive Boosting:

— алгоритм адаптується під проблемні елементи навчальної вибірки;

— величезна швидкість роботи;

— нескладна реалізація.

Недоліки методу Adaptive Boosting:

— висока чутливість до шумів і викидів даних;

— витрати часу на навчання залежать від кількості класифікаторів і розміру навчальної вибірки.

Метод SNoW (Sparse Network of Winnows) — алгоритм виявлення облич, що є двошаровою мережею, вхідний шар якої складається з кутів, кожен з яких відповідає деякій характеристиці вхідного зображення, а вихідний — лише з двох вузлів, кожен з яких відповідає класам зображень, що розпізнаються [5]. Як ознаки в цьому алгоритмі використовуються SMQT (Successive Mean Quantization Transform) ознаки. Це перетворення дає змогу витягти з локальної області зображення складову, що не залежить від освітленості. Воно полягає в квантуванні області зображення з порогом квантування, що дорівнює середньому значенню пікселів, що входять у цю область [6].

Переваги методу Sparse Network of Winnows:

— зміни освітлення не мають впливу в локальних областях зображень;

— точність виявлення відмінна при використанні просіювання вектора.

Недоліки методу Sparse Network of Winnows:

— висока чутливість до шумів і викидів даних.

Метод нейромережеві способи — які складаються з цілого класу алгоритмів і перетворюють поступово сигнали паралельно працюючими нейронами. Процес навчання нейронних мереж полягає у зменшенні середньоквадратичної помилки. Усі системи виявлення об'єктів на зображеннях, які засновані на нейронних мережах, мають ієрархічну структуру. Спочатку вектор ознак обробляється грубою мережею з високим рівнем помилок другого роду, далі, якщо вектор не був класифікований як не об'єкт, рішення коригується більш точною і більш повільною мережею[7].

Особливістю нейронних мереж є їх здатність до навчання на наборі готових прикладів, заздалегідь занесених у базу даних. Мережа автоматично витягує ключові ознаки і будує взаємозв'язок між ними. Надалі, для того щоб розпізнати раніше невідомий об'єкт, навчена нейронна мережа застосовує отриманий досвід. Нейромережеві методи показують чудові результати в галузі розпізнавання, але вважаються найскладнішими для реалізації.

Переваги методу:

— точність виявлення залежить від налаштування параметрів мережі.

Недоліки методу:

— потрібне ретельне налаштування параметрів нейронної мережі (кількість нейронів, шарів, характер зв'язків);

— висока чутливість до шумів;

— складна процедура внесення змін (внесення будь-якої зміни вимагає перенавчання мережі);

— висока обчислювальна складність.

Метод Support vector machine — метод опорних векторів. Головне завдання методу опорних векторів — це знаходження гіперплощини в ознаковому просторі, яка відокремлюватиме клас зображень облич від

зображень «не-осіб». Після вибору гіперплощин нам необхідно вибрати з них таку гіперплощину, де відстань до неї від кожного класу максимальна. Ця гіперплощина називається оптимальною розділювальною гіперплощиною, а відповідний їй лінійний класифікатор називається оптимально розділювальним класифікатором [8].

Переваги методу Support vector machine:

— найвища стійкість до перенавчання;

— порівняно з нейронними мережами швидкість роботи набагато краща;

— знижуючи точність можна зменшити чутливість до шуму.

Недоліки методу Support vector machine — недостатня точність роботи порівняно з іншими методами (AdaBoost і SNoW).

Порівняльна таблиця 1.1 [9] показує ефективність методів виявлення облич.

Таблиця 1.1 — Порівняння ефективності методів виявлення облич

Метод виявлення облич	Відсоток вірних виявлень облич	Помилка другого роду
Нейроенні мережі	~92%	~1.3%
Метод опорних векторів	~72%	~0.6%
SNoW	~94%	~0.12%
Метод Віоли – Джонса	~94%	~0.00001%

Результати таблиці 1.1. за показниками відсотка вірних виявлень і помилки другого роду є метод Віоли-Джонса.

1.2 Методи, засновані на значеннях яскравості пікселів

Ці методи ґрунтуються на значеннях пікселів і для розпізнавання облич використовують кольори та яскравість. Для цього використовують зіставлення, і міра подібності визначається відстанню між векторами яскравостей пікселів зображень. Слабкою стороною цих методів є нестійкість до змін освітлення,

різних розташувань обличчя, масштабування. Ці методи також мають складність в обчисленні і досить складне їх застосування в реальному часі. Тому, дуже часто використовують методи, що використовують перехід векторного опису зображень у просторі з меншою розмірністю, в яких порівняння набагато ефективніше[10].

Метод Eigenfaces — алгоритм, запропонований 1991 року Метью Терком і Алексом Пентландом [11]. Є одним із перших вдалих методів розпізнавання облич, який здобув величезну популярність. Основне завдання методу — це знаходження векторів, що описують зображення облич за допомогою методу головних компонент. Метод дає змогу виявити різні мінливості в навчальній вибірці зображень облич і описати їх у базисі кількох ортогональних векторів, які називаються власними (Eigenfaces).

Комплект особистих векторів, отриманих у результаті навчальної вибірки, використовують для шифрування всіх інших зображень облич, які представляють як зважену композицію цих особистих векторів. Використовуючи обмежену кількість власних векторів, можна отримати стислу апроксимацію вхідного зображення обличчя, яку потім можна зберігати в базі даних у вигляді вектора коефіцієнтів, що слугує водночас ключем пошуку в базі даних осіб[11]. До переваг цього методу можна віднести простоту реалізації, придатність для розпізнавання в реальному часі та можливість компактно зберігати великі обсяги даних[10].

Відмінні характеристики зображення обличчя показуються за допомогою двовимірних зображень у градаціях сірого. Для розпізнавання обличчя, отриманого з кадру, роблять порівняння з іншим зареєстрованим шаблоном у базі та далі розраховують коефіцієнт відмінності, який визначає ступінь схожості. Для отримання хороших результатів методу EigenFaces необхідні хороше освітлення і сканування обличчя у фас. Якщо таких умов немає, то ефективність розпізнавання буде низькою.

Цю проблему спричинено тим, що найважливіші власні вектори більшою мірою описують особливості освітлення, ніж характеристики облич, оскільки

спочатку метод головних компонент вибирає підпростір із метою апроксимації даних, а не їхньої класифікації [10].

Переваги методу:

— нескладна реалізація;

— можливість використання в реальному часі;

— компактне зберігання великих обсягів даних Недоліки методу;

— велика чутливість до зміни умов (освітлення, розміри, повороти);

— суворе збереження початкових умов зйомки.

Метод Fisherfaces — метод має здатність більш високої точності розпізнавання при змінах умов освітлення або виразу обличчя. Метод Fisherfaces використовує лінійний дискримінант Фішера. Алгоритм працює на пошуку проекції даних, за якої класи зображень облич максимально розділені, що допомагає вирішити проблему чутливості до змін освітлення. При використанні ж методу головних компонент проводиться максимізація розкиду даних по всій базі облич. Ця відмінність дає змогу розв'язати проблему високої чутливості до змін освітлення[12].

Метод Локальні бінарні шаблони (LBP - Local Binary Pattern) — простий і ефективний оператор перетворення зображень, уперше запропонований 1996 року для класифікації текстур[13]. Однак, пізніше знайшов застосування і для розпізнавання облич[14].

Під час розпізнавання методом LBP, кожному пікселю зображення з використанням функції присвоюється значення яскравості, що описує його околицю. Далі, отримане зображення підрозділяється на області і для кожної підраховується гістограма. Потім ці гістограми конкатенуються і порівнюються за підтримки методів машинного, в класичному варіанті вживається метод найближчого сусіда.

До переваг цього методу належить нескладність реалізації та велика швидкість роботи, що збільшується при використанні різних модифікацій алгоритму. Алгоритм також стійкий до змін освітлення, і це дає можливість його використання для розпізнавання облич у системах обробки в реальному часі.

1.3 Методи, засновані на характерних точках

Надана група методів використовує координати характерних точок на зображенні. Такими характерними точками можуть бути, наприклад, центри очей, положення носа, лінія брів, рота тощо[10]. До цього класу методів належать активні моделі зовнішнього вигляду й активні моделі форми.

Метод Активні моделі зовнішнього вигляду (Active Appearance Models, AAM) — це статистичні моделі зображень, що використовують різного роду деформації, які потім будуть підігнані під реальне зображення. Цей тип моделей у двовимірному варіанті був запропонований Тімом Кутсом і Крісом Тейлором у 1998 році[15]. Активна модель зовнішнього вигляду складається з параметрів форми і параметрів зовнішнього вигляду. Параметри форми — це параметри, пов'язані з формою. Параметри зовнішнього вигляду - це параметри, пов'язані зі статистичною моделлю пікселів зображення або текстурою. Модель навчається на величезній кількості зображень, розмітка яких робиться вручну.

Активні моделі форми (Active Shape Models, ASM) — ця модель використовує статистичні зв'язки у взаємному розташуванні антропометричних точок. Для цього на всіх зображеннях вибірки розмічають антропометричні точки і далі розраховується загальний прокрустовий аналіз, у результаті якого всі точки приводяться до одного масштабу і центруються. Розраховується середня форма і матриця коваріації, на базі матриці коваріації далі розраховуються власні вектори, які сортуються в порядку убування власних значень. Локалізація ASM моделі на новому зображенні, яке не входить до навчальної вибірки, здійснюється в процесі розв'язання оптимізаційної задачі[16].

Ця модель була призначена для точної локалізації відмітних точок на зображеннях обличчя, а не для розпізнавання облич. Використання моделі давало змогу вирівнювати обличчя вибірки та приведення їх до однієї системи координат для подальшого більш точного розпізнавання різними методами. Для прискорення продуктивності береться невелика кількість точок. Для завдань

розпізнавання, навпаки, потрібна велика кількість характерних точок, що збільшить точність класифікації та знизить швидкість роботи системи[17].

1.4 Модель згорткової нейронної мережі (CNN)

Для моделі згорткової нейронної мережі (CNN) були використані деякі особливості зорової кори мозку, такі як прості та складні клітини. Прості клітини, мали можливість реагувати на прямі лінії під різними кутами, а у складних клітин реагування було пов'язане з активацією певного набору простих клітин. Згорткова нейронна мережа (convolutional neural network - CNN) — це клас глибоких штучних нейронних мереж прямого розповсюдження, які застосовують для аналізу зображень без попереднього оброблення. Основна перевага мережі — це автоматичний вибір коефіцієнтів фільтрів, у той час, коли інші методи це роблять вручну. На даний момент згорткові нейронні мережі є найкращими за точністю і швидкістю методами класифікації великої кількості складних візуальних об'єктів на зображеннях [18].

Ідея згорткових нейронних мереж являє собою чергування згорткових шарів (англ. convolution layers) і субдискретизуючих шарів (англ. subsampling layers або англ. pooling layers, шарів підвибірки) рис 1.1.

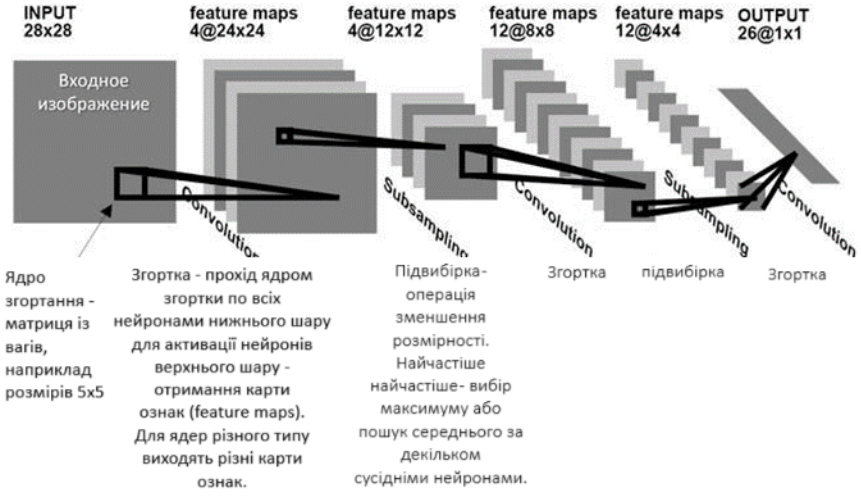


Рисунок 1.1— Реалізація згорткової нейронної мережі

Згорткова НС зроблена з великої кількості шарів. Отже, сигнал проходить спочатку початковий шар (вхідного зображення), потім серію згорткових шарів,

у яких іде шар згортки та шар субдискретизації (пулінг). Ця послідовність дає можливість створення своєї «карти ознак» із карт ознак, враховуючи при цьому, що з кожним наступним шаром карта зменшуватиметься в розмірі, а кількість каналів збільшуватиметься. Потім, коли карта ознак пройде певну кількість шарів, перероджуються у вектор і їхня кількість збільшується.

2 АНАЛІЗ ЗАДАЧІ РОЗПІЗНАВАННЯ ОБЛИЧ У ВІДЕОПОТОКАХ

Завдання розпізнавання облич у відеозображеннях для системи реєстрацій подій міститиме в собі виявлення і розпізнавання різними методами.

2.1 Метод Віоли-Джонса для розпізнавання облич

Метод Віоли-Джонса — алгоритм, що дає змогу виявляти об'єкти на зображеннях у реальному часі. Був розроблений і презентований Полом Віолою і Майклом Джонсом у 2001 році. [3] Він є одним із найкращих методів розпізнавання. Класифікатор проходить навчання повільно, але результати пошуку обличчя швидкі. Відмінні результати розпізнавання обличчя за малого кута нахилу, близько 30 градусів. Якщо ж кут нахилу буде більшим за 30 градусів, то і відсоток виявлень низький.

Механізм виявлення об’єктів:

— інтегральне представлення зображень;

— використання ознак Хаара;

— використання прискорення (бустінг);

— використання класифікатор для порівняння ознак;

— використання каскадів ознак для видалення непотрібних вікон.

2.2 Інтегральне представлення зображення

Для розрахунку яскравості прямокутної ділянки зображення використовуються так зване інтегральне подання[19]. Інтегральне подання обчислює сумарну яскравість довільного прямокутника на наданому зображенні швидко з урахуванням того, що час розрахунку залишається незмінним. Являє собою матрицю, що збігається за розмірами з початковим зображенням, у якій кожен елемент є сумою інтенсивностей усіх пікселів, що знаходяться лівіше і вище поточного елемента і розраховуються за формулою (2.1).

Елементи матриці розраховуються за такою формулою (2.1):

$$\text{□}(\text{□},\text{□}) = \sum_{\substack{\text{□}\le\text{□},\text{□}\le\text{□} \\ \text{□}=0,\text{□}=0}} \text{□}(\text{□},\text{□}) \qquad (2.1)$$

де $I(i,j)$ — яскравість пікселя вихідного зображення.

Отже, кожен елемент матриці містить необхідну суму пікселів у прямокутнику від (0, 0) до (x, y). Розрахунок матриці займає лінійний час, пропорційний числу пікселів вхідного зображення, у зв'язку з цим скористаємося для розрахунку такою формулою (2.2):

$$\text{□}(\text{□},\text{□}) = \text{□}(\text{□},\text{□}) - \text{□}(\text{□}-1,\text{□}-1) + \text{□}(\text{□},\text{□}-1) + \text{□}(\text{□}-1,\text{□}) \qquad (2.2)$$

у методі Віоли-Джонса застосовуються прямокутні ознаки, які називаються ознаками Хаара. При застосуванні в бібліотеці Open CV разом зі звичайними ознаками використовуються додаткові ознаки. Значенням ознаки для досліджуваної області зображення вважається число F=W-B.

2.3 Примітиви Хаара

Уперше використання для виявлення об'єктів ознак, що ґрунтуються на вейвлетах Хаара, було запропоновано в роботі Папагеоргіу в 1998 році [20] Віола і Джонс адаптували цю ідею у своїй роботі й отримали прямокутні ознаки, названі ознаками Хаара. [3]

Примітиви Хаара являють собою набір пікселів, частина з яких білі, а частина чорні. Алгоритм є одним із перших алгоритмів детектування облич, що працює в реальному часі. Для виявлення обличчя використовують скануюче вікно, яке пересувається по зображенню з кроком в 1 піксель. Далі розраховують ознаки Хаара з різним масштабом і розташуванням у вікні, передають класифікатору, який описує за їхніми значеннями і визначає, чи є область зображення, що відповідає вікну, особою чи ні.

Примітиви Хаара являють собою прямокутники, які складаються із суміжних областей (рис. 2.1 і 2.2).

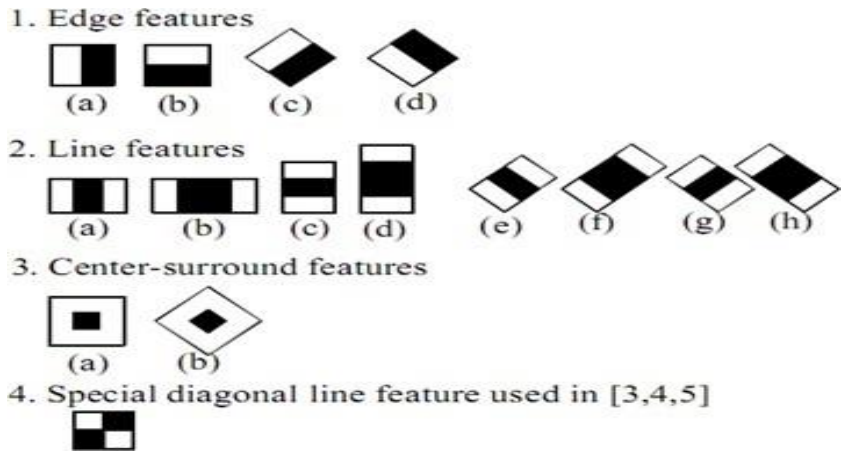


Рисунок 2.1 — Примітиви Хаара

Рисунок 2.2 — Додаткові примітиви Хаара, що використовуються в OpenCV

Рисунок 2.3 — Приклад знаходження ознак Хаара на зображенні

Примітиви Хаара являють собою набір пікселів, частина з яких білі, а частина чорні. Ознака — це відображення f: $x \Longrightarrow Df$, де Df — множина допустимих значень ознаки. Якщо задано ознаки $f1$, ... fn то вектор ознак $f1(x)$, ... $fn(x)$ називається ознаковим описом об'єкта $x \in X$.

При цьому множину $X = Df1 * ... ... * Dfn$ називають ознаковим простором. Ознаки поділяють на такі типи залежно від множини Df :

— бінарна ознака, $Df = \{0,1\}$;

— номінальна ознака, Df = скінченна множина;

— порядкова ознака, Df = скінченна впорядкована множина;

— кількісна ознака, Df = множина дійсних чисел.

Обчислюваним значенням такої ознаки буде $F = X - Y$ (2.3) де X — сума значень яскравостей точок, що закриваються світлою частиною ознаки, Y — сума значень яскравостей точок, що закриваються темною частиною ознаки. Для їх обчислення використовується поняття інтегрального зображення. Ознаки Хаара дають точкове значення перепаду яскравості по осі X і Y, відповідно. Перевагою ознак Хаара є порівняно висока швидкість обчислення.

Під час пошуку обличчя необхідно виділити його основні компоненти, як: ніс, лоб, очі, губи тощо. Це можна зробити за допомогою використання шаблонів (вони ж примітиви Хаара) (рис. 2.4).

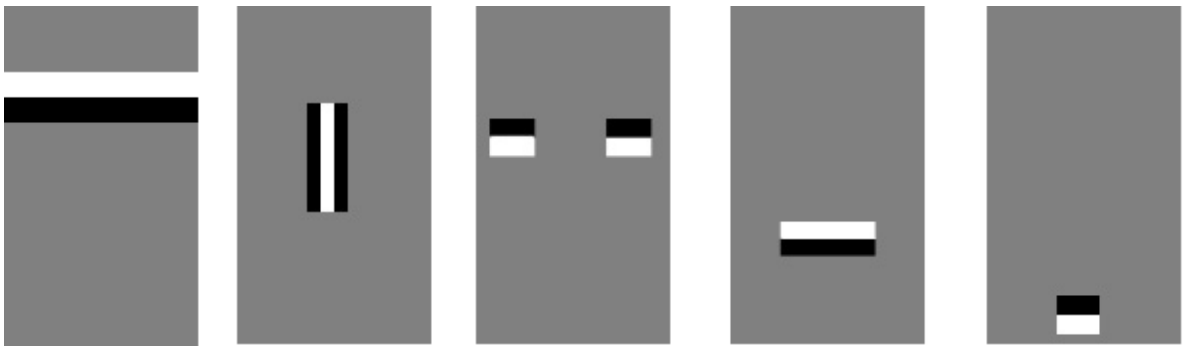


Рисунок 2.4 — Примітиви Хаара

Якщо шаблони відповідають конкретним областям на зображенні, то вважається, що на зображенні є людське обличчя. Кількість шаблонів необмежена. Далі розраховується різниця між яскравістю білої та чорної областей і отримане значення порівнюється з еталоном і вирішують, існує там частина людського обличчя чи ні.

2.4 Метод Eigen Faces

Eigenface заснований на методі головних компонент (PCA). PCA має

можливість зменшити розмірність даних, не втрачаючи при цьому великої кількості інформації. Основним підходом цього методу є стиснення інформації вихідного зображення без істотних втрат інформативності за допомогою методу головних компонент (PCA — Principle Component Analysis)[21].

Алгоритм PCA в Eigenfaces:

— у PCA застосовується тренувальна добірка зображень розмірності $X = \{x1, x2, ..., ... , xn\}$, де xi матриці осіб, перевтілені у вектори, що є стовпчики спільної матриціX (Рисунок 2.5);

— проводимо розрахунок математичного очікування (див. 2.12);

— розраховуємо матрицю коваріації(див. 2.13).

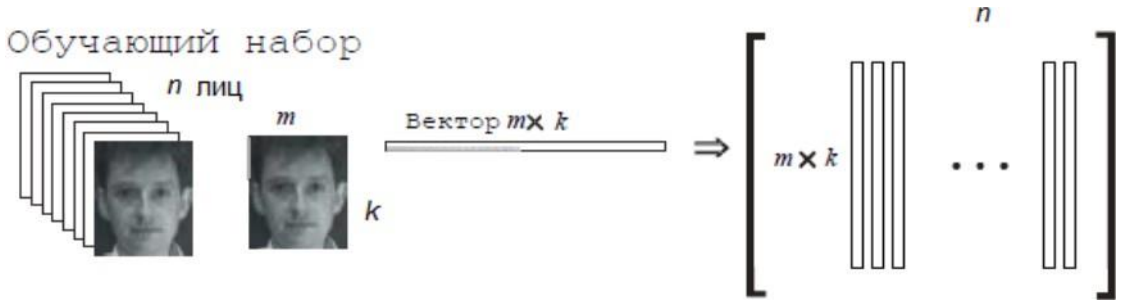


Рисунок 2.5 — Вибірка облич

$$\bar{\Gamma} = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n \qquad (2.12)$$

Математичне очікування у форматі зображення бачимо на рисунку 2.6

Рисунок 2.6 — Приклад середнього обличчя

$$\square = \frac{1}{\square}\sum_{\square=1}^{\square}(\square_{\square} - \square_{\square})(\square_{\square} - \square_{\square})^{\square} \quad (2.13)$$

Для залишення необхідної нам інформації про обличчя віднімаємо середнє з кожного зображення. При виведенні кожного стовпчика у вигляді зображення виходить картина (рисунок 2.7):

Рисунок 2.7 — Еталонні зображення та їх відхилення від середнього

Розраховуємо власні значення λ і власні вектори ν:

$$Sv_i = \lambda_i v_i,\ i = 1 \dots n. \quad (2.14)$$

Обираємо k власних векторів, у якого найбільше власне значення λ. Перебудовуємо базис і отримуємо новий простір:

$$y = W^T(x - m_x) \quad (2.15)$$

$$\{\ x = Wy + m$$

де $W = \{v1, \dots, vn\}$, x — новий скорочений простір.

Цей метод розпізнає обличчя за допомогою:

— проектування всіх тренувальних зразків усередині підпростору PCA;

— проектування вхідного зображення всередині простору PCA;

— проектування вхідного зображення всередині простору PCA;

— пошуку найближчого сусіда між тренувальними зразками та вхідним зображенням.

Існує одна проблема, яка виникає під час використання цього методу. Наприклад, якщо дано 400 зображень розміром 100 x 100. Застосовуючи PCA отримуємо матрицю коваріації $S = XXXT$, де розмір матриці X дорівнює 400 x 10000, а розмір матриці S - 10000 x 10000, це приблизно 0,8 GB. Для розв'язання цієї проблеми використовують правило лінійної алгебри: матриця $M\ x\ N$, за $M > N$, може мати тільки N - 1 ненульових власних значень.

Для цього береться розкладання власних значень $S = XTX$ з розміром N x N на заміну:

$$[illegible] = [illegible] \quad (2.16)$$

і в результаті отримуємо вихідні власні вектора.

$$[illegible]([illegible]) = [illegible]([illegible]) \quad (2.17)$$

У підсумку ми отримуємо відстань та індекс об'єкта з бази, до якого вхідне зображення розміщене ближче. Власні вектори, у яких розмірність така сама, як у початкового зображення, називаються власними зображеннями або власними обличчями.

Переваги методу Eigen Faces:

— швидкість і швидкість розпізнавання;

— проста база облич, яка використовує мало місця.

Недоліки методу Eigen Faces:

— вплив зміни ракурсу та освітлення на чутливість

— додавання нових облич до бази призводить до необхідності будувати простір із власних векторів наново, оскільки навчання роблять тільки один раз.

2.5 Метод Fisher Faces

Основою методу Eigenfaces вважається метод головних компонент (PCA), що використовує лінійну комбінацію ознак і максимізує єдину дисперсію даних. Сильний метод представлення даних, але не передбачає класи об'єктів і тому є ймовірність втрати великої кількості характерної інформації через відкидання компонент, які не увійшли до базису. Приміром, якщо в базі даних збережені фотографії з різним освітленням і маємо компоненти, визначені PCA, але які не містять усієї характерної інформації, то зразки різних класів перемішуються один з одним, подальше класифікування неможливе. Для розв'язання цієї проблеми використовують алгоритм LDA (Лінійний дискримінантний аналіз), який є основою методу Fisher Faces [21].

Алгоритм Fisherface був винайдений для вдосконалення результатів розпізнавання при зміні освітлення або виразу обличчя. Основою його є лінійний дискримінантний аналіз LDA. Лінійний дискримінантний аналіз — це метод статистики та машинного навчання, що використовується для знаходження лінійних комбінацій ознак і розділяє два або кілька класів об'єктів або подій, це означає, що об'єкти одного й того самого класу мають бути близькими у просторі, поряд із цим має відбуватися максимізація відстані між класами. Ця комбінація дає можливість застосування її як лінійного класифікатора або для зменшення розмірності простору ознак перед наступною класифікацією.

Алгоритм LDA (Linear Discriminant Analysis) у Fisher Faces — це підхід для розпізнавання облич, що поєднує в собі методи аналізу головних компонент (PCA) та лінійного дискримінантного аналізу (LDA). Він використовується для покращення розпізнавання облич шляхом зменшення розмірності вхідних даних та максимізації роздільності між різними класами облич.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.18}$$

розраховуємо математичне очікування для кожного класу:

$$\mu_i = \frac{1}{|X_i|}\sum_{x_j \in X_i} x_j \tag{2.19}$$

розраховуємо матрицю міжкласового розкиду:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{2.20}$$

розраховуємо матрицю всередині класового розкиду:

$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T \tag{2.21}$$

де c — кількість класів у навчальній базі.

У результаті отримуємо підсумковий базис:

$$W_{opt} = W^T \frac{|W^T S_B W|}{|W^T S_W W|} \tag{2.22}$$

Метод Fisher Faces набагато краще розпізнає обличчя, ніж метод EigenFaces, за умови, що навчання відбувалося при різних освітленнях.

AT&T є однією з найзвичайнісіньких баз, у якій дотримано чудових умов: хороше освітлення, достатня кількість фотографій для кожної особи тощо. На рисунку 2.8 зображено результати роботи з базою осіб AT&T.

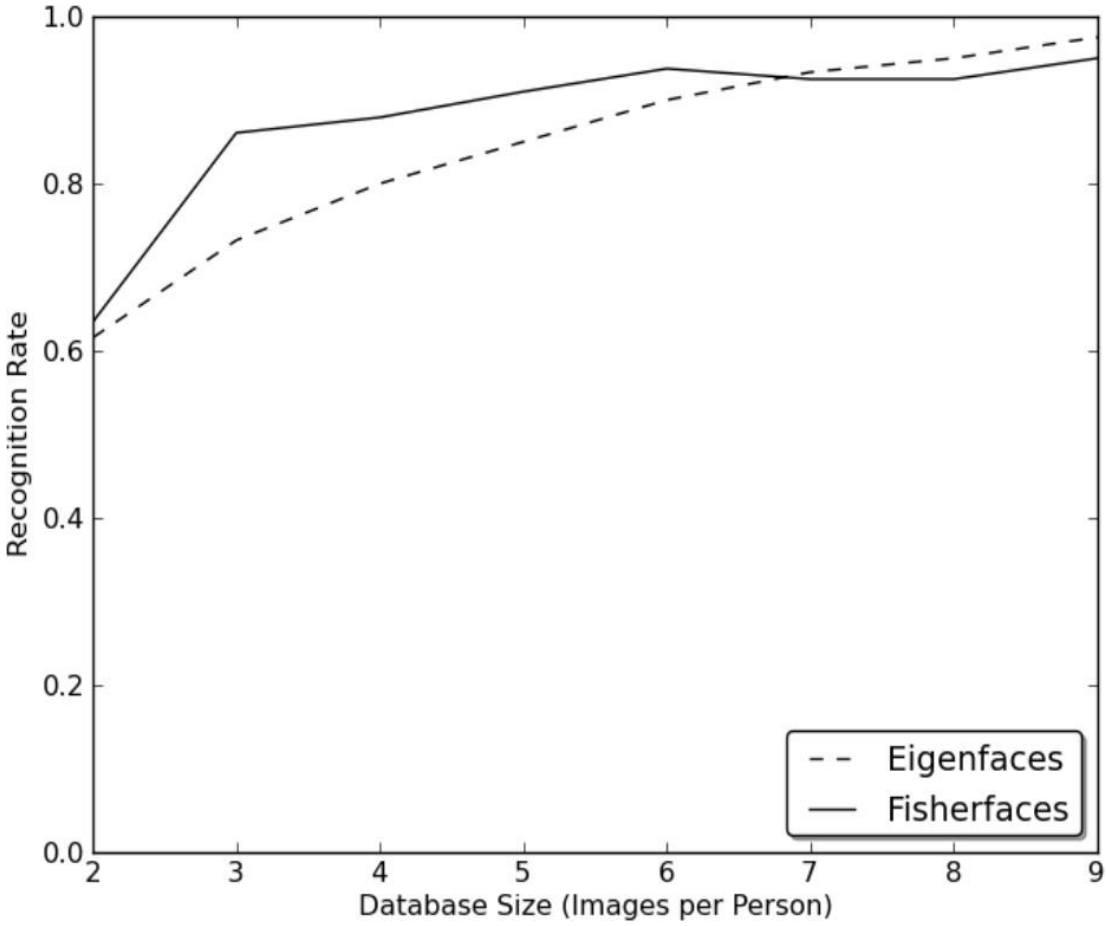


Рисунок 2.8 — Графік залежності точності розпізнавання від кількості фотографій для кожної людини в базі

2.6 Метод локальні бінарні шаблони (LBPH)

Локальні бінарні шаблони (англ. Local Binary Patterns, LBP) — простий оператор, який використовують для класифікації текстур у комп'ютерному зорі. LBP оператор уперше було запропоновано 1996 року для класифікації текстур[22]. Пізніше був застосований і для розпізнавання облич. Сутність оператора в застосуванні до пікселів зображення порогового перетворення і порівняння значень яскравості оброблюваного пікселя зі значеннями яскравостей пікселів його околиць, результати потім конкатенують у двійкове число.

Опис роботи методу LBPH:

1) Зображення розділяється на однакові блоки, які утворюють сітку (рис.2.9).

2) Будується гістограма кодів для кожного блоку і обчислюється. Далі береться піксель і порівнюється з сусідами. Якщо інтенсивність центрального пікселя буде більшою або дорівнюватиме інтенсивності сусіда, то він позначається 1, а інакше — 0.

3) У результаті порівнянь кожному пікселю буде відповідати двійкове число.

Ці перетворення наочно показано на рисунку 2.9. Отримані гістограми об'єднуються в одну загальну, яка є підсумковим дескриптором, що використовується для класифікації обличчя [18].

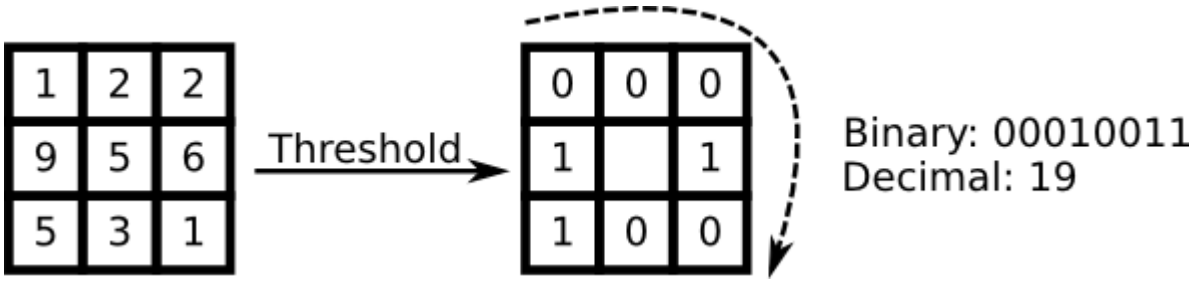


Рисунок 2.9 — Перетворення області в двійковий код

Отже, після застосування LBP-оператора, зображення ділиться на прямокутні області та розраховуються гістограми, які описують частоту зустрічаючихся в даній області пікселів різних значень яскравості. Значення елементів LBP гістограми можуть бути описані такою формулою (2.23):

$$H_i = \sum_{x,y} I\{f(x,y) = i\}, i = 0, \dots, n-1 \quad (2.23)$$

де $f(x, y)$ — значення яскравості пікселя LBP зображення з координатами (x, y);

n — кількість різних значень яскравості пікселів; $I\{A\} = 1$, якщо A — правда, інакше $I\{A\} = 0$.

Отримані гістограми нормалізуються, конкатенуються і будуть використовуватися як ознаки класифікації. Приклад поділу зображення на прямокутні області та формування гістограм можна побачити на рисунку 2.10.

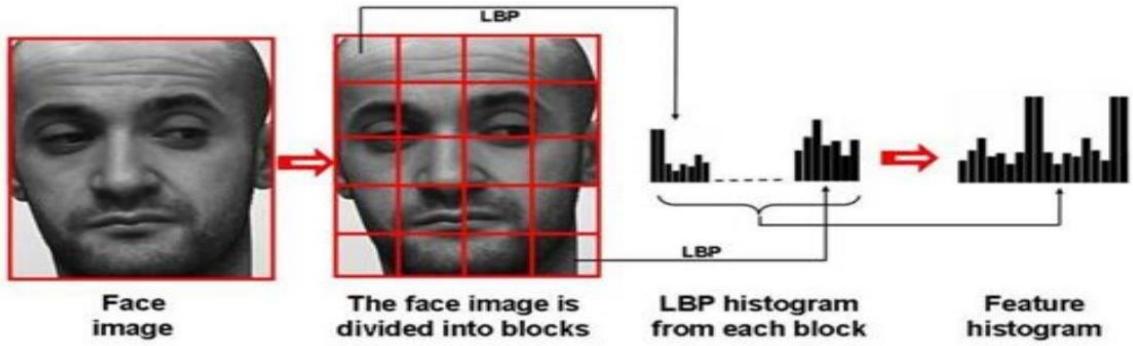


Рисунок 2.10 — Схема роботи алгоритму LBPH

У підсумку опис зображення обличчя буде на трьох рівнях локалізації, і незалежний від змін освітлення.

Подальші дослідження локальних бінарних шаблонів показали, що істотну інформацію про форму об'єктів на зображенні несе тільки частина з них [1]. Ці локальні бінарні шаблони були названі рівномірними (uniform local binary patterns). До них відносяться шаблони, у яких двійковий код містить не більше двох переходів між нулем і одиницею, і описують найважливіші локальні особливості зображення, як-от кінці ліній, грані, плями тощо. Приклади рівномірних LBP наведено на рисунку 2.11

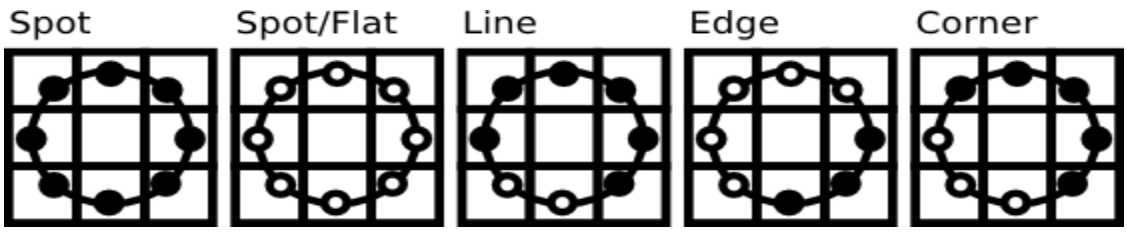


Рисунок 2.11 — Ознаки кругової околиці

Усього налічується 58 рівномірних LBP. У підсумку виходить 59-мірна гістограма ознак (додатковий розряд відводиться для підрахунку всіх нерівномірних LBP), на відміну від 256-мірної гістограми в оригінальному алгоритмі. Це обмеження розмірності допомагає зменшити витрати пам'яті, збільшити швидкість класифікації та поліпшити її показники за рахунок застосування принципово важливих ознак. Отримані гістограми класифікуються методом найближчого сусіда, де об'єкт належатиме до того класу, до елемента якого він розташований найближче. Наприклад, на рис. 2.12 зелене коло відповідно до алгоритму класифікується як червоне.

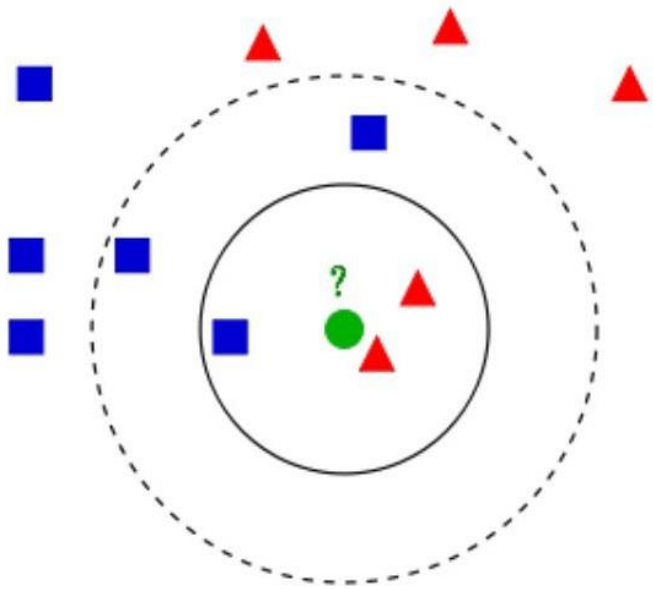


Рисунок 2.12 — Метод найближчого сусіда

Для вдосконалення підсумків вживають метод, у якому цей об'єкт відносять до того класу, до якого належить більшість його сусідів в околиці заданого розміру. Але під час розв'язання задачі класифікації осіб це має поганий вплив на роботу класифікатора. Математично алгоритм можна описати таким чином. На першому кроці визначається елемент xs навчальної вибірки з N елементів, який найближче до пред'явленого образу x, тобто (2.24)

$$\|\square - \square_{\square}\| = \min\{\|\square - \square_{\square}\| : \square = 1, \ldots, \square\} \tag{2.24}$$

на другому кроці перевіряється умова приналежність до класу: якщо $xs \in \acute{\omega}j$, то тоді вважається, що і $x \in \acute{\omega}j$, то тоді вважається, що і $x \in \acute{\omega}j$.

Метод використовується за умови, коли помилки неправильної класифікації великі, а помилки даних маленькі. Головний недолік методу — це чутливість до значень окремих даних. Незважаючи на це, метод показує високу ефективність при застосуванні в широкому спектрі завдань класифікації [23].

На особливий інтерес заслуговує питання вибору метрики, що характеризує відстань між гістограмами. Для досягнення найбільшої точності класифікації потрібно вибрати ту метрику, яка найправильніше б відображала відмінності між гістограмами зображень різних класів. В оригінальному дослідженні використовується так звана відстань Chi- Square, яку розраховують за такою формулою:

$$\sum_{\square=1}^{\square} \frac{(\square_{\square} - \square_{\square})^2}{(\square_{\square} + \square_{\square})} \tag{2.25}$$

де xi- і-те значення першої гістограми; і-те значення другої гістограми.

2.7 Згорткові нейронні мережі

Згорткова нейронна мережа — це клас глибоких штучних нейронних мереж прямого розповсюдження, що використовуються для аналізу зображень і мають можливість проаналізувати зображення без попереднього опрацювання, оскільки мережа може автоматично підібрати коефіцієнти фільтрів. Шар згортки (convolutional layer) — складається з набору фільтрів для всіх каналів або ядер згортки, що обробляють отриману на вхід карту ознак. Вагові коефіцієнти ядра згортки інсталюються в процесі навчання. Операція субдискретизації (pooling, subsampling) зменшує розмірності сформованих карт ознак.

Структура згорткової НС, представлена на рисунку 3.1, складається з різних видів шарів: згорткові (convolutional) шари, шари субдискретизації

(subsampling), а також повнозв'язні шари, що служать для класифікації зображень. (рис. 2.13)

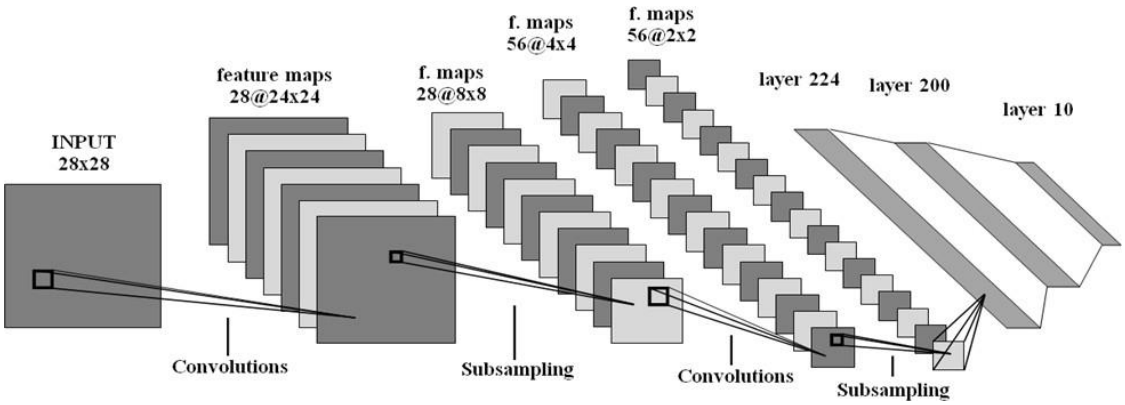


Рисунок 2.13 — Згорткові нейронні мережі

Згорткові кола і субдискретизація, чергуючись між собою, формують вихідну карту показників для мультишарового повнозв'язного класифікатора. Чисельність цих шарів перебуває залежно від топології підступи, яку характеризують, орієнтуючись на:

— розв'язуване завдання (класифікація, прогнозування, генерація тощо);

— обмеження в розв'язуваній задачі (швидкість, точність відповіді та ін.);

— вхідні дані (тип, розмір, формат);

— вихідні дані.

Згорткові шари і субдискретизація, чергуючись між собою, формують вихідну карту ознак для багатошарового повнозв'язного класифікатора.

Чисельність цих шарів перебуває залежно від топології мережі, яку визначають, орієнтуючись на:

— розв'язуване завдання (класифікація, прогнозування, генерація тощо);

— обмеження в розв'язуваній задачі (швидкість, точність відповіді та ін.);

— вхідні дані (тип, розмір, формат);

— вихідні дані;

— методи оптимізації структури та вибору гіперпараметрів мережі [24].

2.7.1 Архітектура згорткових нейронних мереж

Стандартний блок згорткової НС складається зі згортки, пулінгу та субдискретизації. Шар згортки (convolutional layer) — складається з набору фільтрів для всіх каналів або ядер згортки, що обробляють отриману на вхід карту ознак. Вагові коефіцієнти ядра згортки інсталюються в процесі навчання. Операція субдискретизації (pooling, subsampling) зменшує розмірності сформованих карт ознак.

Головною особливістю згорткових нейронних мереж вважається використання операції згортки для аналізу оброблюваних образів. Операція згортання відбувається так: ядро згортання ковзає оброблюваною ділянкою та виконує зважену суму елементів цієї ділянки щодо значень коефіцієнтів ядра:

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] * [k, l] \quad (2.26)$$

де f— вихідна матриця зображення;

g — ядро згортання.

Математично операція, що виконується у згортковому шарі, матиме такий вигляд:

$$x^l = f(x^{l-1} * w^l + b^l) \quad (2.27)$$

де xl — вихід шару l;

$f()$) — функція активації;

bl — коефіцієнт зсуву шару l ;

*- операція згортки входу x з ядром k.

При цьому розмір вихідних матриць зменшується.

$$x_j^l = f(\sum_i x_i^{l-1} * w_j^l + b_j^l)$$

де $x\ l$ — карта ознак jj (вихід шару ll);

$f()$) — функція активації;

bl — коефіцієнт зсуву шару l для карти ознакj;

kl — ядро згортки j карти, шаруl ;

* — операція згортки входуx з ядром k.

Кожен згортковий шар складається з великої кількості настроюваних синоптичних ядер згортки, що налаштовуються. Таке подання дає змогу втілити систему поділюваних ваг у нейронній мережі. На відміну від повнозв'язної НС, у згортковій мережі спільні ваги дають змогу зменшити кількість зв'язків і втілити ймовірність виділення схожих ознак на всій ділянці зображення. Згортковий шар у результаті роботи сформує набір карт ознак (featuremaps). Усі карти ознак, створювані згортковим шаром, мають однаковий розмір, який обчислюють за такою формулою:

$$(w, h) = (mW - kW + 1, mH - kH + 1) \qquad (2.29)$$

де (w, h) — обчислюваний розмір згорткової карти;

mW — ширина попередньої карти;

mH — висота попередньої карти;

kW — ширина ядра;

kH — висота ядра

Ядро — це фільтр, який визначає ознаки об'єктів на всій області попередньої карти. У процесі навчання ядро виділятиме певні ознаки, наприклад, такі, як області обличчя, очей тощо. Величина ядра підбирається відповідно до розмірів ознак, що виділяються. Зазвичай його задають у межах від 3x3 до 7x7.

Приступаємо до розв'язання задачі зменшення розмірності карт ознак, отриманих на згортковому шарі. Для цього переходимо до операції пулінгу (pooling), яка використовує функцію, що розраховує найбільше значення на оброблюваній карті. Відбираємо ознаки, що мають найбільше значення.

Математично це описується такою формулою:

$$\square^{\square} = \square(\square^{\square} * \square\square\square\square\square\square\square\square\square\square(\square^{\square-1}) + \square^{\square}) \qquad (2.30)$$

де xl — вихід шару l

$f()$ — функція активації;

al , bl — коефіцієнти зсуву шару l ;

$subsample$ — операція вибірки локальних максимальних значень.

Повнозв'язний шар вирішує завдання класифікації. Цей шар моделює складну нелінійну функцію, оптимізуючи параметри якої, поліпшується якість розпізнавання. Число нейронів прихованого шару дорівнює числу карт шару субдискретизації, оскільки нейрони кожної карти попереднього шару субдискретизації пов'язані з одним нейроном прихованого шару.

$$x_i^l = f(\sum_i x_j^{l-1} * w_{i,j}^{l-1} + b_j^{l-1})$$

де xl — карта ознак j (вихід шару l);

$f()$ — функція активації;

bl — коефіцієнт зсуву шару l ;

wl-11— матриці вагових коефіцієнтів шару l.

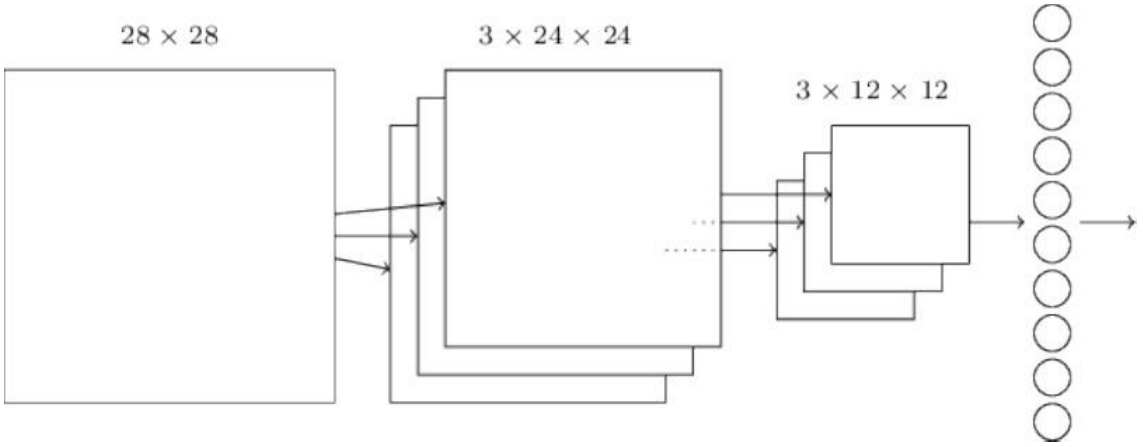


Рисунок 2.15 — Приклад згорткової нейронної мережі

На рисунку 2.15 представлено схему згорткової нейронної мережі для розпізнавання рукописних цифр. На вхід мережа приймає чорно-біле зображення розміром 28 х 28 пікселів. На останньому шарі і-й нейрон характеризує ймовірність того, що мережа отримала на вхід зображення з цифрою *i*.

3 РОЗРОБКА СИСТЕМИ РЕЄСТРАЦІЇ НА ОСНОВІ АНАЛІЗУВАННЯ ВІДЕОЗОБОРАЖЕНЬ

3.1 Бібліотека OpenCV

У цій роботі буде використано мову програмування Python, бібліотеку Open CV, бібліотеку Face_ Recognition.

Python — це високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника та читабельності коду. Для python розпізнавання обличчя реалізовано на OPENCV 2 покоління.

Бібліотека Open CV — це бібліотека алгоритмів комп'ютерного зору, оброблення зображень і чисельних алгоритмів загального призначення з відкритим кодом.

Бібліотека Face_Recognition — це бібліотека, яка охоплює функцію розпізнавання облич dlib.

Для створення системи реєстрацій буде реалізовано та протестовано:

— бібліотека Open CV і методи LBPN, Eigen Face, Fisher Face;

— бібліотека Face_ Recognition і модель згорткової нейронної мережі.

Після порівняння результатів буде обрано найкращу бібліотеку, за допомогою якої буде створено систему реєстрацій подій.

Для тестування з бібліотеки Open CV буде обрано три методи LBPN, Eigen Face, Fisher Face. Потім буде проведено тестування бібліотеки Face_Recognition і після отриманих результатів зробимо вибір найбільш підходящої для нашої системи реєстрацій подій бібліотеки. Тестування відбуватиметься на відеопотік, що дасть змогу визначити ефективність методів і бібліотек.

3.1.1 Метод LBPH (Local Binary Patterns Histograms)

У цьому розділі тестування відбуватиметься з використанням бібліотеки OpenCV і методом локальних бінарних шаблонів. На цьому етапі наша програма повинна розпізнати обличчя, збережені в базі даних. Тестування методом локальних бінарних шаблонів необхідне для подальшого порівняння та вибору

найкращого методу для розпізнавання облич і створення нашої системи реєстрацій подій.

Для того, щоб використовувати Open CV, нам необхідно спочатку імпортувати бібліотеку. (лістинг 3.1).

Лістинг 3.1 — Імпортуємо бібліотеку Open CV

```
Import cv2
```

Потім ми імпортуємо бібліотеку Numpy (лістинг 3.2).

Лістинг 3.2 — Імпортувати бібліотеку numpy

```
Import numpy as np
```

Завантажуємо класифікатор (лістинг 3.3).

Лістинг 3.3 — Завантажити класифікатор

```
F    aceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
```

Додаємо об'єкт відео захоплення (лістинг 3.4).

Лістинг 3.4 — Додамо об'єкт відео захоплення

```
cam=cv2.VideoCapture(1);
```

Для виявлення обличчя з веб-камери в реальному часі нам необхідно створити цикл, який отримуватиме зображення з веб-камери кадр за кадром, виявлятиме обличчя і показуватиме їх у вікні (лістинг 3.5).

Лістинг 3.5 — Виявити обличчя з веб-камери в реальному часі

```
while(True):
    ret,img=cam.read();
    gray =
    cv2.cvtColor(img,cv2.COLOR_BGR2GR
    AY)
    faces=faceDetect.detectMultiScale(gray,1.
    3,5); for(x,y,w,h) in faces:
```

```
cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.imshow("Face",i
    mg);
    if(cv2.waitKey(1)==
    ord('q')):
      break;
```

Після закінчення програми нам необхідно звільнити об'єкт захоплення відео, для цього ми знищуємо всі вікна (лістинг 3.6).

Лістинг 3.6 — Звільнити об'єкт захоплення відео та знищити всі вікна

```
cam.release()
cv2.destroyAllWindow()
```

Запускаємо код і дивимося отримані результати (рис. 3.1)

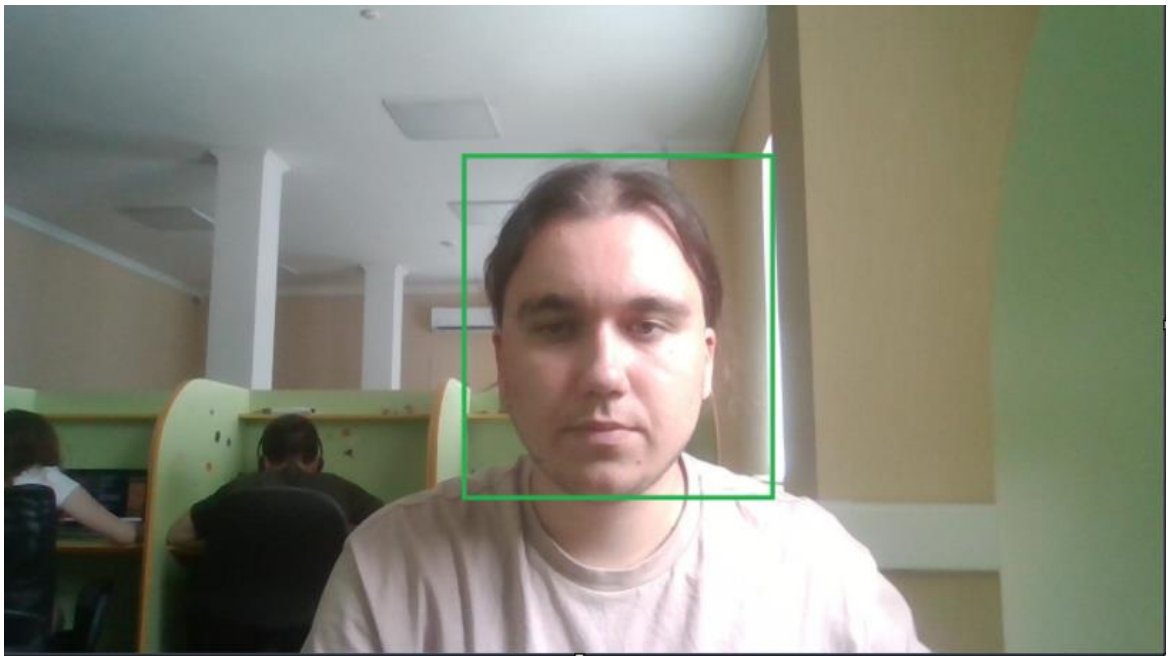


Рисунок 3.1 — Виявлення обличчя

Для створення програми розпізнавання облич, нам необхідно навчити розпізнавач. Для навчання використовуємо набір даних, у яких збережено раніше захоплені обличчя разом із його ідентифікатором. Наприклад, у нас є двоє людей, тоді в першої людини буде ID 1, а в другої людини буде ID 2, і всі зображення першої людини в наборі даних матимуть ідентифікатор 1, а всі зображення 2-ї

людини в наборі даних матимуть ідентифікатор 2. Ці зображення з набору даних використовуватимуться для навчання розпізнавача, який має передбачити 1 з нещодавно представлених облич із живого відеокадру.

Наш генератор набору даних захопить кілька зразків обличчя однієї людини з живої відеокарти і призначить їй ідентифікатор, зберігаючи ці зразки в папці, яку ми створимо, і назвемо її dataSet.

Отримуємо ідентифікатор користувача з оболонки як введення та ініціалізуємо змінну лічильника для зберігання номера зразка (лістинг 3.7)

Лістинг 3.7 — Ідентифікатор користувача

```
id=raw_input('enter
user id') sampleNum=0;
```

Додаємо ці два рядки й отримуємо номер зразка та зберігаємо обличчя у форматі jpg за допомогою нашої угоди про імена (лістинг 3.8)

Лістинг 3.8 — Отримати номер зразка і зберегти обличчя у форматі jpg

```
sampleNum=sampleNum+1;
cv2.imwrite("dataSet/User."+str(id)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
```

Для того, щоб захопити обличчя під різними кутами, необхідно, щоб швидкість була набагато повільнішою. Отримуємо це за допомогою збільшення затримки між кадрами і ламаємо цикл відразу після того, як набереться 20 зразків (лістинг 3.9)

Лістинг 3.9 — Цикл захоплення обличчя

```
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img,
    cv2.COLOR_BGR2GRAY) faces =
    detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        # збільшення числа зразків
```

```
        sampleNum=sampleNum+1
        # збереження захопленої особи в папці набору даних
        cv2.imwrite("dataSet/User."+Id +'.'+ str(sampleNum) + ".jpg",
        gray[y:y+h,x:x+w])
 #
        cv2.imshow('frame',img)
    # чекати 100 мілісекунд
    if cv2.waitKey(100) & 0xFF ==
        ord('q'): break
    # break якщо номер зразка більше 20elif sampleNum>20:
break
```

Запускаємо цей код і бачимо, що він захопив обличчя з відео в реальному часі та зберіг у папці dataSet (Рис.3.2.)

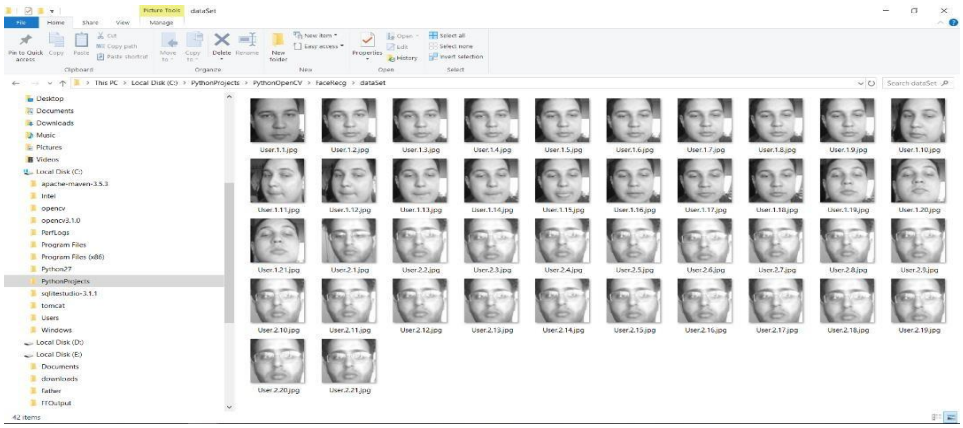


Рисунок 3.2 — Dataset

Як зазначалося раніше, щоб виконати розпізнавання обличчя, необхідно навчити спочатку розпізнавач, використовуючи попередньо набір даних, у якому позначені обличчя. У нас є вже створений нами помічений набір даних для нашої системи розпізнавання облич і перейдемо до його використання та навчання розпізнавача обличчя з використанням Open CV Python.

Тепер нам потрібно ініціалізувати розпізнавач і детектор обличчя (лістинг3.10)

Листинг 3.10 — Инициализировать распознаватель и детектор лица.

```
recognizer = cv2.createLBPHFaceRecognizer()
detector= cv2.CascadeClassifier(«haarcascade_frontalface_default.xml»);
```

Далі нам необхідно створити функцію, яка захоплюватиме навчальні зображення з папки dataSet і отримуватиме відповідні ідентифікатори з їхнього імені файлу (лістинг 3.11)

Лістинг 3.11 – Захоплення початкові навчальні зображення

```
def getImagesAndLabels(path):
    imagePaths=[os.path.join(path,f) for f in
    os.listdir(path)] faceSamples=[]

    Ids=[]
for imagePath in imagePaths:
    pilImage=Image.open(imagePath).conve
    rt('L')
    imageNp=np.array(pilImage,'uint8')
    Id=int(os.path.split(imagePath)[-
    1].split(".")[1])
    faces=detector.detectMultiScale(imageN
    p)

    for (x,y,w,h) in faces:
        faceSamples.append(imageNp[y:y+h,
        x:x+w]) Ids.append(Id)

    return faceSamples,Ids
```

Викликаємо цю функцію і передаємо дані розпізнавачу для навчання (лістинг 3.12)

Лістинг 3.12 — Викликання функції

```
Ids,faces=getImagesWithID(path)
recognizer.train(faces,Ids)
recognizer.save('recognizer/trainingData.yml')
```

Далі нам необхідно створити файл FaseBase.db у нашій папці проєкту за допомогою SQLite (рисунок 3.3)

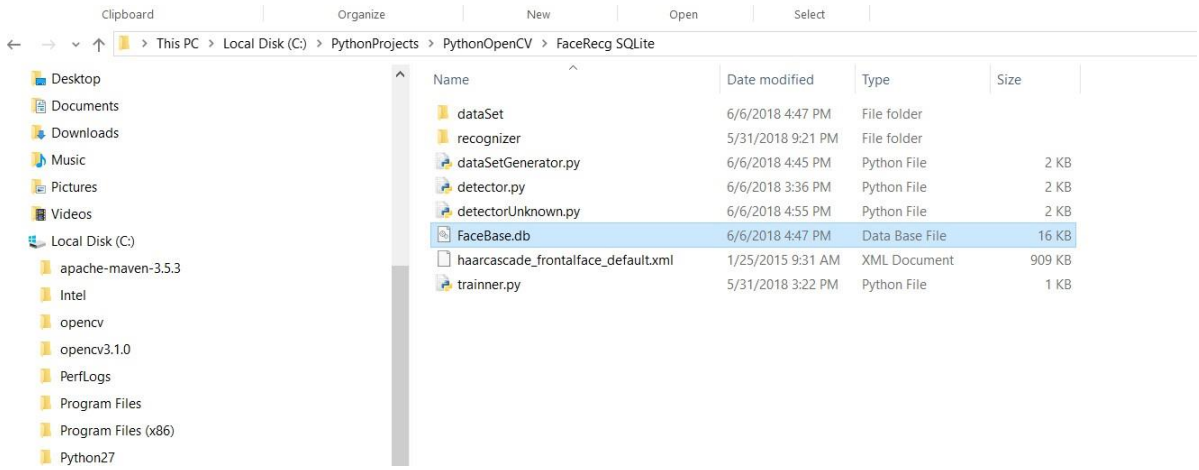


Рисунок 3.3 — FaseBase.db

Потім створюємо таблицю з ім'ям People (рисунок 3.4)

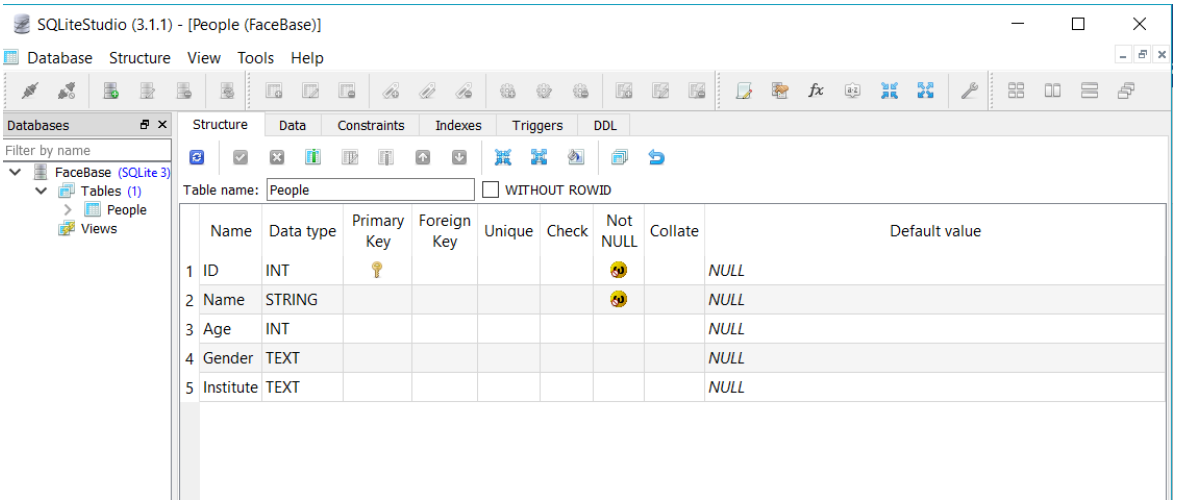


Рисунок 3.4 — Таблиця People

Тепер ми створюємо функцію getProfile (id) у python, яка підключається до нашої таблиці People IN SQLite (лістинг 3.13)

Лістинг 3.13 — функція getProfile (id)

```
def getProfile(id):
    conn=sqlite3.connect("FaceBase.db")
    cmd="SELECT * FROM People WHERE
    ID="+str(id) cursor=conn.execute(cmd)

    profile=None
```

```
    for row in cursor:
        profile=row
    conn.close()
    return
    profile
```

Напишемо основний цикл, який захоплюватиме кадри з об'єкта камери та перетворюватиме його на Gray Scale, потім виявлятиме та витягуватиме обличчя із зображень і використовуватиме розпізнавач для розпізнавання ідентифікатора користувача і далі зберігатиме його в базу даних. (лістинг 3.14)

Листинг 3.14 – Цикл While

```
  while(True):
    ret,img=cam.read();
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3
    ,5); for(x,y,w,h) in faces:
      id,conf=rec.predict(gray[y:y+h,x:x+w])
      cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
      profile=getProfile(id)
      if(conf<75):
        profile=getPro
        file(id)
      else:
        id=0
        profile=getPro
        file(id)
      if(profile!=None):
         cv2.cv.PutText(cv2.cv.fromarray(img),
         "Name:
"+str(profile[1]),(x,y+h+30),font,255)
         cv2.cv.PutText(cv2.cv.fromarray(img),"Age:
"+str(profile[2]),(x,y+h+60),font,255)
         cv2.cv.PutText(cv2.cv.fromarray(img),"Gender:
"+str(profile[3]),(x,y+h+90),font,255)
         cv2.cv.PutText(cv2.cv.fromarray(img),"Institute:
"+str(profile[4]),(x,y+h+120),font,255
```

```
cv2.imshow("Face",i
mg);
if(cv2.waitKey(1)==
ord('q')):
    break;
```

Запускаємо код і отримуємо такі результати (рисунок 3.5)

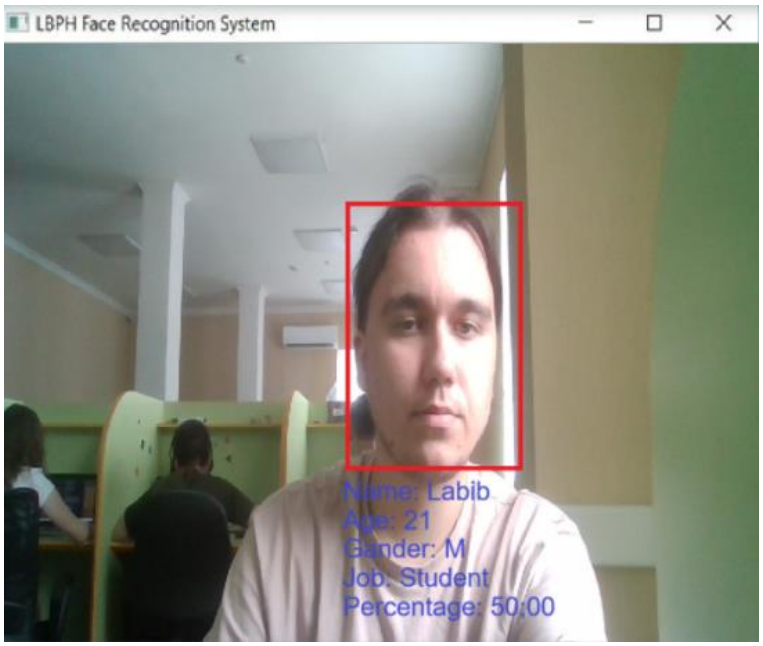


Рисунок 3.5 — Розпізнавання облич методом LBPH

3.1.2 Метод Eigenfaces

Eigenfaces — алгоритм, запропонований 1991 року Метью Терком і Алексом Пентландом, який здобув широку популярність як перший успішний метод розпізнавання облич. Алгоритм застосовує метод головних компонент для знаходження векторів, які найкращим чином описують зображення облич. Отриманий один раз на навчальній вибірці зображень облич набір власних векторів використовується для кодування всіх інших зображень облич, які представляються зваженою комбінацією цих власних векторів. Метод головних компонент добре зарекомендував себе в практичних додатках. Однак, у тих випадках, коли на зображенні обличчя присутні значні зміни в освітленості або виразі обличчя, ефективність методу значно падає.

До переваг цього методу можна віднести простоту реалізації, придатність для розпізнавання в реальному часі та можливість компактно зберігати великі обсяги даних.

Основним недоліком же є висока чутливість до зміни освітлення, розміру і поворотів, і, як результат, необхідність суворого збереження вихідних умов зйомки.

Для роботи методу Eigen Faces необхідно додати в dataSetGenerator наступний рядок (лістинг 3.15), оскільки розміри зображення мають бути фіксованими при збереженні в DataSet.

Лістинг 3.15 – Resize

```
gray_face=cv2.resize(gray[y:y+h,x:x+w],(110,110));
```

Потім ми змінюємо в detector LBPH Recognaizer на EigenFace Recognizer

```
rec=cv2.createEigenFaceRecognizer();
```

Після цього виконуємо всі етапи, які були згадані в попередньому пункті, й отримуємо такий результат: (рисунок 3.7)

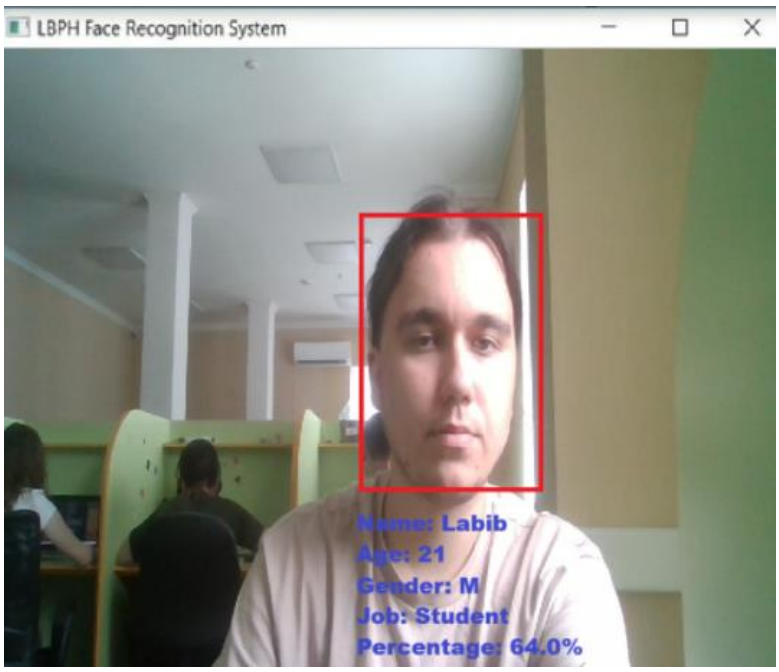


Рисунок 3.7 — Розпізнавання облич методом Eigenfaces

3.1.3 Метод Fisher Faces

Fisherfaces — алгоритм, у якому на відміну від методу eigenfaces використовується лінійний дискримінантний аналіз, а саме лінійний

дискримінант Фішера. Дія алгоритму ґрунтується на пошуку проекції даних, за якої класи зображень облич максимально розділені. При використанні ж методу головних компонент проводиться максимізація розкиду даних по всій базі облич. Ця відмінність дає змогу розв'язати проблему високої чутливості до змін освітлення.

Для фіксації розміру зображення при збереженні в DataSet додаємо наступний рядок (лістинг 3.16)

Лістинг 3.16 – Resize

```
gray_face=cv2.resize(gray[y:y+h,x:x+w],(110,110));
```

Змінюємо в detector LBPH Recognaizer на FisherFace Recognizer

```
rec=cv2.createFisherFaceRecognizer();
```

Повторюємо всі попередні етапи, які були в LBPH пункті, і отримуємо такий результат: (рисунок 3.8)

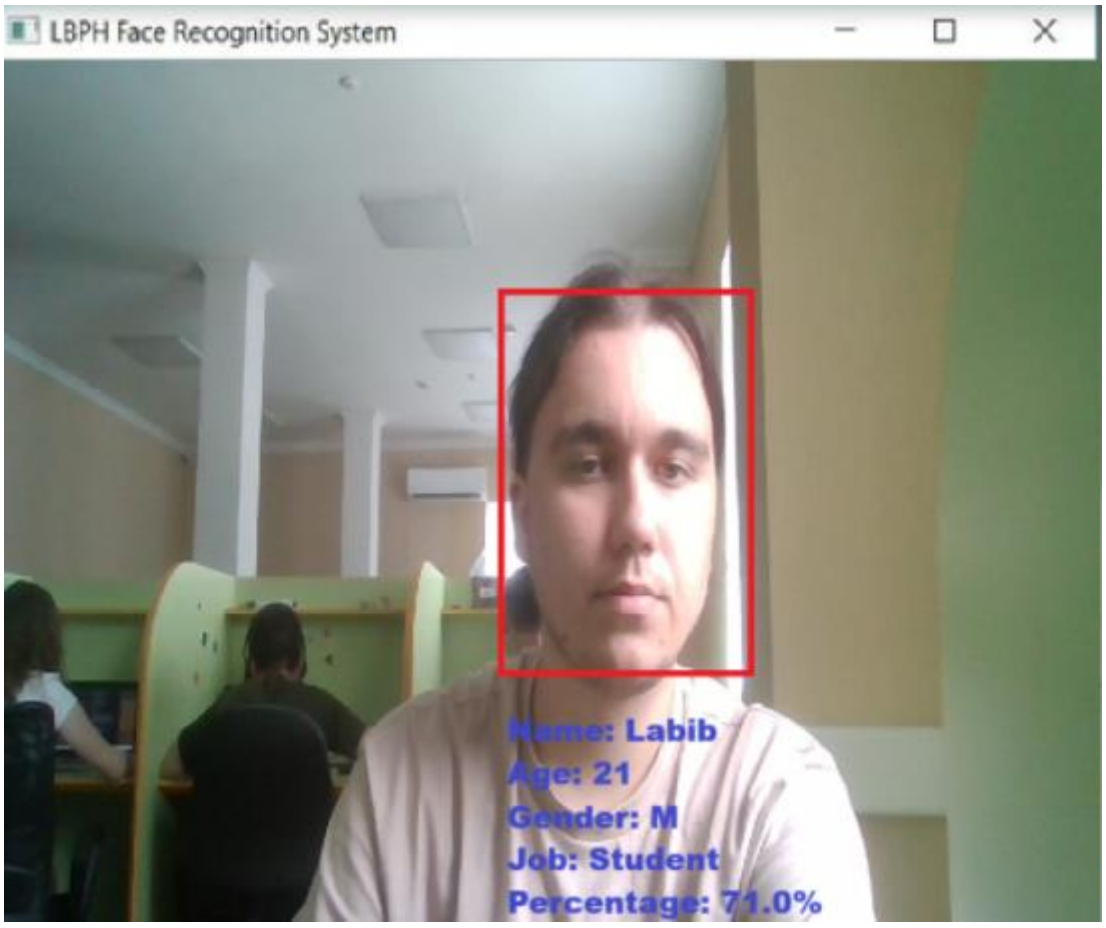


Рисунок 3.8 — Розпізнавання облич методом Fisherfaces

На таблиці (3.1) ми побачимо результати ймовірності методів розпізнавання облич.

Метод Fisherfaces краще розпізнає обличчя, ніж LBPH і Eigenfaces за необхідних умов. Якщо ж будуть змінені умови освітлення, то Fisherfaces і Eigenfaces починають некоректно працювати, на відміну від них LBPH працює коректно.

Усі три методи розпізнають обличчя за допомогою зіставлення обличчя людини, яку розпізнають, із навчальним набором даних, у якому збережені відомі нам люди. Перед методами було поставлено завдання розпізнавання облич і їх приналежність тій чи іншій людині.

Результати тестування показали, що метод Fisherfaces краще розпізнає обличчя, ніж LBPH і Eigenfaces за необхідних умов. Якщо ж будуть змінені умови освітлення, то Fisherfaces і Eigenfaces починають некоректно працювати, на відміну від них LBPH працює коректно.

Таблиця 3.1 — Імовірність розпізнавання облич

Методи	Точність розпізнавання %
LBPH	50%
Eigenfaces	64%
Fisherfaces	71%

Іноді програма визначає обличчя кількох людей як обличчя однієї людини, і це вказує на помилку розпізнавання і не ефективність методу

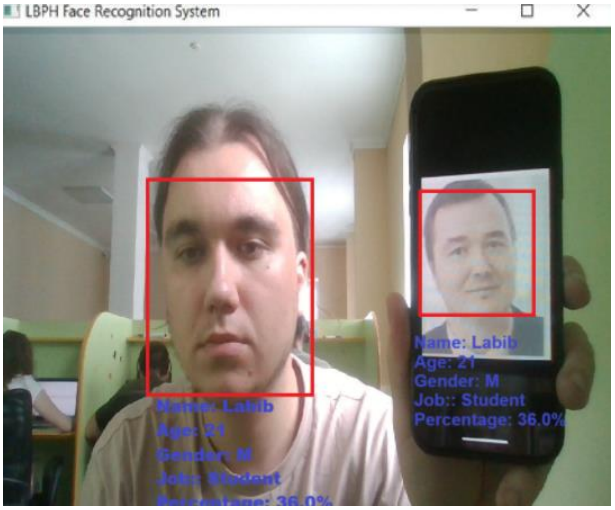
Також програма визначає кадри не-осіб як обличчя людини, що вказує на помилку розпізнавання і не ефективність методу.

Наступним тестом буде тестування для виявлення помилки розпізнавання облич. Якщо буде виявлено хоча б одну з вищевказаних помилок, це означає, що метод не є ефективним і ми не повинні його використовувати для створення нашої системи реєстрацій подій.

У першому тесті тестування будуть проводитися за участю облич двох різних людей. Програма повинна або розпізнати ці обличчя, як обличчя різних

людей, або детектувати їх. Якщо ж програма розпізнає ці обличчя, як обличчя однієї людини, то це свідчить про неефективність методу і не повинен використовуватися для створення системи реєстрацій подій.

Проводимо тест і бачимо такий результат. (рис. 3.9 і 3.10)



Рисунок— 3.9. Визначення двох різних людей як однієї людини метод Eigenface

Як ми бачимо з рис.3.10. і рис.3.9, що метод Eigenface показав під час тестування результати некоректного розпізнавання людей, і ми не можемо його використовувати для створення системи реєстрацій події.

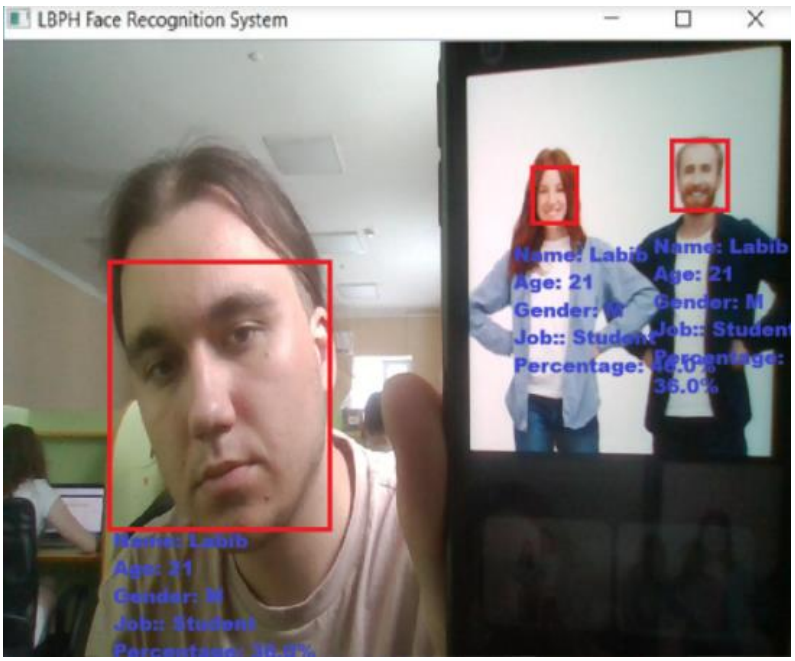


Рисунок 3.10 — Визначення трьох різних людей як однієї людини метод Eigenface

Тестування інших методів дало такі результати.

Метод LBPH розпізнає обличчя людей, але одного як обличчя відомої нам людини, а інше обличчя як невідомого. Метод Fisherfaces також має помилку в розпізнаванні, визначаючи обличчя однієї людини правильно, а іншої з бази даних неправильно. (рис. 3.11)

Рисунок 3.11 — Розпізнавання людини як
когось іще з бази даних метод Fisherfaces

Програма плутає людей і некоректно розпізнає цих людей, і ми також не рекомендуємо його для створення системи реєстрацій події.

Метод Eigenface не є хорошим алгоритмом для бібліотеки OpenCV, але ми продовжимо проводити тестування інших методів бібліотеки OpenCV.

Наступний тест — це розпізнавання обличчя за моєю фотографією, зробленою десять років тому.

Людина протягом десяти років змінює свій зовнішній вигляд і для звичайної людини іноді буває складно розпізнати її, але для того щоб навчити комп'ютер розпізнати, необхідні великі зусилля для програмування. Отже, чи зуміє програма розпізнати людину за фотографією десятирічної давності чи ні. Для тестування використовуємо всі методи.

Проводимо цей тест і будемо дивитися такі результати: (рис. 3.13 і 3.14 і 3.15)

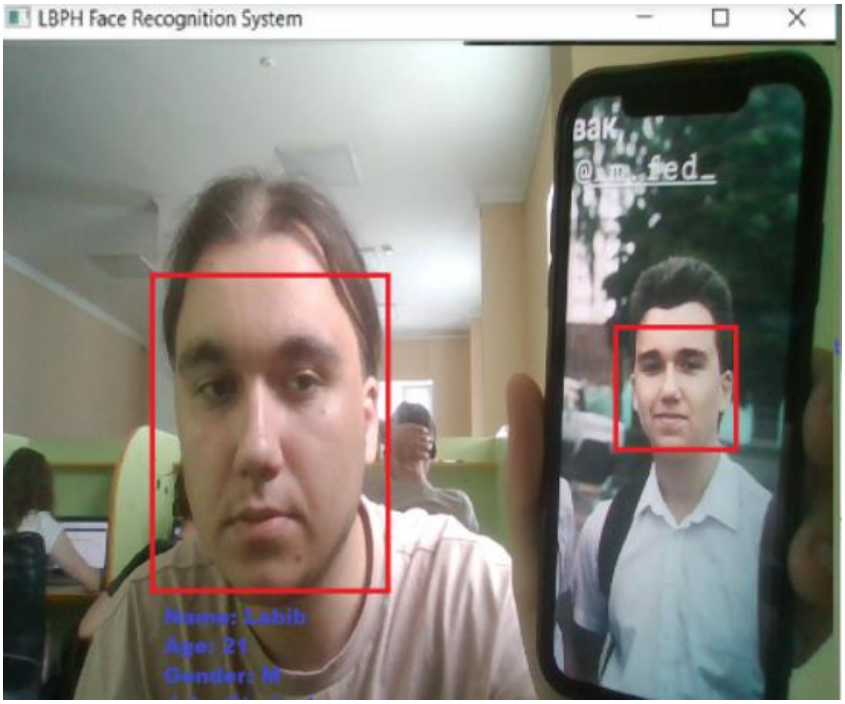


Рисунок 3.12 – Нездатність розпізнати обличчя користувача десять років тому метод LBPH

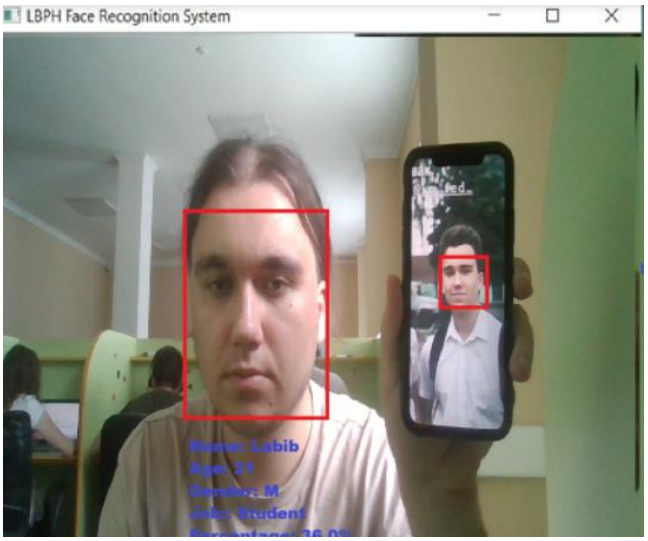


Рисунок 3.13 — Нездатність розпізнати обличчя користувача десять років тому метод метод метод Eigenface

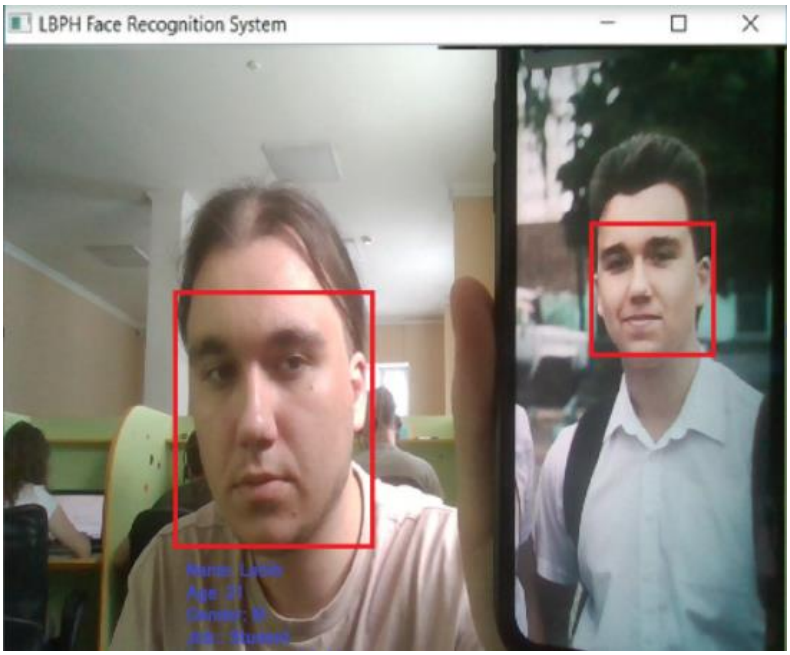


Рисунок 3.14 — Нездатність розпізнати обличчя користувача десять років тому метод Fisherfaces

Результати тестування показують, що методи неефективні і не змогли дати хороші результати, що привело нас до рішення не використовувати методи бібліотеки OpenCV для створення системи реєстрацій подій.

3.2 Бібліотека розпізнавання облич face_recognition

Розпізнавання облич (Face Recognition - англ.) — це один із найперспективніших методів біометричної безконтактної ідентифікації людини за обличчям.

Бібліотека face_recognition дає змогу розпізнавати і маніпулювати обличчями з Python або з командного рядка. Побудована з використанням бібліотеки dlib.

Метод пошуку працюватиме так:

— отримання прикладів зображень, навчання моделі на них;

— створення масивів відомих кодувань облич та їхніх імен;

— отримання кадру з веб-камери;

— визначення всіх облич і кодування облич у поточному кадрі відео;

— зіставлення облич відомому набору;

— отримання результатів збігів;

— визначення рамки навколо обличчя та мітки з ім'ям під обличчям;

— виведення отриманого зображення.

Створюємо папку dataSet і вставляємо фотографії облич в одному екземплярі в dataSet. (рис. 3.16)

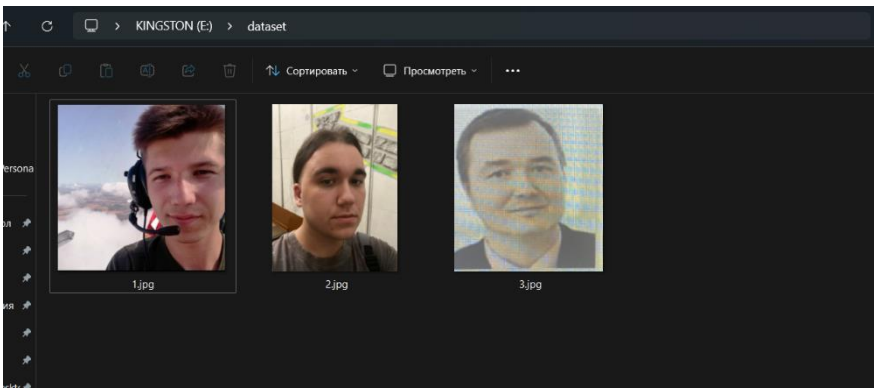


Рисунок 3.15 — DataSet

Якщо у вас до цього були поставлені бібліотеки python, а саме:

— NumPy;

— OpenCV-2.

То програма запуститься коректно і працюватиме без помилок.

Щойно буде натиснуто кнопку «Run», алгоритм почне свою роботу. Насамперед програма рахує кількість папок (користувачів), кількість фотографій у кожній з них, а потім наявність облич на запропонованих програмі фотографіях.

Після аналізу зображення, алгоритм почне своє навчання, а потім виведе зображення з камери пк. На відеопотоці з камери можна буде протестувати працездатність алгоритму, показуючи відомі йому обличчя, і обличчя, які не були завантажені в програму. У другому випадку програма позначить ці обличчя як «Unknown», а якщо розпізнавання пройшло коректно, то на екрані, в зеленій рамці, з'явиться ім'я ідентифікованої особи. Запускаємо код і отримуємо такі результати: (рисунок 3.17 і рисунок 3.18)

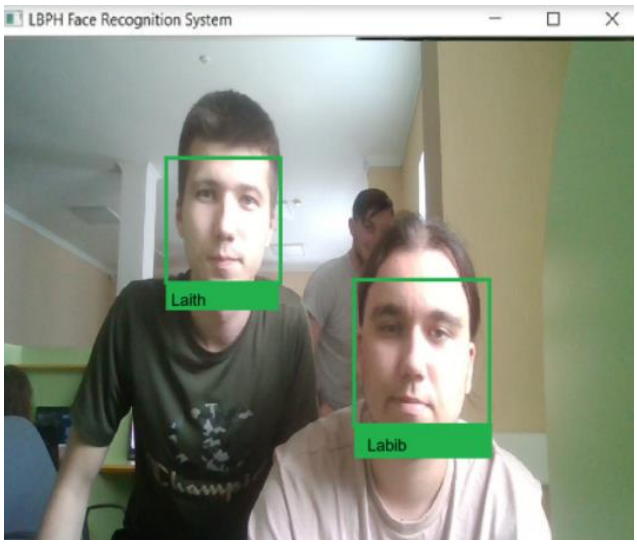


Рисунок 3.17 — Результат розпізнавання облич (1)

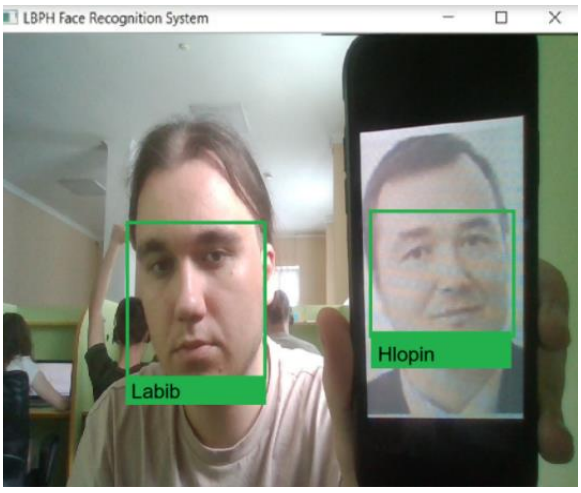


Рисунок 3.18 — Результат розпізнавання облич (2)

Робимо тестування бібліотеки можемо з невідомою особою, якої немає в dataSet, і отримуємо такий результат: (рисунок 3.19)

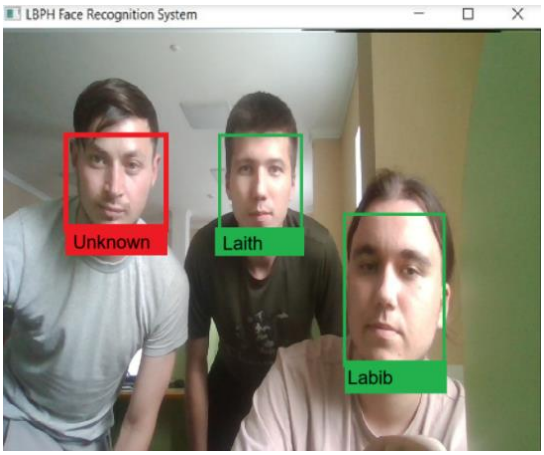


Рисунок 3.19 — Результат розпізнавання облич із невідомим обличчям, якого немає в dataSet.

Як показують результати, під час використання бібліотеки Face_Recognition ймовірність і точність розпізнавання висока і зміни умов освітлення не впливають на результати розпізнавання. Порівнюємо результати тестування бібліотеки Open CV та Face_Recognition і обираємо кращу. Тому для реалізації системи реєстрацій подій ми будемо використовувати бібліотеку Face_Recognition.

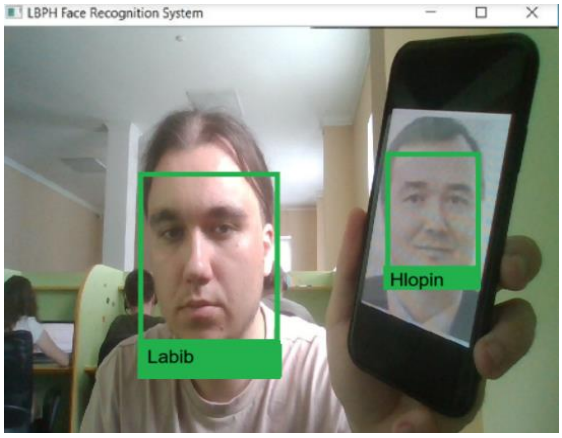


Рисунок 3.20 — Визначення двох різних облич як різних людей.

Результат дуже хороший, тому що програма точно визначила обличчя двох різних людей.

Наступний тест — розпізнавання людини за фотографією, знятою десять років тому. Як ми знаємо, за десять років людина змінює свій зовнішній вигляд і іноді люди не можуть її розпізнати.

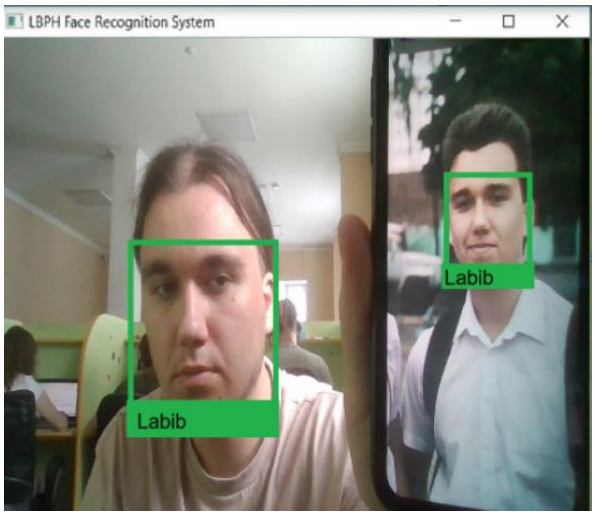


Рисунок 3.21 — Здатність розпізнати обличчя
користувача десять років тому

Отримані результати дуже хороші, і це означає про ефективність бібліотеки Face Recognition і рекомендується її використовувати для створення системи реєстрації події.

3.3 Створення системи реєстрацій подій на основі аналізу відеозображень

Для реалізацій системи будуть використані:

- бібліотека Face_Recognition;
- база даних Sqlite3;
- бібліотека Open CV;
- функція часу datetime;
- мова програмування високого рівня Python.

У цьому підрозділі я буду реалізувати систему реєстрації події, система повинна виконувати такі функції:

— отримання відеопотоку з камери (камер);

— детектування облич у кадрі відеопотоку;

— пошук відповідності знайдених облич знайденим детектованим обличчям із відеопотоку;

— у разі детектування нового обличчя людини програма повинна вирізати і зберегти обличчя невідомої людини в базу даних;

— під час детектування відомої людини програма має зберігати час і дату її останнього виявлення.

Під час увімкнення камери програма має отримувати відеопотоки з камери, і камера має бути постійно увімкнена, щоб ми могли перейти до наступного етапу — це детектування облич.

На наступному етапі програма повинна детектувати обличчя, які потрапляють у кадрі.

Під час детектування нового обличчя людини програма повинна вирізати і зберегти обличчя невідомої людини в базу даних програма повинна реєструвати дату і час виявленого невідомого обличчя під час останнього його або її виявлення.

Після збереження невідомої виявленої особи в нашій системі реєстрації подій, розглядаємо його або її фотографію, яка була збережена під час виявлення, і якщо ця особа відома для користувача, то ми додаємо цю особу як нового відомого користувача.

Під час детектування відомої людини програма має зберігати час і дату її останнього виявлення. Обмеження часу при тестуванні буде 1 хвилина, але можна збільшити це обмеження на більший проміжок часу як година, щоб легше проводити експерименти і тестування було вирішено.

Перший крок у програмі — це читання з бази даних зображення та ім'я людини, як показано на рисунок 3.14.

Далі, після читання програма детектуватиме і розпізнаватиме обличчя і потім реєструватиме події, як показано на рисунку 3.15.

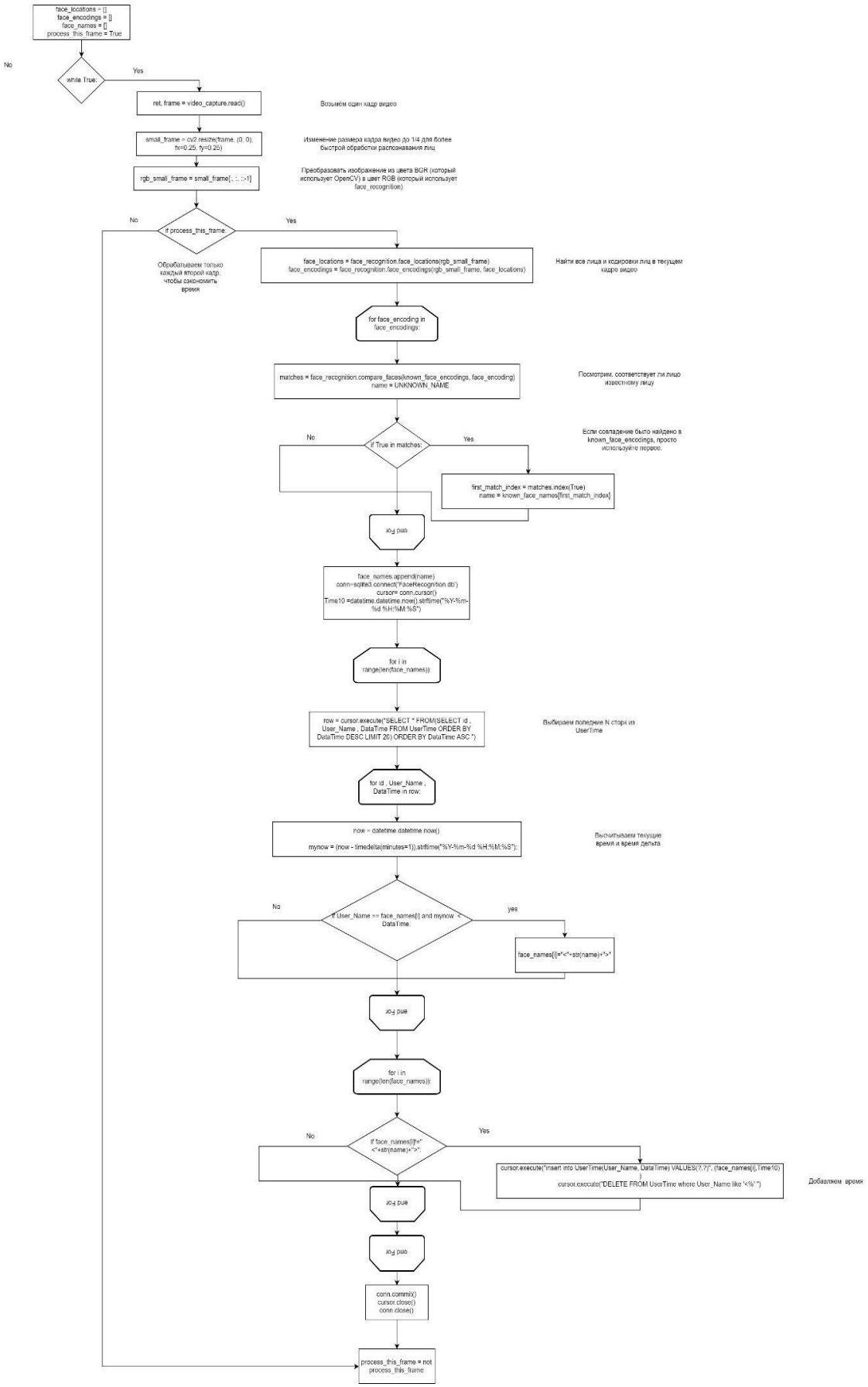


Рисунок 3.15 — Детектування, розпізнавання облич і реєстрація події

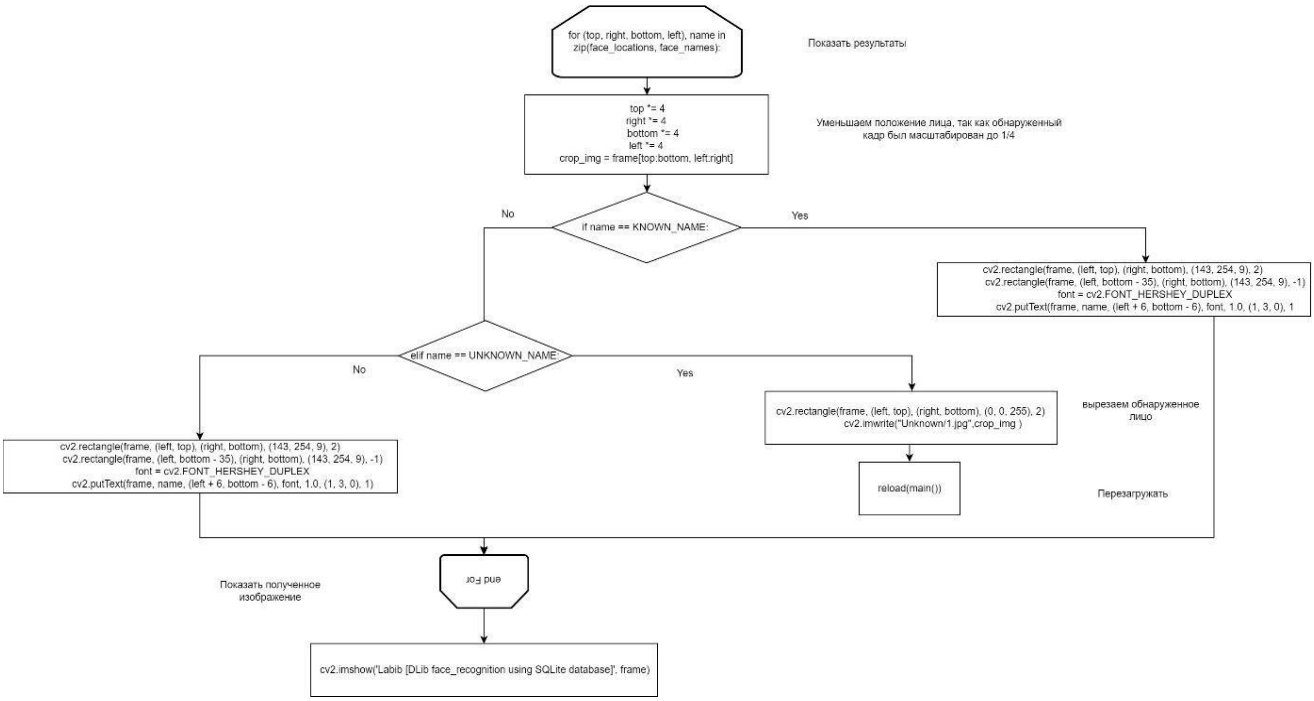


Рис.3.16. Виведення результатів

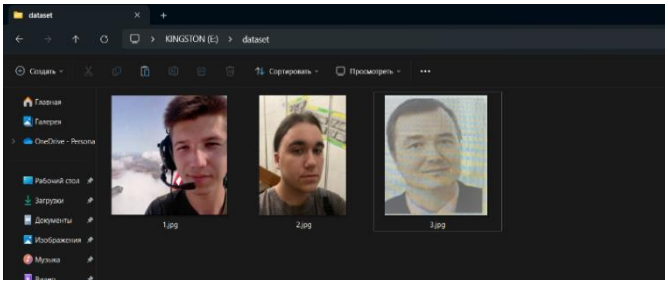
Для реалізації програми використовуємо бібліотеку Face_Recognition. Починаємо з реалізації алгоритму розпізнавання. Для цього завантажуємо фотографії з папки dataSet1 у базу даних sqlite3, використовуючи такий алгоритм. (Лістинг 3.17)

Лістинг 3.17 — Завантаження фото з dataSet1 в sqlite3

```
PATH_IMAGES = 'dataSet1/
#--- Extract files from folder following

pattern files  =

glob.glob(PATH_IMAGES+"*.jpg")

n_files = len(files)

print('Number of files in folder: ', n_files)

name = 'Face Not Known'
```

```
#--- Simple database

creator def

create_db(filename):

   db =

   sqlite3.connect(filename)

   cursor = db.cursor()

   cursor.execute("DROP TABLE IF EXISTS FaceImages")
   cursor.execute("CREATE TABLE FaceImages(ID INTEGER PRIMARY
KEY AUTOINCREMENT,ObjId INTEGER, img BLOB, size INT, Name TEXT)")
   db.com

   mit()

   db.close

   ()

filename_db = 'FaceRecognition.db'

create_db(filename_db)

#--- Open database and loop over files to insert in

database con = sqlite3.connect(filename_db)

cur = con.cursor()
for i, file_i in enumerate(files):
   #--- Read image as a binary

   blob with open(file_i, 'rb') as

   f:

      image_bytes =

   f.read() f.close()

   #--- Decode raw bytes to get image size
```

```
        nparr  = np.fromstring(image_bytes, np.uint8)

        img_np = cv2.imdecode(nparr,

        cv2.IMREAD_COLOR)image_size =

        img_np.shape[1]


        #--- Extract file name without extension

        filename = os.path.relpath(file_i,

        PATH_IMAGES) objid =

        int(os.path.splitext(filename)[0])


        #--- Insert image and data into table
        cur.execute("insert into FaceImages(ObjId , img ,size , Name)
VALUES(?,?,?,?)", (objid,sqlite3.Binary(image_bytes),image_size,name)  )
        #--- Cheap

        progress if ((i %

        100) == 0):

            print(i,

    n_files)

    con.commit()

    cur.close()

    con.close()
```



Рису.3.17. dataSet1

Результатом успішного запуску скрипта отримуємо такий результат. (рис. 3.18)

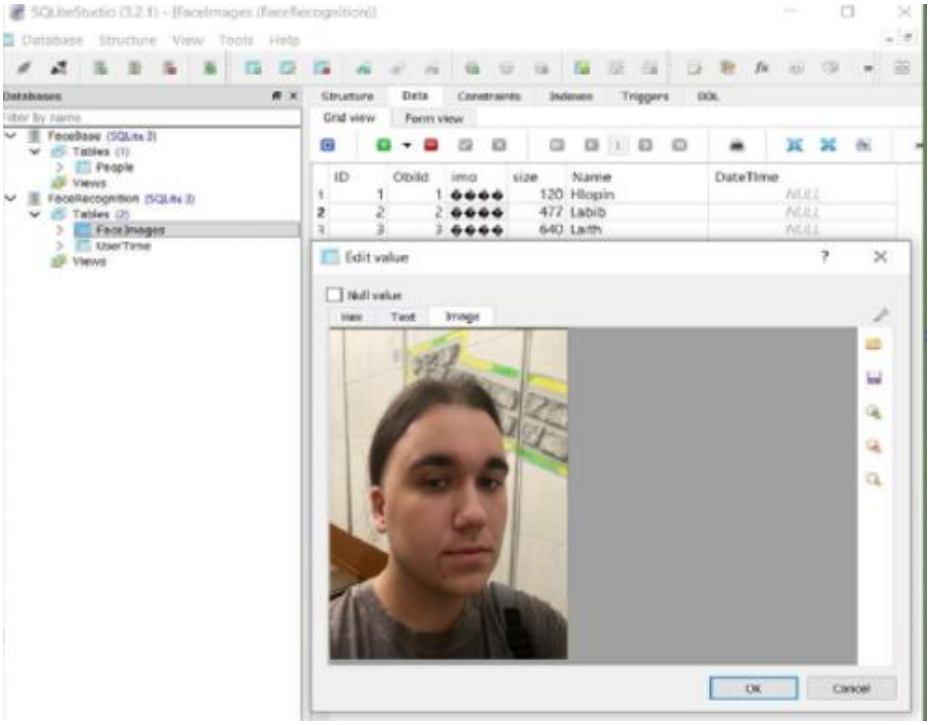


Рисунок 3.18 — Додавання фото в базу даних sqlite3

Потім знаходимо спосіб читання фото та імена з бази даних і додаємо їх у масиви My_Face1_face_encoding і known_face_names. (Лістинг 3.18)

Лістинг 3.18 — Читання фото та імена з бази даних

```
row = cursor.execute("""SELECT ObjId, img from FaceImages""")
    for ObjId, img  in row:
        nparr  = np.fromstring(img , np.uint8)
        img_np = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        encoding = face_recognition.face_encodings(img_np)[0];
        My_Face1_face_encoding.append(encoding);
    row = cursor.execute("""SELECT ObjId, Name from FaceImages""")
    for  ObjId, Name in row:
known_face_names.append(Name)
```

Далі, знаходимо всі обличчя і кодування облич у поточному кадрі відео. Лістинг (3.19)

Лістинг 3.19 — Знайти всі обличчя та кодування облич у поточному кадрі відео.

```
if process_this_frame:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
```

Починаємо порівнювати обличчя і дивимося, чи відповідає обличчя відомим нам обличчям, і воно збіглося, використовуємо перший вибір. (Лістинг 3.20)

Лістинг 3.20 — пошук обличчя

```
for face_encoding in face_encodings:
        matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
        name = UNKNOWN_NAME
        if True in matches:
first_match_index = matches.index(True) name =
known_face_names[first_match_index
```

Тепер переходимо до додавання часу до реєстрації подій. Як варіант для прототипу робимо обмеження дельта часу на одну хвилину, можна і більше, але для зручності ми зробимо так. Це означає, що в разі виявлення відомого нам користувача, програма має зареєструвати його щохвилини. (Лістинг 3.21)

Лістинг 3.21 — Додавання часу реєстрацій події

```
for i in range(len(face_names)):
        row = cursor.execute("SELECT * FROM(SELECT id , User_Name ,
DataTime FROM UserTime ORDER BY DataTime DESC LIMIT 20) ORDER BY
DataTime ASC ")
        for id , User_Name , DataTime in row:
```

```
            now = datetime.datetime.now()
            mynow = (now - timedelta(minutes=1)).strftime("%Y-%m-%d
            %H:%M:%S"); if User_Name == face_names[i] and mynow <
            DataTime:
                print (User_Name);
                face_names[i]="<"+str(name)
                +">"
        for i in range(len(face_names)):
            if face_names[i]!="<"+str(name)+">":
                cursor.execute("insert into UserTime(User_Name,
                                DataTime) VALUES(?,?)",
(face_names[i],Time10) )
```

Далі, у разі виявлення невідомого обличчя людини програма вирізає обличчя цієї людини і зберігає в базі даних. (Лістинг 3.22)

Лістинг 3.22 - Детектування невідомої людини та збереження в базі даних.

```
crop_img = frame[top:bottom, left:right]
        if name == KNOWN_NAME:
            #insert_prefix2(frame, prefix2, (top, bottom, left, right)) cv2.rectangle(frame,
            (left, top), (right, bottom), (143, 254, 9), 2)
            cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (143, 254, 9), -1)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (1, 3, 0), 1) elif
        name == UNKNOWN_NAME:
            # Draw a box around the face
```

```
cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

cv2.imwrite("Unknown/1.jpg",crop_img )

reload(main())

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), -1)

font = cv2.FONT_HERSHEY_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (1, 3, 0), 1)

else:

#insert_prefix(frame, prefix, (top, bottom, left, right))

cv2.rectangle(frame, (left, top), (right, bottom), (143, 254, 9), 2)

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (143, 254, 9), -1) font

= cv2.FONT_HERSHEY_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (1, 3, 0), 1)
```

Після успішного запуску коду програма має виконувати такі функції:

1 Пошук відповідності знайдених осіб детектованим особам із відеопотоку проводимо за допомогою бази даних sqlite3. (рис. 3.20, 3.21, 3.22)

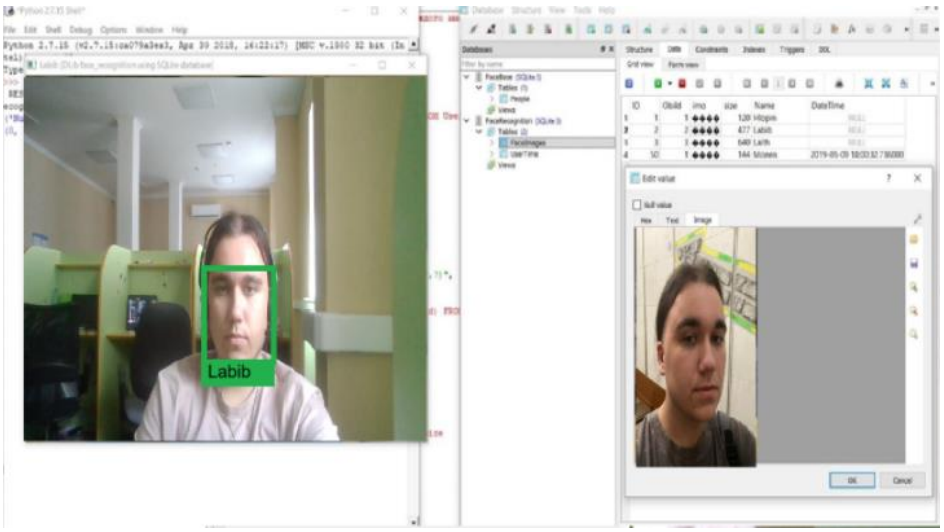


Рисунок 3.20 — Розпізнавання мого обличчя

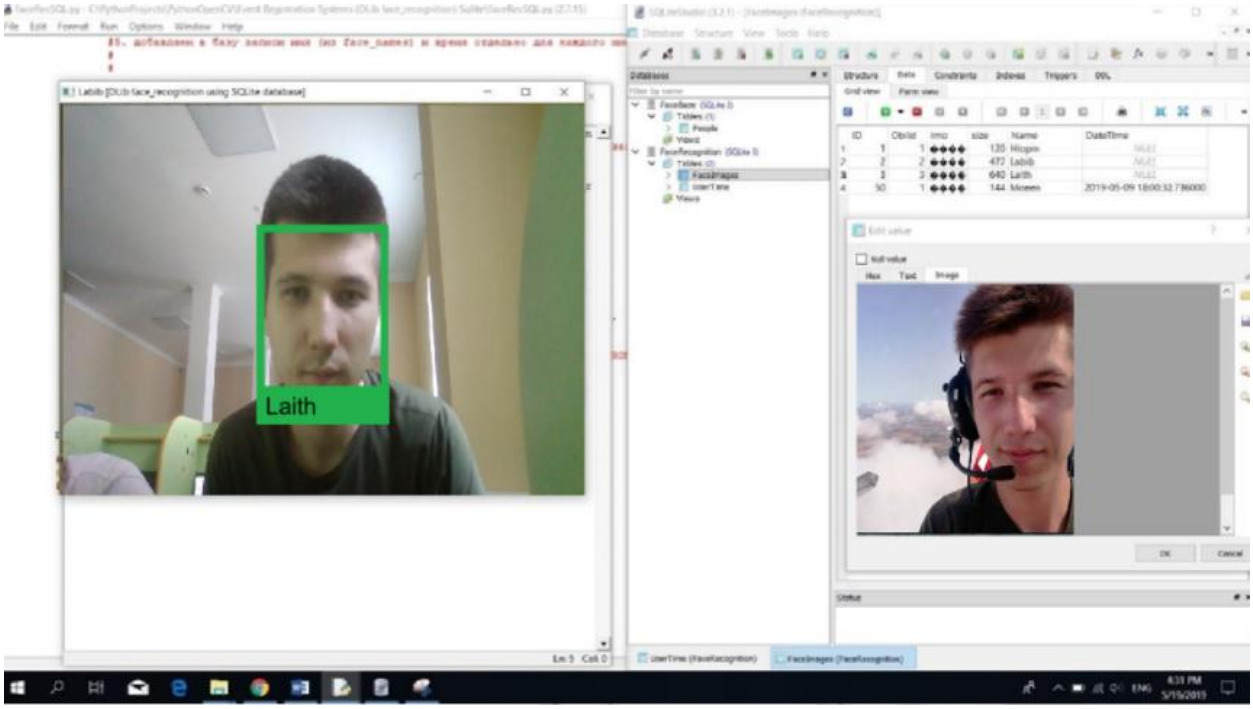


Рисунок 3.21 — Розпізнавання обличчя людини з бази даних

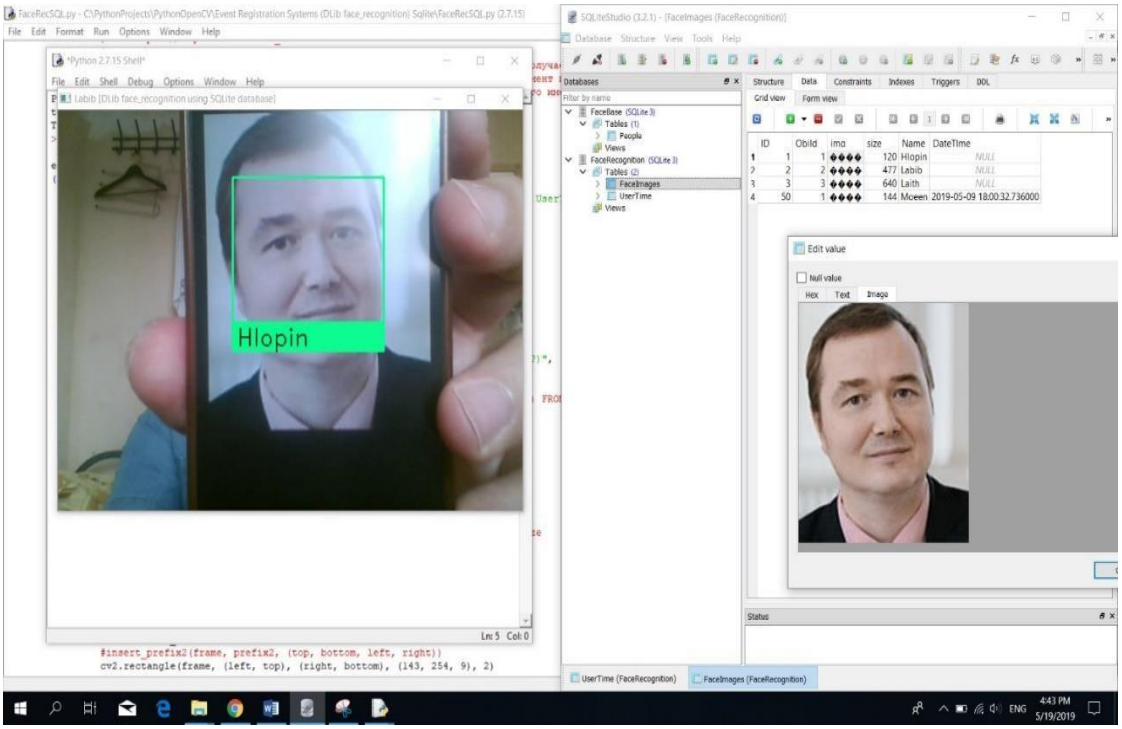


Рисунок 3.22 — Розпізнавання обличчя людини з бази даних

2 Під час детектування відомої людини програма зберігає час і дату її останнього виявлення. (рис. 3.23 і 3.24)

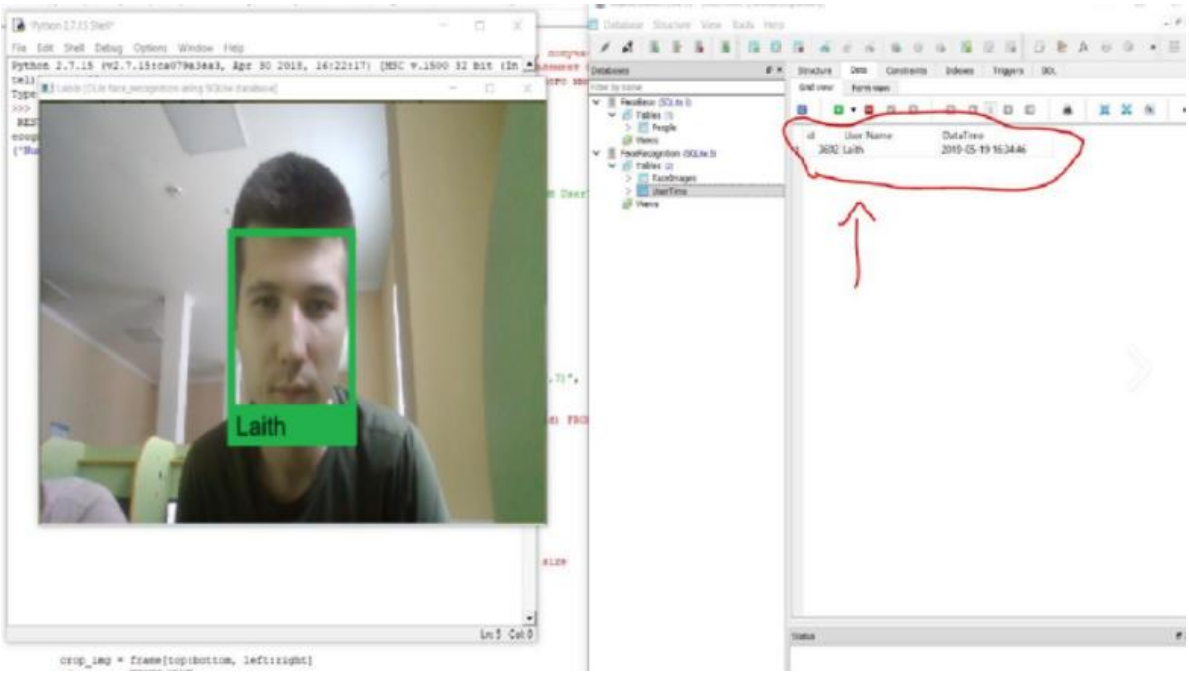


Рисунок 3.23 — Реєстрація подій відвідуваності користувачів

З Під час детектування нового обличчя людини програма вирізає і зберігає його обличчя в базі даних. (рис. 3.25)

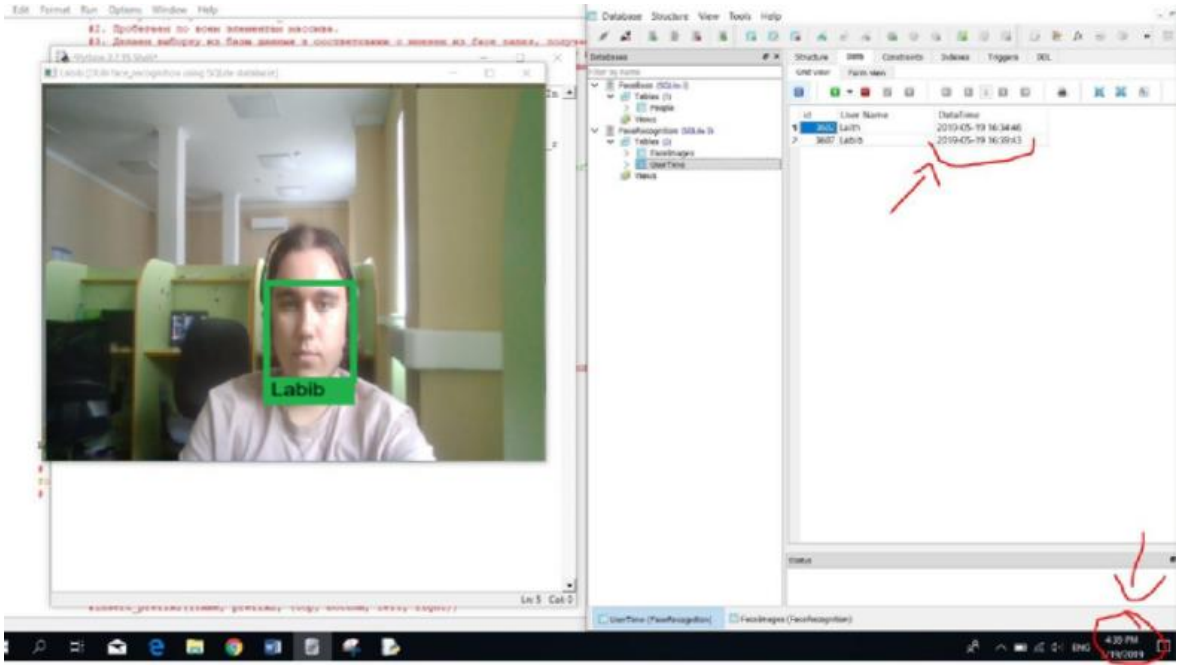


Рисунок 3.24 — Реєстрація подій відвідуваності користувачів

Після успішної реалізації моєї системи реєстрації події на основі аналізу відеозображення я думаю, що ця система ефективна і може бути застосована, бо вона розпізнає відомих осіб, додає або реєструє цих відомих осіб або відомих

користувачів, реєструє час і дату їхньої присутності за обмеження в 1 хвилину для тестування. Ці обмеження можна збільшити і встановити час, який є необхідним для визначення і реєстрування подій.

3.4 Розробка системи реєстрації на основі аналізу відеозображень

Цей розділ присвячений проведенню експерименту та тестуванню створеної програми. Ми побачимо її роботу і можливості детектування та розпізнавання облич.

Тестування відбуватиметься за різних умов. Для першого тесту, будуть застосовані дві умови:

— детектування облич при поганому освітленні;

— детектування облич при хорошому освітленні.

Кількість проходжень облич відзначатиметься через кожен кадр. Ми будемо дивитися кількість виявлень обличчя за кожного кадру, тобто скільки разів програма знаходитиме обличчя за кожного кадру. Потім обчислюємо кількість виявлень обличчя на кожні 100 кадрів за допомогою такої формули (4.1.):

$$\frac{\square\square\square\square\square\square\square\square\square}{\square\square\square\square\square\square} \qquad (4.1)$$

Далі додаємо нову умову — це колір шкіри. Тепер тестування відбуватиметься за наявності таких умов:

— при різному освітленні;

— і з різним кольором шкіри.

Перший учасник експерименту (користувач 1) — це буде людина з темною шкірою. Наша програма повинна буде детектувати і розпізнавати її за хорошого і поганого освітлення і далі порахувати ймовірність детектування і розпізнавання за формулою (4.1) Результат детектування і розпізнавання обличчя людини з темною шкірою за поганого освітлення показано на рисунку 4.1.

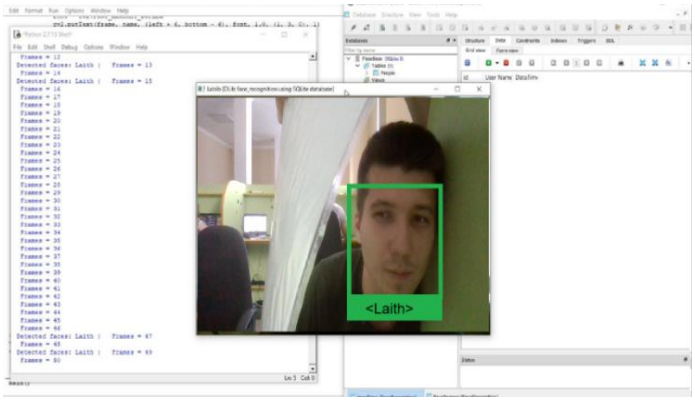


Рисунок 4.1 — Детектування та розпізнавання облич за поганого освітлення (користувач 1)

Використовуючи формулу (4.1), знаходимо ймовірність розпізнавання.

$$\frac{\text{кількість кадрів}}{\text{кадрів}} = \frac{11}{100} = 0{,}11 = 11\%$$

Тепер тестування відбуватиметься за хорошого освітлення для людини з темною шкірою. Результат детектування та розпізнавання для людини з темною шкірою за хорошого освітлення показано на рисунку 4.2.

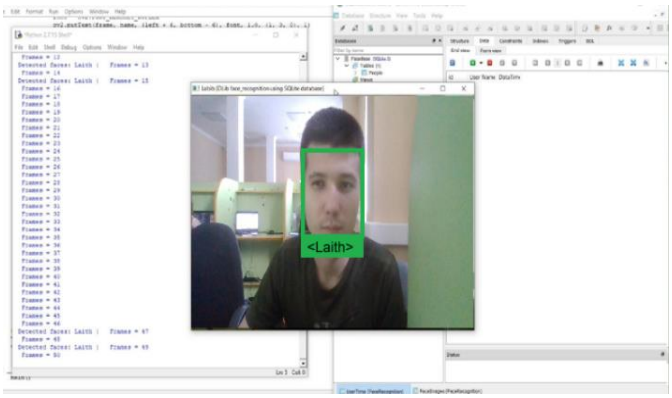


Рисунок 4.2 — Детектування та розпізнавання облич за хорошого освітлення (користувач 1)

Використовуючи формулу (4.1), знаходимо ймовірність розпізнавання.

$$\frac{\text{кількість кадрів}}{\text{кадрів}} = \frac{21}{100} = 0{,}21 = 21\%$$

Другий учасник експерименту — це людина зі світлою шкірою (користувач 2). Проведемо ті самі етапи тестування й отримуємо такі результати.

Результати детектування і розпізнавання для людини зі світлою шкірою за поганого освітлення показано на рисунку 3.4.

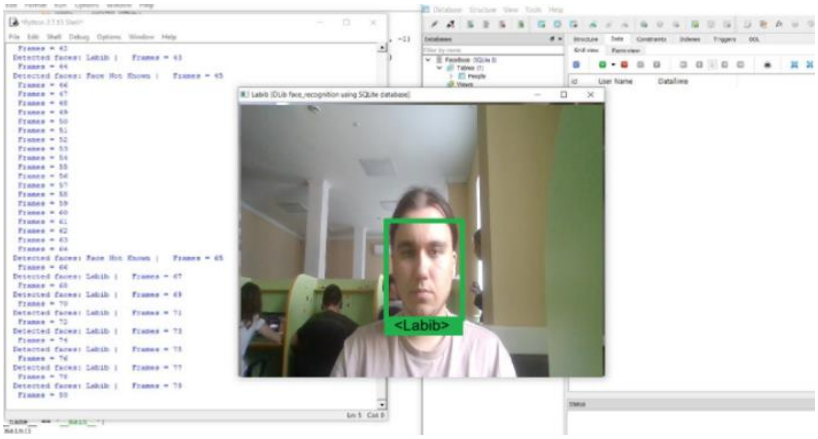


Рисунок 3.4 — Детектування та розпізнавання облич при поганому освітленні (користувач 2)

Використовуючи формулу (4.1), знаходимо ймовірність розпізнавання.

$$\frac{\text{кількість кадрів}}{\text{загальне}} = \frac{33}{100} = 0{,}33 = 33\%$$

Далі, детектування та розпізнавання людини зі світлою шкірою відбуватиметься за хорошого освітлення. Результат ми бачимо на рисунку 4.4.

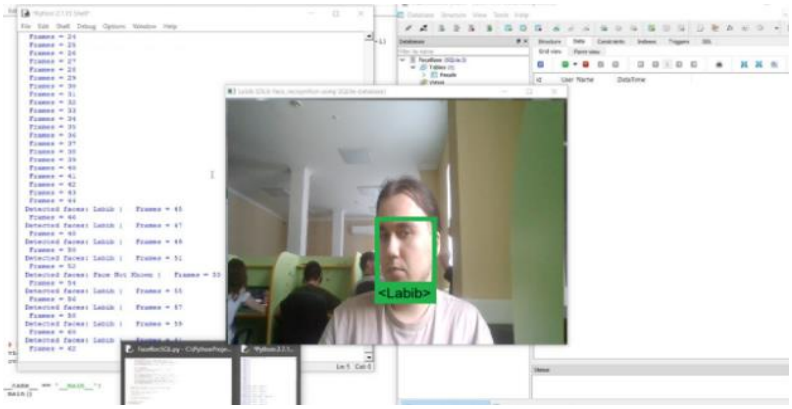


Рисунок 4.4 — Детектування та розпізнавання облич за хорошого освітлення (користувач 2)

Використовуючи формулу (4.1), знаходимо ймовірність розпізнавання.

$$\text{Точність виявлень} = \frac{44}{100} = 0{,}44 = 44\%$$

Після того, як було протестовано ймовірність розпізнавання в домашніх умовах, ми переходимо до тестування ймовірності детектування та розпізнавання в нових і найкращих умовах. Тестування відбуватиметься в Науковому Дослідницькому корпусі з використанням камери Logitech HD 1080p і хорошому освітленні. Запускаємо програму й отримуємо такі результати. (рис.4.5)

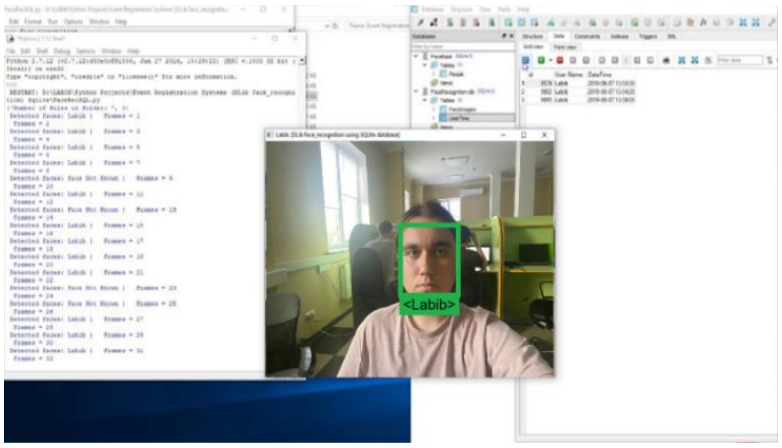


Рисунок 4.5 — Детектування та розпізнавання облич за хорошого освітлення (користувач 2)

Використовуючи формулу (4.1), знаходимо ймовірність розпізнавання.

$$\text{Точність виявлень} = \frac{56}{111} = 0{,}504 = 50{,}4\%$$

Порівняння точності виявлень і ймовірності детектування та розпізнавання облич за різних умов і використання різних камер наведено в таблиці 4.1. Результати свідчать, що якість камери та освітлення відіграють більшу роль для поліпшення детектування та розпізнавання облич.

Таблиця 4.1 – Порівняння ймовірності детектування та розпізнавання облич

Тип камери та освітлення	Користувач 1	Користувач 2
BisonCam , NB Pro (хороше освітлення)	21%	44%
BisonCam , NB Pro (погане освітлення)	11%	33%
HD 1080p logitech (хороше освітлення)	Не брав участі в експерименті	50,4%

Як ми бачимо з таблиці, що програма добре розпізнає обличчя людини, і за хороших умов освітлення та високої якості камери програма розпізнає обличчя з імовірністю 50 %.

50 % показує, що програма розпізнає обличчя за хорошого кадру тільки, а погані кадри вона не детектує, оскільки обличчя може бути розмазане або не визначене, тому 50 % це розпізнавання хороших кадрів.

Нашим наступним завданням буде розгляд коректності детектування виявлення невідомих облич і пошук цих облич у базі даних для розпізнавання. На рисунку 4.6 ми бачимо, як програма під час детектування невідомих облич, спочатку вирізає ці обличчя і потім зберігає їх у базу даних sqlite3.

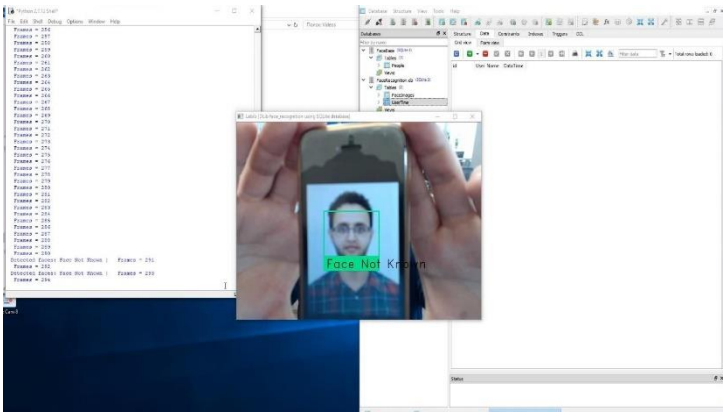


Рисунок 4.6 — Виявлення невідомої особи.

Під час додавання невідомої особи як відомого користувача і під час подальшого детектування невідомої іншої особи, інколи програма помиляється і

визначає невідому іншу особу як відомого користувача, який був збережений у базі даних. (рис. 4.7 і 4.8)

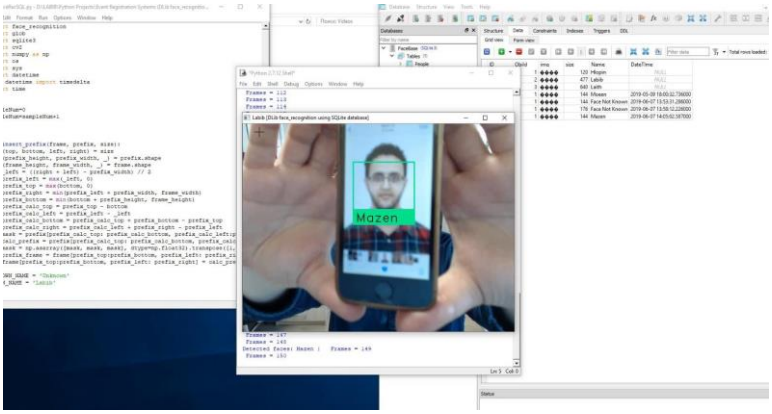


Рисунок 4.7 — Виявлення відомої особи

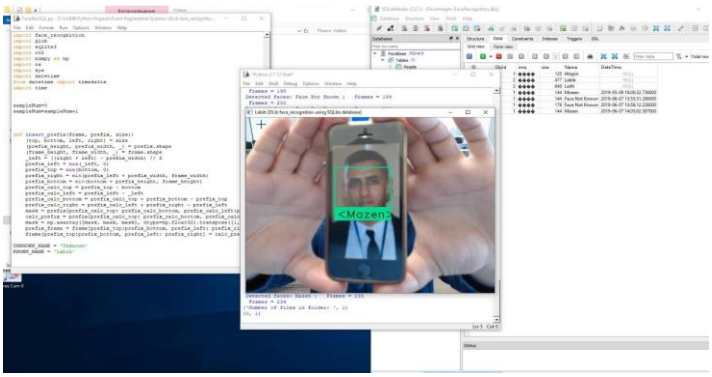


Рисунок 4.8 — Розпізнавання невідомої особи як відомого користувача

Ці обличчя генерується нейронною мережею. Для тестування цих облич моєю системою буде використовуватися сайт https://thispersondoesnotexist.com/ (рис. 4.9)

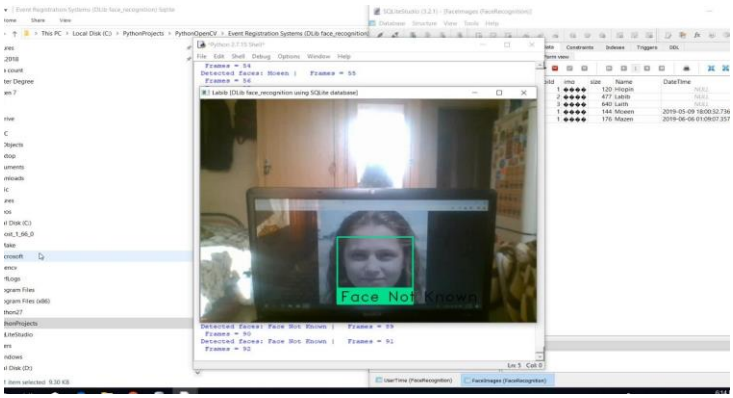


Рисунок 4.9 — Обличчя неіснуючої людини

Під час тестування програмі подаватимуться 100 випадкових генеруючих осіб нейронної мережі і потім отримуємо та розглядаємо результати, які описані в таблиці 4.2.

Таблиця 4.2 — Тест облич неіснуючих людей

	Виявлено та збережено в базі даних	Не виявлено	Виявлено як Face Not Known
Кількість виявлення	41	9	50

Як видно з таблиці 4.2, що кількість виявлення і збережених у базі даних - це 41 раз. Виявлення, яке було визнане як Face Not Known — це 50 разів і з них було розпізнано дві особи, як користувач Labib і користувач Moeen (рис. 4.10 і 4.11).

Результати тестування 100 випадкових генерованих нейронною мережею осіб були неочікувані, тому що програма розпізнала невідомих осіб і додала їх до бази даних, і цих осіб було лише 50 і 100 осіб.

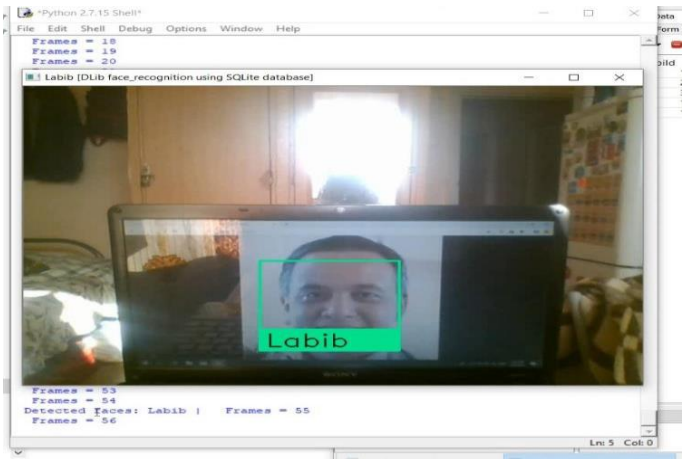


Рисунок 4.10 — Розпізнано як користувача labib

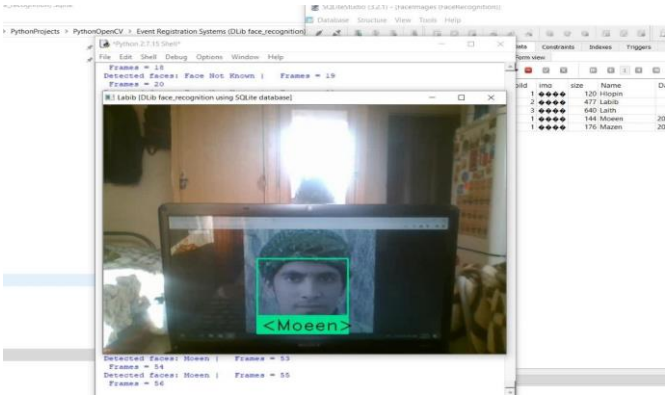


Рисунок 4.11 — Розпізнано як користувача Moeen

Після проведення тесту моєї системи реєстрації події я переконався, що моя система працює коректно і показує хороші результати. Я думаю, що цю систему реєстрації події можна застосувати в реальному світі і ця система вже була протестована за різних умов.

Але щоб переконатися в роботі системи, я повинен зробити аналіз результатів, і це буде розглядатися в наступному розділі.

3.5 Аналіз триманих результатів і рекомендації щодо їхнього розвитку

Після проведення аналізу можемо сказати, що результати тестування роботи системи реєстрації події позитивні, оскільки вона виконує всі поставлені перед нею завдання. І ми пропонуємо її для застосування в реальному житті: наприклад, на підприємствах, школах і ВУЗ для автоматизованої реєстрації відвідуваності, оскільки система реєструє дату і час під час виявлення і розпізнавання облич користувачів.

Розроблена система показує хороші результати, але вона не ідеальна система, тому що під час аналізу і тестування було виявлено деякі недоліки.

Такі системи реєстрацій подій після подальшого розроблення можуть знайти застосування у сфері безпеки та захисту, наприклад, у метро, поліції та інших громадських місцях.

Система розпізнавання облич, як будь-яка технологія, має свої позитивні та негативні сторони. З одного боку, вона забезпечує нам безпеку і допомагає

поліції або правоохоронним органам у пошуку правопорушників, з іншого боку страждає наша конфіденційність.

Моя система реєстрацій події допоможе поліпшити відвідуваність співробітників, оскільки вона не тільки розпізнає обличчя співробітників, але й додає до події дату сьогоднішнього дня і фіксує час приходу і відходу.

Система також збільшує безпеку компанії, оскільки в разі виявлення невідомих осіб вона зберігає ці обличчя в базу даних і якщо трапиться якась аварійна ситуація, то ми можемо визначити цих людей через базу даних відомих і невідомих осіб, які були або були присутні в цей день.

ВИСНОВКИ

Було проведено дослідження методів розпізнавання облич на відеозображеннях із використанням згорткової нейронної мережі (CNN). Для виконання роботи застосовували мову програмування Python та бібліотеки NumPy, PIL, OpenCV, Face_Recognition, а також базу даних SQLite.

Під час дослідження було досягнуто таких результатів:

— проведено огляд методів виявлення і розпізнавання облич;

— проаналізовано завдання розпізнавання облич;

— обрано інструментарій та вивчено особливості його застосування;

— реалізовано і протестовано обрані методи;

— виконано порівняльний аналіз між методами та бібліотеками OpenCV і Face_Recognition;

— розроблено систему реєстрації подій на основі аналізу відеозображень.

Тестування методів розпізнавання облич показало, що метод Fisherfaces краще розпізнає обличчя, ніж LBPH і Eigenfaces, за належних умов освітлення. Проте, якщо умови освітлення змінюються, то Fisherfaces і Eigenfaces працюють некоректно, тоді як LBPH залишається стабільним.

Для подальшого розвитку системи та її застосування було обрано бібліотеку Face_Recognition, оскільки її результати показали вищу ймовірність і точність розпізнавання облич, які не залежать від змін умов освітлення.

Також проведено тестування точності детектування та розпізнавання облич за різних умов освітлення (погане і хороше освітлення). Результати тестування показали, що освітлення та якість камери відіграють важливу роль у процесі детектування та розпізнавання облич.

Розроблена система може бути використана для розв'язання різних завдань відеоаналітики, особливо в системах контролю доступу та ідентифікації особи. Основним напрямком подальшого розвитку системи є вдосконалення класифікатора осіб, для чого доцільно замінити існуючі алгоритми класифікації на більш досконалі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шапіро Л. Комп'ютерний зір / К.: БІНОМ. Лабораторія знань, 2013. — 752 с.
2. Форсайт Д., Понс Ж. Комп'ютерний зір. Сучасний підхід: Пер. з англ. - К.: Видавничий дім Вільямс, 2004. - 928 с.
3. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features / 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. 1. 8–14 December 2001 / The Institute of Electrical and Electronics Engineers, Inc. C. 511–518.
4. Зір роботів: Пер. з англ. – К.: Світ, 1989. – 487 с., іл. ISBN 5-03-000570-6
5. Freund Y., Schapire R. E. A Short Introduction to Boosting // Journal of Japanese Society for Artificial Intelligence. 1999, №14(5), С. 771-780.
6. Roth D. The SNoW Learning Architecture // Technical Report UIUCDCS-R-99-2102. UIUC Computer Science Department, 1999.
7. Nilsson M., Dahl M., Claesson I. The successive mean quantization transform / Proceedings of IEEE Int. Conf. ICASSP 2005, Vol. 4 / The Institute of Electrical and Electronics Engineers Signal Processing Society C. 429–432.
8. Rowley H. A., Baluja S., Kanade T. Neural Network-Based Face Detection // PAMI, January 1998.
9. Osuna E., Freund R., Girosi F. Training support vector machines: an application to face detection // In Proceedings of Computer Vision and Pattern Recognition 1997, С. 130-136.
10. Machine learning methods // Graphicon. URL: https://anaconda.cloud/ippolito-machine-learning-visualization (дата звернення: 28.04.2016).
11. Rawlinson T., Bhalerao A., Wang L. Principles and Methods for Face Recognition and Face Modelling // Handbook of research on computational forensics, digital crime and investigation: methods and solutions. IGI Global, С. 53-78.

12. Turk M., Pentland A. Eigenfaces for recognition // Cognitive Neuroscience, 1991, №3(1), С. 71–86.

13. Belhumeur P. N., Hespanha J. P., Kriegman D. J. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection // IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, July 1997, №7, С. 711–720.

14. Ojala T., Pietikäinen M., Harwood D. A Comparative Study of Texture Measures with Classification Based on Feature Distributions // Pattern Recognition, Vol. 29, 1996, С. 51–59.

15. Ahonen T., Hadid A., Pietikäinen M. Face Description with Local Binary Patterns: Application to Face Recognition // IEEE Trans. Pattern Analysis and Machine Intelligence, 1996, №28(12), С. 2037–2041.

16. Edwards G. J., Cootes T. F., Taylor C. J. Face Recognition Using Active Appearance Models // Computer Vision — ECCV’98, Volume 1407 of the series Lecture Notes in Computer Science, С. 581–595.

17. Prabhu U., Seshardi K. Facial Recognition Using Active Shape Models, Local Patches and Support Vector Machines // ECE Department Carnegie Mellon University.

18. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE. – 1998. – Т. 86. – №. 11. – С. 2278-2324.

19. Гонсалес Р., Вудс Р. Цифрова обробка зображень, К: Техносфера, 2005 - 1072 с.

20. Papageorgiou O., Papageorgiou P. A general framework for object detection / International Conference on Computer Vision, 1998 / The Institute of Electrical and Electronics Engineers, Inc. С. 555-562.

21. Face Recognition with Local Binary Patterns. URL: http://uran.donetsk.ua/~masters/2011/frt/dyrul/library/article8.pdf (Дата звернення 15.03.2016).

22. Ojala T., Pietikainen M., Harwood D. A comparative study of texture measures with classification based on feature distribution // Pattern Recognition, Vol. 29, С. 51-59.

23. Ahonen T., Hadid A., Pietikainen M. Face Description with Local Binary Patterns: Application to Face Recognition // IEEE Trans. Pattern Analysis and Machine Intelligence, 1996, №28(12), С. 2037-2041.

24. Le Cun Y., Bengio Y., Hinton G. Deep learning // Nature, 2015, Т. 521, № 7553, С. 436.

25. Face Recognition with OpenCV. URL: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html (Дата звернення 15.03.2016).

ДОДАТОК А

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
"____" _____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Спеціалізоване програмне забезпечення для захоплення і стеження за обличчям користувача у режимі реального часу»
08-54.БДР.028.00.000 ПЗ

Науковий керівник: ст. викл. каф. ОТ
_____ Кисюк Д. В.
Студент групи 2СП-20б
_____ Федоренко М. С.

ВНТУ 2024

1 Підстава для виконання бакалаврської дипломної роботи (БДР)

1.1 Актуальність даного дослідження зумовлена зростаючою потребою у високоякісних системах для захоплення і стеження за обличчям користувача в режимі реального часу. Сучасні технології ідентифікації та верифікації особистості стають все більш важливими у різних сферах, включаючи безпеку, охорону здоров'я, розваги та маркетинг. Зокрема, в системах безпеки обличчяві технології використовуються для контролю доступу, моніторингу територій та виявлення підозрілої поведінки.

1.2 Наказ про затвердження теми бакалаврської кваліфікаційної роботи.

2 Мета і призначення БДР

2.1 Мета даної бакалаврської кваліфікаційної роботи полягає у розробці спеціалізованого програмного забезпечення для захоплення і стеження за обличчям користувача в режимі реального часу. Ця мета передбачає досягнення високої точності і швидкодії системи, а також забезпечення її стабільної роботи в умовах реального використання.

2.2 Призначення розробки — виконання бакалаврської кваліфікаційної роботи із подальшим впровадженням та розвитком продукту.

3 Вихідні дані для виконання БДР

3.1 Запропонувати нові підходи для реалізації методу виділення та відстеження обличчя користувача у режимі реального часу.

3.2 Кольорове цифрове зображення з розширенням не менше за 240 пікселів по горизонталі та 240 пікселів по вертикалі, розміром не більше 2Мб.

3.3 Частота надходження відеокадрів не менше 15 кадрів/сек.

4 Вимоги до виконання БДР

Запропонувати нові підходи для реалізації методу виявлення та відстеження облич користувачів у режимі реального часу. Розробити

послідовність для виявлення та відстеження облич користувачів у отриманому відеопотоці.

Виконати розробку програмного забезпечення для виділення та відстеження облич користувачів, провести його тестування. Схеми послідовності виділення та відстеження облич користувачів та лістинги програми представити в додатках до роботи.

Виконати розробку програмного забезпечення для виділення та відстеження пересування людини, провести його тестування. Схеми послідовності виділення та відстеження пересування людинита лістинги програми представити в додатках до роботи.

5 Етапи БДР та очікувані результати

Етапи БДР та очікувані результати приведені у таблиці А.1.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	25.03.24	30.03.24	Вступ
2	Аналіз способів розпізнавання обличчя на картинках та відеопотоках	01.04.24	10.04.24	розділ 1
3	Розробка послідовності та засобів длявідстеження обличчя у відеопотоці	11.04.24	21.04.24	Розділ 2, розробка способу
4	Розробка програми, проектування засобів розпізнавання облиичя у режимі реального часу	22.04.24	14.05.24	Розділ 3, розробка програми
5	Практична реалізація, результати	15.05.24	30.05.24	Розділ 4 та 5
7	Оформлення пояснювальної записки	01.06.24	12.06.24	ПЗ, презентація

6 Матеріали, що подаються до захисту БДР: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на

кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БДР

Виконання етапів розрахункової та графічної документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БДР

8.1 При оформлювання БДР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Системне програмування»). Кафедра обчислювальної техніки ВНТУ 2024;

— документами, на які посилаються у вище вказаних.

8.2 Порядок виконання БДР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

Лістинг програми

Системи реєстрацій подій.

```
importface_recognition

import glob

import sqlite3

import cv2

import numpy as np

import os

import sys

import datetime

from datetime import timedelta

import time

sampleNum=0
sampleNum=sampleNum+1

def insert_prefix(frame, prefix, size):

  (top, bottom, left, right) = size

  (prefix_height, prefix_width, _) =
  prefix.shape (frame_height,
  frame_width, _) = frame.shape

  _left = ((right + left) - prefix_width)
  // 2 prefix_left = max(_left, 0)

  prefix_top = max(bottom, 0)

  prefix_right = min(prefix_left +
  prefix_width, frame_width)
  prefix_bottom = min(bottom +
```

```
    prefix_height, frame_height)
    prefix_calc_top = prefix_top -
    bottom

    prefix_calc_left = prefix_left - _left

    prefix_calc_bottom =
    prefix_calc_top + prefix_bottom -
    prefix_top prefix_calc_right =
    prefix_calc_left + prefix_right -
    prefix_left

    mask = prefix[prefix_calc_top:
    prefix_calc_bottom,
    prefix_calc_left:prefix_calc_right, 3]
    calc_prefix = prefix[prefix_calc_top:
    prefix_calc_bottom,
    prefix_calc_left:prefix_calc_right,
    :3] mask = np.asarray([mask, mask,
    mask],
    dtype=np.float32).transpose([1, 2,
    0]) / 255.0 prefix_frame =
    frame[prefix_top:prefix_bottom,
    prefix_left: prefix_right]

    frame[prefix_top:prefix_bottom,
    prefix_left: prefix_right] =
    calc_prefix * mask + (prefix_frame
    * (1.0 - mask))

UNKNOWN_NAME = 'Unknown'
KNOWN_NAME = 'Labib'

def main():

    video_capture =
    cv2.VideoCapture(0) count = 0
```

```
count1 = 0

#--- Extract files from folder following pattern

conn=sqlite3.connect('FaceRecogniti on.db') cursor= conn.cursor()

My_Face1_image =[];
My_Face1_face_encoding = [];
known_face_names = [];
known_face_date =[];

date = datetime.datetime.now()

PATH_IMAGES = 'Unknown/'

files  = glob.glob(PATH_IMAGES+"*.jpg")
n_files = len(files)

print('Number of files in folder: ', n_files)

name = 'Face Not Known'

#row = cursor.execute("""SELECT ObjId, Name from FaceImages""")
#for  ObjId, Name in row:

for i, file_i in enumerate(files):

  #--- Read image as a binary blob
  with open(file_i, 'rb') as f:

    image_bytes = f.read() f.close()

  #--- Decode raw bytes to get image size
```

```
    nparr =
    np.fromstring(image_bytes,
    np.uint8)

    img_np = cv2.imdecode(nparr,
    cv2.IMREAD_COLOR)
    image_size = img_np.shape[1]

    #--- Extract file name without
    extension

    filename = os.path.relpath(file_i,
    PATH_IMAGES) objid =
    int(os.path.splitext(filename)[0])

    #--- Insert image and data into
    table

    cursor.execute("insert into FaceImages(ObjId , img ,siz
e , Name, DateTIme) VALUES(?,?,?,?,?)",
(objid,sqlite3.Binary(image_bytes),im
age_size,name,date,) )

    #--- Cheap progress if ((i % 100)
    == 0):

       print(i, n_files) if n_files > 0:

      os.remove("Unknown/1.jpg")

  row = cursor.execute("""SELECT
  ObjId, img from FaceImages""") for
  ObjId, img in row:

   #--- Decode blob

    nparr = np.fromstring(img ,
    np.uint8)
```

```
  img_np = cv2.imdecode(nparr,
  cv2.IMREAD_COLOR) encoding
  =
  face_recognition.face_encodings(i
  mg_np)[0];
  My_Face1_face_encoding.append(
  encoding);

row = cursor.execute("""SELECT
ObjId, Name from FaceImages""")
for  ObjId, Name in row:

  known_face_names.append(Name)

conn.commit() cursor.close()
conn.close()

known_face_encodings =
My_Face1_face_encoding

# Initialize some variables
face_locations = [] face_encodings =
[] face_names = []
process_this_frame = True

while True:

  # Grab a single frame of video ret,
  frame = video_capture.read() count
  += 1

  # Resize frame of video to 1/4 size
  for faster face recognition
  processing small_frame =
  cv2.resize(frame, (0, 0), fx=0.25,
  fy=0.25)
```

```
# Convert the image from BGR
color (which OpenCV uses) to
RGB color (which
face_recognition uses)
rgb_small_frame = small_frame[:,
:, ::-1]

# Only process every other frame
of video to save time if
process_this_frame:

  # Find all the faces and face
  encodings in the current frame of
  video face_locations =
  face_recognition.face_locations(r
  gb_small_frame)

  face_encodings =
  face_recognition.face_encodings
  (rgb_small_frame,
  face_locations)

  face_names = []

  for face_encoding in
  face_encodings:

    # See if the face is a match for
    the known face(s)

    matches =
    face_recognition.compare_face
    s(known_face_encodings,
    face_encoding) name =
    UNKNOWN_NAME
```

```
        # If a match was found in
        known_face_encodings, just
        use the first one. if True in
        matches:

          first_match_index =
          matches.index(True)

          name =
          known_face_names[first_mat
          ch_index] sys.stdout.write("
          Detected faces: "+str(name)+
          " | ")

        face_names.append(name)

        conn=sqlite3.connect('FaceRecog
        nition.db') cursor= conn.cursor()

        Time10
        =datetime.datetime.now().strftim
        e("%Y-%m-%d %H:%M:%S")
        for i in range(len(face_names)):

          row =
cursor.execute("SELECT *
FROM(SELECT id , User_Name ,
DataTime FROM UserTime ORDER
BY DataTime DESC LIMIT 20)
ORDER BY DataTime ASC ")

          for id , User_Name , DataTime
          in row:

           now =
           datetime.datetime.now()
```

```
    mynow = (now -
    timedelta(minutes=1)).strftim
    e("%Y-%m-%d
    %H:%M:%S"); if User_Name
    == face_names[i] and mynow
    < DataTime:

      #sys.stdout.write(" Detected
      faces: "+str(User_Name)+ "
      | ")

      face_names[i]="<"+str(name
      )+">"

      #print ();

for i in range(len(face_names)):

  if
  face_names[i]!="<"+str(name)+
  ">":

      cursor.execute("insert into
      UserTime(User_Name,
      DataTime) VALUES(?,?)",
      (face_names[i],Time10)  )
      cursor.execute("DELETE
      FROM UserTime where
      User_Name like '<%' ")

      cursor.execute("DELETE
      FROM UserTime where
      User_Name like 'Face%'")
      cursor.execute("DELETE
      FROM UserTime where
      User_Name like
      'Unknown%'")
```

```
    conn.commit() cursor.close()
    conn.close()

process_this_frame = not
process_this_frame

# Display the results

for (top, right, bottom, left), name
in zip(face_locations, face_names):

# Scale back up face locations
    since the frame we detected in
    was scaled to 1/4 size top *= 4

    right *= 4

    bottom *= 4

    left *= 4

    crop_img = frame[top:bottom,
    left:right] if name ==
    KNOWN_NAME:

        #insert_prefix2(frame, prefix2,
        (top, bottom, left, right))
        cv2.rectangle(frame, (left, top),
        (right, bottom), (143, 254, 9),
        2)

        cv2.rectangle(frame, (left,
        bottom - 35), (right, bottom),
        (143, 254, 9), -1) font =
        cv2.FONT_HERSHEY_DUPL
        EX

        cv2.putText(frame, name, (left
        + 6, bottom - 6), font, 1.0, (1, 3,
        0), 1)
```

```
elif name ==
UNKNOWN_NAME:
  # Draw a box around the face
  cv2.rectangle(frame, (left, top),
  (right, bottom), (0, 0, 255), 2)
  cv2.imwrite("Unknown/1.jpg",
  crop_img )
  reload(main())

  # Draw a label with a name
  below the face
  cv2.rectangle(frame, (left,
  bottom - 35), (right, bottom),
  (0, 0, 255), -1) font =
  cv2.FONT_HERSHEY_DUPL
  EX
  cv2.putText(frame, name, (left
+ 6, bottom - 6), font, 1.0, (1, 3,
0), 1) else:
  #insert_prefix(frame, prefix,
  (top, bottom, left, right))
  cv2.rectangle(frame, (left, top),
  (right, bottom), (143, 254, 9), 2)
  cv2.rectangle(frame, (left,
  bottom - 35), (right, bottom),
  (143, 254, 9), -1) font =
  cv2.FONT_HERSHEY_DUPL
  EX
```

```
            cv2.putText(frame, name, (left
            + 6, bottom - 6), font, 1.0, (1, 3,
            0), 1)

        # Display the resulting image

        cv2.imshow('Labib [DLib
        face_recognition using SQLite
        database]', frame) print("  Frames =
        "+str(count))

        #if ret == True:

          #sys.stdout.write('Read %d
          frame: ' % count,) #print('Read
          %d frame: ' % count, )

          #count += 1
        #print(first_match_index)

        # Hit 'q' on the keyboard to quit!

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

    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()

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

ДОДАТОК В

Лістинг програми для імпортування SQLITE3

```
import glob

import numpy as np import sqlite3
import cv2

PATH_IMAGES = 'dataSet1/'

#--- Extract files from folder
following pattern files  =
glob.glob(PATH_IMAGES+"*.jpg")
n_files = len(files)

print('Number of files in folder: ',

n_files) name = 'Face Not Known'


#--- Simple database creator def
create_db(filename):

db = sqlite3.connect(filename)
cursor = db.cursor()

cursor.execute("DROP TABLE IF
EXISTS FaceImages")

cursor.execute("CREATE TABLE
FaceImages(ID INTEGER PRIMARY
KEY AUTOINCREMENT,ObjId
INTEGER,

img BLOB, size INT, Name
TEXT)") db.commit()

db.close()

filename_db = 'FaceRecognition.db'
create_db(filename_db)

#--- Open database and loop over
files to insert in database con =
sqlite3.connect(filename_db)

cur = con.cursor()

for i, file_i in enumerate(files):

#--- Read image as a binary blob
with open(file_i, 'rb') as f:

image_bytes = f.read() f.close()
```

```
        #--- Decode raw bytes to get image
size
        nparr  = np.fromstring(image_bytes,
np.uint8)
        img_np = cv2.imdecode(nparr,
cv2.IMREAD_COLOR) image_size =
img_np.shape[1]
        #--- Extract file name without
extension
        filename = os.path.relpath(file_i,
PATH_IMAGES) objid =
int(os.path.splitext(filename)[0])
        #--- Insert image and data into table
        cur.execute("insertinto
                        FaceImages(ObjI
d       ,       img       ,size       ,       Name)       VALUES(?,?
,?,?)",
(objid,sqlite3.Binary(image_bytes),image_
size,name)  )
        #--- Cheap progress if ((i % 100) ==
                0):
                print(i, n_files)
        con.commit() cur.close() con.close()
        #con =
sqlite3.connect('FaceReg.db') #cur =
con.cursor()
        #row = cur.execute("SELECT
ObjId, img from FaceImages")
        #for ObjId, item in row:
        #--- Decode blob
        #nparr  = np.fromstring(item,
np.uint8)
        #img_np = cv2.imdecode(nparr,
cv2.IMREAD_COLOR)
        #--- Display image
#cv2.imshow('image',img_np) #k =
cv2.waitKey(0)
        #if k == 27:   # wait for ESC key to
exit #cv2.destroyAllWindows()
        #break
```

#cur.close() #con.close()

ДОДАТОК Г

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Спеціалізоване програмне забезпечення для захоплення і стеження за обличчям користувача у режимі реального часу

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 87,7% Схожість 12,3%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Федоренко М. С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кисюк Д. В.
(підпис) (прізвище, ініціали)