КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(назва випускової кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР
на тему:

«Інформаційна технологія візуального сприйняття для виявлення та розпізнавання облич»

ПОЗДНЯКОВИЧ СЕРГІЙ ОЛЕКСАНДРОВИЧ

(прізвище, ім’я та по батькові здобувача повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(назва кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

_____
„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

«Інформаційна технологія візуального сприйняття для виявлення та розпізнавання облич»

(назва)

Виконав
Позднякович Сергій Олександрович
(прізвище, ім’я та по батькові повністю)

122 «Комп`ютерні науки
(спеціальність)

«Інформаційні управляючі системи та технології»
(освітня програма)

Групи КН-20-1

Керівник Гончаренко Т. А.
(прізвище та ініціали)
доцент, кандидат технічних наук
(вчене звання, науковий ступінь)

Ідентичність підтверджую

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Випускова кафедра: інформаційних технологій

Освітній ступінь: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., доцент Гончаренко Т.А.

_____
„___” _____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Позднякович Сергій Олександрович

Тема роботи: Інформаційна технологія візуального сприйняття для виявлення та розпізнавання облич
затверджена наказом ректора КНУБА № 433/2 від 29.02.2024 р.
2. Керівник роботи: Гончаренко Тетяна Андріївна, к.т.н, доцент кафедри інформаційних технологій
3. Строк подання Здобувачем роботи до захисту:_____
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі

Р.2. Проектні рішення

Р.3. Розробка системи виявлення та розпізнавання облич

Р.4. Ергономіка системи виявлення та розпізнавання облич

5. Графічний матеріал за розділами:

Р.1. Аналіз предметної області та постановка задачі

С.1. – С.3. Актуальність теми, мета, об’єкт, предмет дослідження.

С.4, С.5. Огляд існуючих методів розпізнавання облич.

Р.2. Проектні рішення

С.6, С.7. Автоматичне виявлення обличчя людини.

С.8. Автоматичне виявлення частин обличчя.

С.9. Визначення характеристик обличчя.

С.10. Аналіз виразів обличчя.

С.11. Розпізнавання облич.

Р.3. Розробка системи виявлення та розпізнавання облич

С.12. – С.15. Система виявлення та розпізнавання облич. Основні результати.

Р.4. Ергономіка системи виявлення та розпізнавання облич

С.16. – С.18. Основні ергономічні показники.

С.19. Висновки.

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Березень 2024 р.
Р. 2. Проектні рішення	Березень 2024 р.
Р. 3. Розробка системи виявлення та розпізнавання облич	Квітень 2024 р.
Р. 4. Ергономіка системи виявлення та розпізнавання облич	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи для перевірки на плагіат	Червень 2024 р.
Попередній захист роботи на випусковій кафедрі	Червень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис

8. Дата видачі завдання: 29.02.2024 р.

Зав. кафедри _____ Гончаренко Т. А.
(підпис) (прізвище та ініціали)
Керівник _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)
Здобувач _____ Позднякович С.О.
(підпис) (прізвище та ініціали)

<table>
<tr><td>РЕЗЮМЕ (SUMMARY) до атестаційної випускної роботи Здобувача:</td><td colspan="3">Позднякович Сергій Одександрович

Serhii Pozdniakovych</td></tr>
<tr><td>ЗВО</td><td colspan="3">Київський національний університет будівництва і архітектури</td></tr>
<tr><td>Тема (українською та англійською)</td><td colspan="3">Інформаційна технологія візуального сприйняття для виявлення та розпізнавання облич
Information technology of visual perception for face detection and recognition</td></tr>
<tr><td>Освітній ступінь</td><td colspan="3">Бакалавр</td></tr>
<tr><td>Факультет</td><td colspan="3">Автоматизації і інформаційних технологій</td></tr>
<tr><td>Випускаюча кафедра</td><td colspan="3">Інформаційних технологій</td></tr>
<tr><td>Спеціальність</td><td colspan="3">122 «Комп’ютерні науки»</td></tr>
<tr><td>Освітня програма</td><td colspan="3">Інформаційні управляючі системи та технології</td></tr>
<tr><td>Керівник</td><td colspan="3">Гончаренко Тетяна Андріївна</td></tr>
<tr><td rowspan="2">Обсяг роботи:</td><td>пояснювальна записка, стор.</td><td>розділів</td><td>креслень формату А</td></tr>
<tr><td>102</td><td>4</td><td></td></tr>
<tr><td>Ключові слова:

Keywords:</td><td colspan="3">комп’ютерний зір, виявлення і розпізнавання облич, нейронна мережа, OpenCV, Python

computer vision, face detection and recognition, neural network, OpenCV, Python</td></tr>
</table>

У роботі проаналізовано існуючи методи та алгоритми для розпізнавання облич та досліджено задачі: автоматичного виявлення обличчя, автоматичного виявлення частин обличчя, визначення характеристик людини за зображенням її обличчя, аналізу виразів обличчя, розпізнавання облич. З метою розробки інформаційної системи ідентифікації людини створено програмний продукт для автоматизації процесу виявлення та розпізнавання облич.

Здобувач: _____ / Сергій ПОЗДНЯКОВИЧ /

Керівник: _____/ Тетяна ГОНЧАРЕНКО /

“___” _____ 202_р.

ЗМІСТ

ВСТУП

Виявлення і розпізнавання облич являють собою безконтактні методи розпізнавання людей.

Основна причина інтересу до розробки технологій автоматичного розпізнавання облич на основі комп'ютерного зору випливає з серйозної стурбованості громадською безпекою у сучасному світі, де перевірка особи для фізичного і логічного доступу до багатьох об'єктів є обов'язковою у повсякденному житті.

Справжній поштовх у актуалізації технологій візуального сприйняття для виявлення та розпізнавання облич прийшов разом з розвитком обчислювальних потужностей та алгоритмів, пов'язаних з використанням великих баз даних. Тому методи локалізації та ідентифікації облич активно досліджуються.

Метою досліджень в роботі є створення автоматизованої системи виявлення та розпізнавання облич.

Реалізація мети здійснюється вирішенням наступних основних завдань:

- автоматичне виявлення обличчя (Face Detection);
- автоматичне виявлення частин обличчя (Automatic Facial Features Detection);
- визначення характеристик людини за зображенням її обличчя (Objectís Features Determination);
- аналіз виразів обличчя (Facial Expression Analysis);
- розпізнавання облич (Face Recognition).

Об’єктом дослідження роботи є процес виявлення та розпізнавання облич.

Предметом дослідження є методи виявлення та розпізнавання облич які реалізовани у бібліотеках та фреймворках комп’ютерного зору: OpenCV, PIL, DeepFace та face_recognition мови програмування Python.

Система є інструментом візуального розпізнавання.

Зовнішнім середовищем є цифрове зображення облич.

Розробка системи буде відбуватись за допомогою мови програмування Python у середовищі Visual Studio Code.

Перелік умовних позначень і скорочень

ЗНМ – згорткова нейронна мережа

ЛБШ – локальні бінарні шаблони

НМ – нейронна мережа

AAM – active appearance models

PCA – principal component analysis

OpenCV – Open Source Computer Vision Library

PIL – Python Imaging Library

NumPy – Numerical Python

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми дослідження

Питання формалізації та автоматизації процесу розпізнавання людських облич було розглянуто ще на ранній стадії розвитку систем розпізнавання образів і залишається актуальним і сьогодні. Той факт, що кількість досліджень і публікацій на цю тему зросла в кілька разів за останнє десятиліття, вказує на зростаючу актуальність цієї проблеми. Це пов'язано з тим, що сучасні обчислювальні технології відкривають нові можливості для застосування в найрізноманітніших сферах, в результаті чого виникає ряд прикладних задач, які терміново потребують вирішення.

Одним із практичних застосувань теорії розпізнавання образів є розпізнавання облич, до завдання якого входить автоматична локалізація обличчя на зображенні та ідентифікація людини за обличчям. Інтерес до процедур, що лежать в основі процесу локалізації та розпізнавання облич, досить значний у зв'язку з розмаїттям їхнього практичного застосування в таких галузях, як охоронні системи, верифікація, криміналістична експертиза тощо.

В результаті досліджень алгоритмів розпізнавання образів були розроблені різні бібліотеки. Найвідомішою з них є бібліотека OpenCV, яка комбінує в собі більшість з них. Бібліотека реалізована на C/C++, але також може бути використана в інших мовах програмування, таких як Python, Ruby, Matlab, Lua. Особливу цінність в OpenCV має математичний апарат і можливості обробки зображень.. Бібліотека може вільно використовуватися в академічних та комерційних цілях. Вона розповсюджується за умовами ліцензії BSD.

Готові рішення для розпізнавання вже розроблені, але вони мають певні недоліки. Технологія DeepFace від Facebook може розпізнавати 97,25% облич. Ця технологія використовується в межах Facebook, однак доступу до її архітектури у сторонніх розробників немає, а в публічному доступі є лише документація.

FaceAPI від Google можна використовувати для розпізнавання облич на зображеннях. Ця технологія показує хороші результати, з показником розпізнавання

понад 90 відсотків. Однак доступ до бази даних облич обмежений, і немає можливості додавати власні дані. Тому удосконалення технологій візуального сприйняття для виявлення та розпізнавання облич є актуальною науково технічною задачею.

1.2 Огляд існуючих методів розпізнавання облич

У сфері машинного зору та систем штучного інтелекту виявлення облич і подальша ідентифікація осіб завжди були одним з головних пріоритетів для дослідників.

Впровадження системи виявлення та ідентифікації людини на основі зображень вимагає обробки великих обсягів вхідних даних та виокремлення суттєвих ознак для отримання задовільних результатів.

Процес розпізнавання облич можна розділити на чотири етапи (Рис. 1.1):

1. Виявлення обличчя (Face detection).
2. Вирівнювання обличчя (Face alignment).
3. Виділення ознак обличчя (Face feature extraction).
4. Порівняння обличчя за ознаками (Face feature matching).

На першому етапі проводиться виявлення і локалізація обличчя на зображенні. При розпізнавання виконується вирівнювання зображення обличчя (геометрично та за рівнем яскравості), виокремлення та обчислення характерних ознак. На останньому етапі проводиться безпосередньо розпізнавання - зіставлення обличчя з базою даних відомих облич відповідно до значень ознак і повернення найкращого збігу.

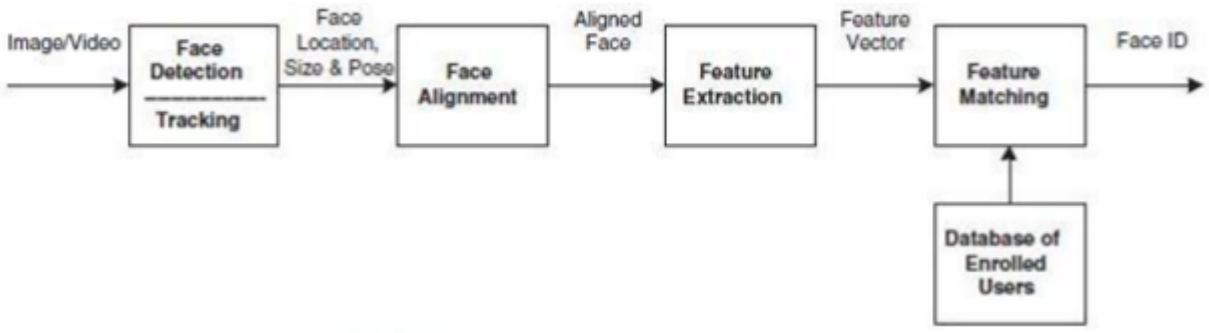


Рис. 1.1. Загальний процес обробки зображення при розпізнаванні обличчя

1.2.1 Аналіз геометричних характеристик обличчя

Даний метод є одним із перших серед використовуваних методів розпізнавання облич [1]. Суть його полягає у виділенні помітних орієнтирів, які також називаються вузловими точками, з кожного обличчя і наступному формуванні набору ознак. До вузлових точок можуть належати куточки очей, губ, кінчик носа, центр ока. Інші характеристики включають відстань між очима або форму вилиць (Рис. 1.2).

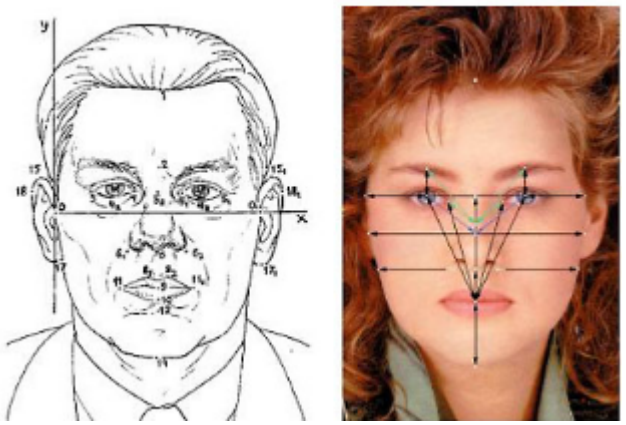


Рис. 1.2. Вузлові точки і відстані

Процес розпізнавання порівнює обличчя невідомої людини з тими, що зберігаються в базі даних, відповідно до значень вузлових точок.

До переваг методу можна віднести використання недорогого обладнання; при відповідному обладнанні є можливість розпізнавання зображень отриманих на значних відстанях.

Серед недоліків можна виділити такі: високі вимоги до освітлення і необхідність фронтального зображення обличчя з мінімальними відхиленнями.

1.2.2 Метод гнучкого порівняння на графах

Даний метод полягає в порівнянні графів, що описують зображення обличчя. Обличчя виглядають як графи зі зваженими вершинами та ребрами. У процесі розпізнавання еталонний граф не змінюється, а інший деформується з метою найкращого наближення до еталонного.

У таких системах розпізнавання графи можуть мати форму прямокутної решітки або являти собою структуру, утворену характерними (антропометричними) точками обличчя (Рис. 1.3).

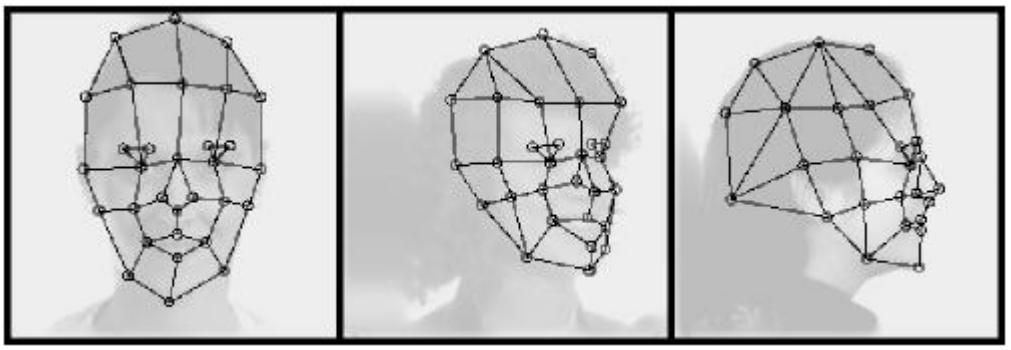


Рис. 1.3. Граф на основі антропометричних точок обличчя

У вершинах графа обчислюються значення ознак, найчастіше використовують комплексні значення фільтрів Габора або їх впорядкованих наборів, які обчислюються в деякій локальній області вершини графа шляхом згортки значень яскравості пікселів з фільтрами Габора [2].

Ребра графа мають ваги, що відповідають відстаням між сусідніми вершинами. Відмінність між двома графами оцінюють за допомогою цінової функції деформації, яка враховує як відмінність значень ознак у вершинах, так і ступінь деформації ребер графа.

Деформація графа зміщує кожну вершину на деяку відстань у певному напрямку відносно вихідного положення, при цьому відбувається вибір такої її позиції, за якої різниця між значеннями ознак (відгуків фільтрів Ґабора) у відповідних вершинах деформованого графа та еталонного графа буде мінімальною. Цей процес виконується поперемінно по всіх вершинах графа до тих пір, поки не буде досягнута мінімальна сумарна відмінність між ознаками деформованого і еталонного графів. Значення цінової функції деформації при такому положенні деформованого графа і буде мірою відмінності між вхідним зображенням обличчя і еталонним графом. Дана процедура деформації повинна виконуватися для всіх еталонних осіб, що додані до бази даних системи. Еталон з

найкращим значенням цінової функції деформації є результатом розпізнавання системи.

Алгоритм методу гнучкого порівняння на графах наведено на Рис. 1.4.

Рис. 1.4 Алгоритм методу гнучкого порівняння на графах

Основною перевагою цього методу є низька чутливість до рівня освітленості обличчя та до зміни кута обличчя.

Недоліки: висока обчислювальна складність процедури розпізнавання. Низька технологічність при запам'ятовуванні нових еталонів. Залежність часу роботи від розміру бази даних осіб - лінійна.

1.2.3 Метод головних компонент

Одним із найвідоміших і опрацьованих є метод аналізу головних компонент (principal component analysis, PCA).

Спочатку цей метод використовувався в статистиці для зменшення простору ознак без суттєвої втрати інформації. У задачі розпізнавання обличчя цей метод застосовують головним чином для представлення зображення особи вектором малої розмірності (головних компонент), який порівнюється потім з еталонними векторами, закладеними в базу даних.

Основна мета методу аналізу головних компонент – значно зменшити розмірність простору ознак, щоб максимально точно описати "типові" образи, що належать безлічі осіб. Використовуючи цей метод можна виявити різні мінливості в навчальній вибірці зображень обличь і описати цю мінливість в базисі декількох ортогональних векторів, які називаються власними (eigenface).

Отриманий один раз на навчальній вибірці зображень обличь набір власних векторів використовується для кодування всіх інших зображень осіб, які представляються зваженої комбінацією цих власних векторів. Використовуючи обмежену кількість власних векторів можна отримати стислу апроксимацію вхідному зображенню особи, яку потім можна зберігати в базі даних у вигляді вектора коефіцієнтів, який служить одночасно ключем пошуку в базі даних осіб.

Суть методу аналізу головних компонент зводиться до наступного (Рис. 1.5).

Весь навчальний набір осіб спочатку перетворюється в одну загальну матрицю даних. Кожен рядок цієї матриці являє собою один екземпляр зображення особи. Усі особи приводяться до одного розміру і з нормованими гістограмами.

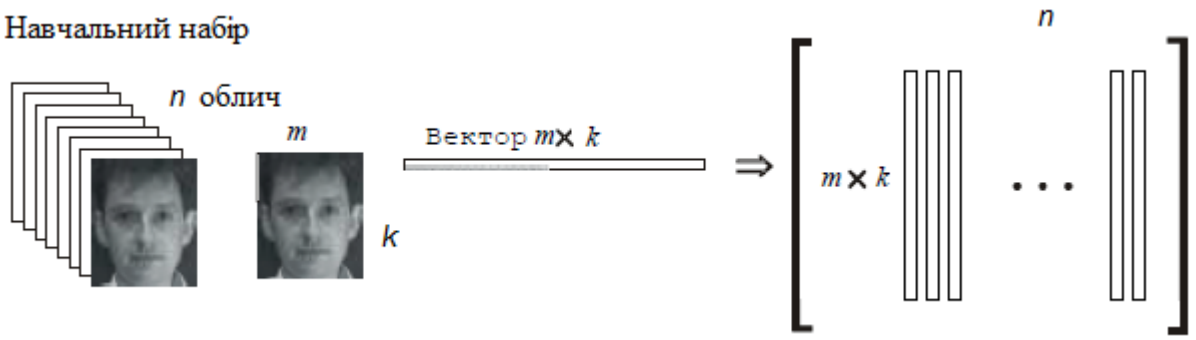


Рис 1.5. Перетворення навчального набору осіб в одну загальну матрицю

Потім проводиться нормування даних і приведення рядків до 0-го середнього і 1-й дисперсії, обчислюється матриця коваріації. Для отриманої матриці коваріації вирішується завдання визначення власних значень і відповідних їм власних векторів (власні особи). Далі проводиться сортування власних векторів в порядку убування власних значень і залишають тільки перші k векторів за правилом:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} > 0,95 .$$

Метод аналізу головних компонент добре зарекомендував себе в практичному застосуванні. Однак ефективність цього методу значно знижується, якщо на зображенні присутні значні зміни в освітленості або виразі обличчя. Вся справа в тому, що PCA вибирає підпростір з такою метою, щоб максимально апроксимувати вхідний набір даних, а не виконати дискримінацію між класами осіб.

В роботі [3] було запропоновано розв'язання цієї проблеми, шляхом використання лінійного дискримінанта Фішера (також відомого як «Eigen-Fisher», «Fisherface», LDA). LDA вибирає лінійний підпростір, який максимізує відношення:

$$\frac{\left|\Phi^T S_b \Phi\right|}{\left|\Phi^T S_w \Phi\right|},$$

де

$$S_b = \sum_{i=1}^{m} N_i \left(\overline{x}_i - \overline{x}\right)\left(\overline{x}_i - \overline{x}\right)^T$$

матриця міжкласового розкиду, і

$$S_w = \sum_{i=1}^{m} \sum_{x \in X_i} \left(\overline{x}_i - \overline{x}\right)\left(\overline{x}_i - \overline{x}\right)^T$$

матриця внутрішньокласового розкиду; m - кількість класів в базі даних.

LDA відшукує проекцію даних, при якій класи є максимально лінійно роздільні (рис. 1.6). PCA шукає проекцію даних, яка забезпечує максимальний розкид по всій базі даних осіб (без урахування класів). За результатами експериментів [3] в умовах сильного бакового і нижнього затінення зображень осіб Fisherface показав 95 % ефективність у порівнянні з 53 % Eigenface.

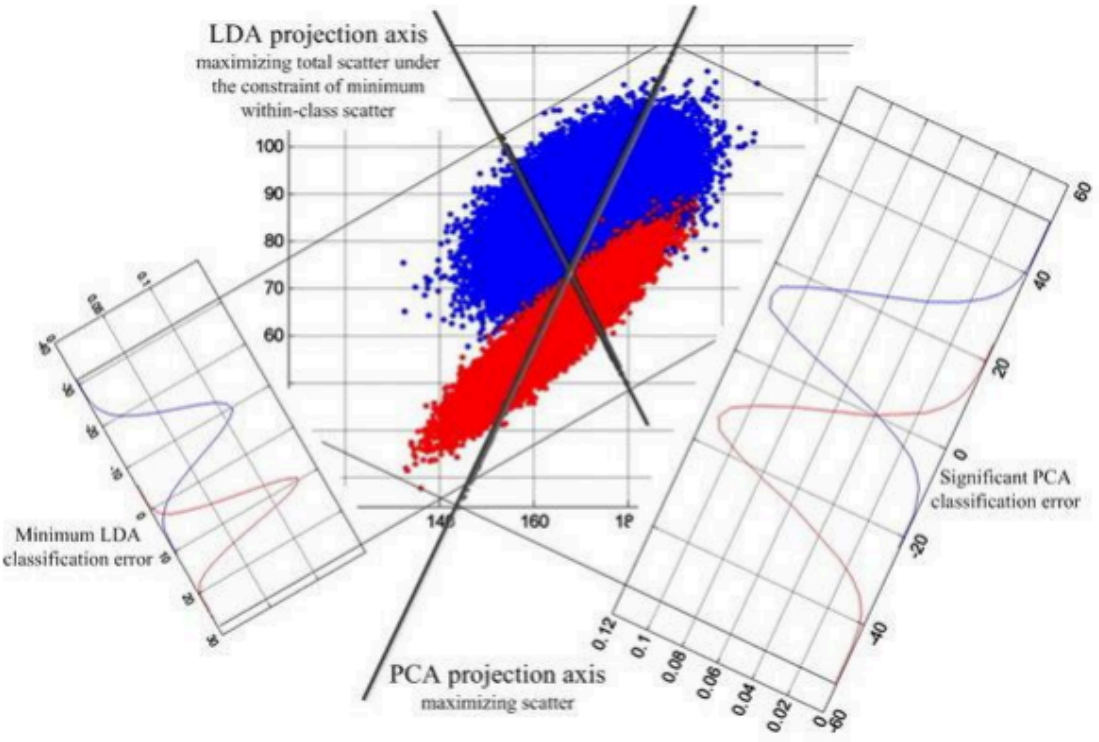


Рис. 1.6. Принципова різниця між формуванням проекції PCA і LDA

Метод головних компонент (PCA) і метод лінійного дискримінанта Фішера (LDA) різняться у своїх підходах до зменшення розмірності даних:

- PCA прагне зберегти якомога більше дисперсії даних, знижуючи розмірність простору з високою розмірністю.
- LDA цілеспрямовано зберігає дискримінаційну інформацію між класами під час зменшення розмірності, що допомагає поліпшити роздільність класів.

1.2.4 Активні моделі зовнішнього вигляду

Активні моделі зовнішнього вигляду (Active Appearance Models, AAM) - є статистичними моделями зображень, які можуть бути адаптовані під реальне зображення за допомогою різного роду деформацій.

В моделях цього виду описуються два типи параметрів: параметри, пов'язані з формою (параметри форми), і параметри, пов'язані зі статистичною моделлю зображення або текстурою (параметри зовнішнього вигляду).

Перед використанням модель необхідно навчити на великій кількості заздалегідь розмічених зображень. (Рис. 1.7). Кожна мітка має свій власний номер, який визначає відповідну точку, яку модель повинна знайти під час адаптації до нового зображення [4].

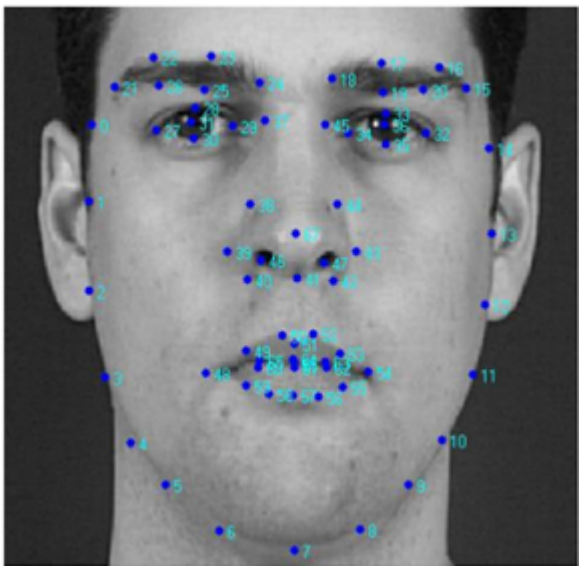


Рис. 1.7. Приклад розмітки зображення обличчя

1.2.5 Метод Віоли-Джонса

Метод Віоли-Джонса дозволяє виявляти об'єкти на зображеннях в реальному часі. В якості ознак для алгоритму розпізнавання використовуються ознаки, на основі вейвлетів Хаара [5]. Білому пікселю в такому вейвлеті відповідає вага +1, а чорному - вага -1. Вейвлети застосовуються для встановлення "приблизної схожості яскравості" ділянки зображення з таким патерном. "Чорний" відповідає темнішим, а "білий" - світлішим ділянкам на обличчях людей. (Рис. 1.8).

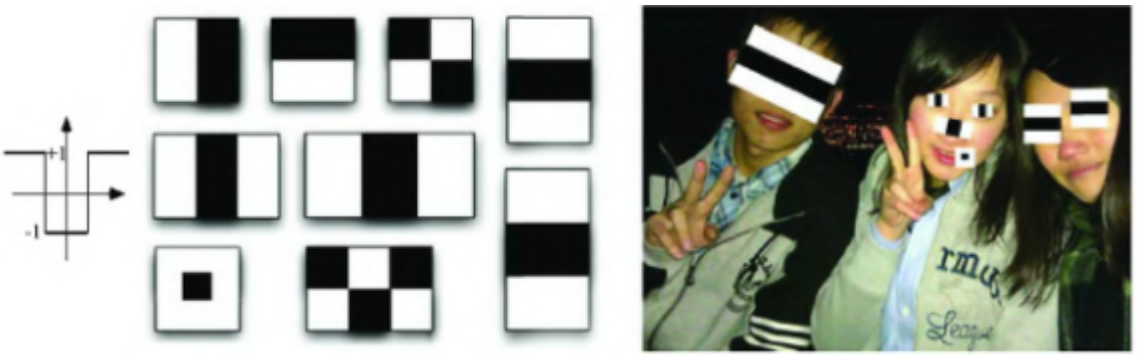


Рис. 1.8. Вейвлети Хаара

Основною перевагою способу є те, що процес виявлення складається в основному з додавань і порогових порівнянь. Виявлення є швидким, навіть в системах з обмеженими ресурсами, такі як мобільні пристрої. З іншого боку, точність детектора особи сильно залежить від використовуваної бази даних для навчання.

Цей метод містить у собі два основні етапи: навчання і розпізнавання.

Метод має такі переваги – можливе виявлення більше однієї особи на зображенні, використання простих класифікаторів, показує високу швидкість, що дозволяє використовувати цей метод для аналізу відеопотоку.

До недоліків відноситься складність навчання, оскільки алгоритму необхідно проаналізувати велику кількість тестових зображень. Для подолання цього в [6] запропоновано використання інтегрального зображення. Завдяки цьому, у вбудованій системі був успішно реалізований детектор осіб в режимі реального часу. Цей метод добре працює при спостереженні за об'єктом під малим кутом, приблизно до 30°, та зазвичай демонструє точність розпізнавання на рівні понад

90%, що вважається досить високим показником. Але якщо кут відхилення перевищує 30° ймовірність розпізнавання різко падає. Враховуючи цю особливість неможна розпізнати особу під довільним кутом [7,8].

Слід відмітити велику кількість реалізацій даного методу, в тому числі у складі бібліотеки комп'ютерного зору OpenCV.

1.2.6 Нейронні мережі

Існує близько десятка різновидів нейронних мереж (НМ). Найпоширенішою є мережа, заснована на багатошаровому персептроні, яка здатна класифікувати зображення або сигнали на вході відповідно до попередніх налаштувань або навчання мережі.

Навчаються нейронні мережі на наборі навчальних прикладів. Основна концепція навчання полягає в налаштуванні ваг міжнейронних зв'язків у процесі розв'язання оптимізаційної задачі з використанням методу градієнтного спуску. У процесі навчання НМ автоматично вилучаються ключові ознаки, визначаються їх важливості та будуються взаємозв'язків між ними. Передбачається, що навчена НМ зможе застосовувати досвід, отриманий в процесі навчання, до невідомого зображення завдяки здатності узагальнювати.

Одні з найкращих результатів в області розпізнавання осіб досягається за допомогою використання згорткових нейронних мереж (ЗНМ), які є логічним розвитком таких архітектур як когнітрон і неокогнітрон. Успіх зумовлений можливістю врахування двовимірної топології зображення, що відрізняє його від багатошарового перцептрона. (Рис. 1.9).

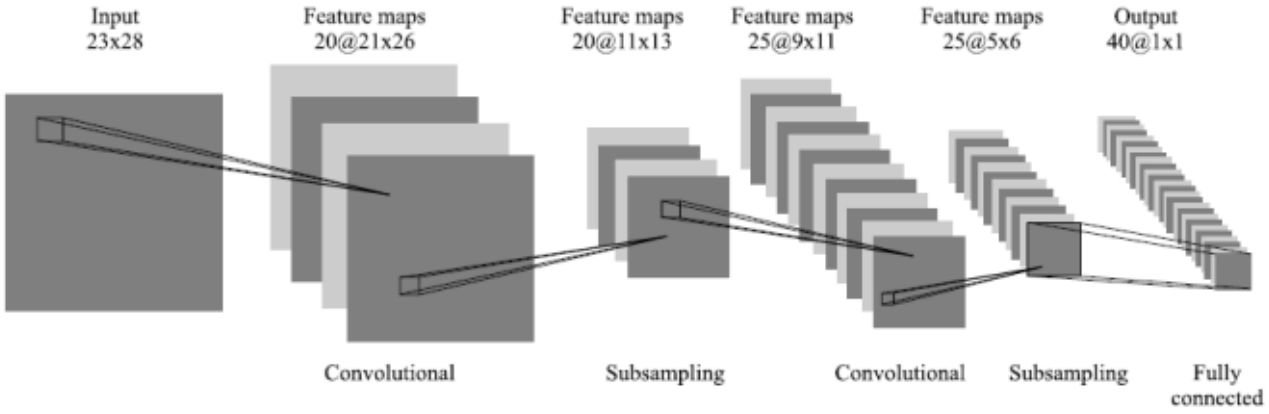


Рис. 1.9. Схематичне зображення архітектури згорткової нейронної мережі

Відмінними рисами ЗНМ є локальні рецепторні поля, які забезпечують локальну двовимірну зв'язність нейронів, загальні ваги, що дозволяють детектування окремих рис в будь-якому місці зображення та ієрархічна організація з просторовими семплінгом. Завдяки цим нововведенням ЗНМ забезпечує часткову стійкість до змін масштабу, зсувам, поворотам, зміні ракурсу та іншим спотворень [9-12].

Експерименти з тестування методу на базі даних ORL [13], що включає зображення облич із різними змінами освітлення, масштабу, просторового повороту, положення і вираження емоцій, показали точність розпізнавання на рівні 96%.

Свій розвиток ЗНМ отримали у розробці DeepFace [10], яка використовується для розпізнавання користувачів соцмереж. Слід відмітити, що особливості архітектури даної нейронної мережі носять закритий характер.

До недоліків, методів заснованих на нейронних мережах, відноситься додавання нових еталонних осіб в базу даних. Для цього потрібне повне перенавчання мережі на всьому наявному наборі. Це досить тривалий процес, який залежить від розміру вибірки і вимагає багато часу. Також виникають проблеми математичного характеру, пов'язані з навчанням: попадання в локальні екстремуми, вибір оптимальних кроків оптимізації, перенавчання, тощо. Крім того, складнощі виникають у процесі формалізації етапу вибору архітектури мережі (кількість нейронів, шарів, характер з'єднання).

1.2.7 Локальні бінарні шаблони

Локальні бінарні шаблони (ЛБШ) були розроблені для аналізу текстури напівтонових зображень. ЛБШ не залежать від невеликих змін в умовах освітлення та невеликих поворотів зображення [14,15]. Це опис околу пікселя зображення в двійковому представленні.

Початкове зображення розбивається на блоки 3x3 пікселя. Оператор ЛБШ використовує вісім пікселів околу, приймаючи значення інтенсивності центрального пікселя в якості порогу. Пікселі зі значенням інтенсивності більшим або рівним значенню інтенсивності центрального пікселя приймають значення рівні «1», інші приймають значення рівні «0». Таким чином, результатом застосування базового оператора ЛБШ до пікселя зображення є восьмирозрядний бінарний код, який описує окіл цього пікселя [15], наприклад, 11100001 (Рис. 1.10).

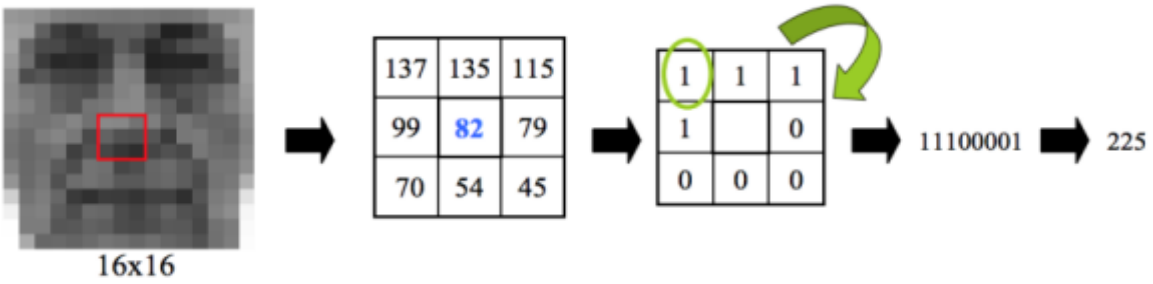


Рис. 1.10. Отримання локального бінарного шаблону пікселя

ЛБШ обчислюється для кожного блоку зображення, після цього обчислюються гістограми кожного блоку і конкатенуються в загальну гістограму особливостей зображення людського обличчя (Рис. 1.11).

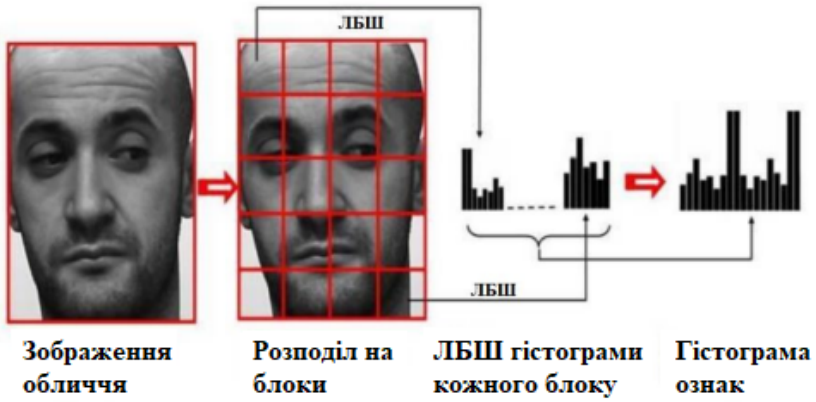


Рис. 1.11. Розбиття зображення і формування гістограми

Розпізнавання відбувається завдяки порівнянню гістограм облич із тренувальної бази даних і гістограми особи, що розпізнається. Точність розпізнавання [13] становить 95%.

Методи на основі ЛБШ добре працюють при використанні зображень облич із різною мімікою, різним освітленням, поворотами голови.

Серед недоліків цього методу можна відзначити необхідність проведення якісної попередньої обробки зображень через їхню високу чутливість до шуму. За наявності шуму кількість хибних бінарних кодів може значно зрости.

1.3 Аспекти, що впливають на якість розпізнавання облич

На якість розпізнавання обличчя впливають різні фактори:

- **Розмір та орієнтації обличчя на фотографії.** Оскільки фотографії можуть бути непрофесійними, то розміри обличчя та його орієнтація на знімку можуть досить сильно відрізнятися від одного зображення до іншого. Методи обробки по-різному чутливі до незначних і істотних змін розмірів обличчя та його орієнтації, однак якщо обличчя занадто мале або сильно повернене убік, то людину досить складно, а іноді й неможливо розпізнати автоматично, навіть після масштабування і внесення відповідних повороту коригувань у роботу алгоритму. Тому потрібно вводити певні вимоги до даних, що надходять.
- **Освітлення.** Яскравість і чіткість зображення дуже сильно залежать від умов освітлення в момент зйомки. Погана якість фотографії може призвести до збоїв в алгоритмах бінаризації та групування, і, отже, загальний коефіцієнт розпізнавання системи також значно знизиться. Необхідно передбачати додаткові алгоритми фільтрації для зменшення можливого негативного ефекту.
- **Відкритість обличчя на зображенні.** Іноді ділянки обличчя можуть закриватися іншими предметами, такими, як капелюхи, окуляри або волосся. Більшість систем розпізнавання не можуть успішно впоратися з цією проблемою. Зазвичай алгоритми розпізнавання базуються на певних ділянках обличчя,

найчастіше на очах. Тому вимагається, щоб вхідні зображення мали необхідний фрагмент для повного аналізу.

- **Емоційний вираз.** Для деяких алгоритмів вираз обличчя не відіграє особливої ролі, проте навіть у людини періодично виникають складнощі з впізнаванням знайомих їй облич у моменти сильного емоційного напруження. Тому доцільно вводити також деякі обмеження на вхідні зображення або доповнювати систему алгоритмом визначення виду емоцій і отримання «нейтрального» виразу обличчя.

2. ПРОЕКТНІ РІШЕННЯ

2.1 Завдання, пов'язані з аналізом зображень обличчя людини

Існує низка завдань, пов'язаних з аналізом зображень обличчя людини, до яких можна застосувати методи комп'ютерного зору та машинного навчання. Ці завдання варіюються від виявлення та розпізнавання облич до аналізу емоцій (Рис. 2.1).

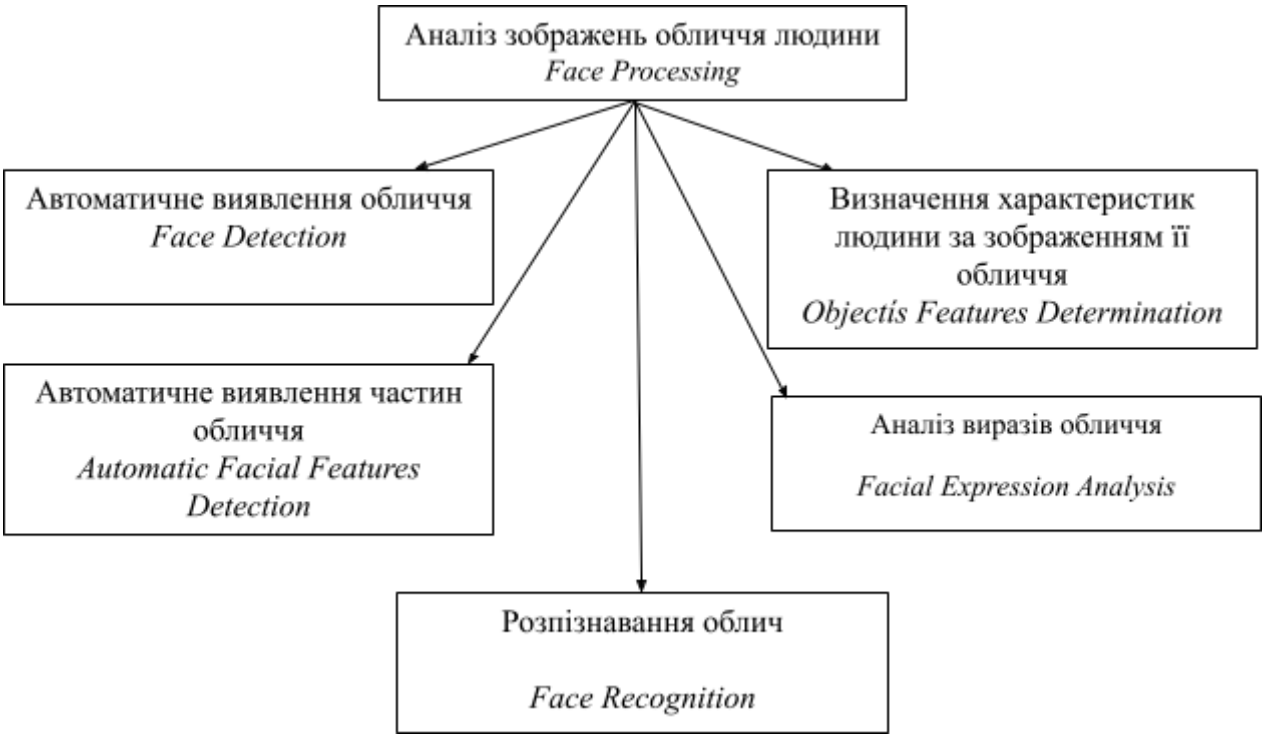


Рис. 2.1. Класифікація завдань, пов'язаних з аналізом зображень обличчя людини

2.2 Автоматичне виявлення обличчя людини (Face Detection)

Найпопулярнішим методом виявлення облич у реальному часі є використання каскадного класифікатора Хаара - ефективного методу виявлення об'єктів, запропонованого Полом Віолою та Майклом Джонсом.

Каскад Хаара - спосіб виявлення об'єктів на зображенні, заснований на машинному навчанні. Навчений каскад Хаара, приймаючи на вхід зображення, визначає, чи є на ньому шуканий об'єкт, тобто виконує завдання класифікації, розділяючи вхідні дані на два класи: є шуканий об'єкт, немає шуканого об'єкта. Власне, сам алгоритм роботи зводиться до того, що визначаються ознаки Хаара.
Це, по суті, набір прямокутних областей, які проходять по зображенню і знаходять у ньому деталі шуканого об'єкта.

До бібліотеки комп'ютерного зору OpenCV включено набір вже навчених класифікаторів у вигляді XML-файлів (табл. 2.1).

Таблиця 2.1. Набір навчених класифікаторів Хаара

№	Назва файлу	Розпізнані об'єкти
1	haarcascade_eye.xml	Очі
2	haarcascade_eye_tree_eyeglasses.xml	Очі в окулярах
3	haarcascade_frontalcatface.xml	Фронтальна котяча морда
4	haarcascade_frontalcatface_extended.xml	Фронтальна котяча морда
5	haarcascade_frontalface_alt.xml	Фронтальне обличчя
6	haarcascade_frontalface_alt2.xml	Фронтальне обличчя 2
7	haarcascade_frontalface_alt_tree.xml	Фронтальне обличчя каскад
8	haarcascade_frontalface_default.xml	Фронтальне обличчя
9	haarcascade_fullbody.xml	Фронтальне тіло
10	haarcascade_lefteye_2splits.xml	Ліве око
11	haarcascadejowerbody.xml	Нижня частина тіла
12	haarcascade_profileface.xml	Профіль обличчя
13	haarcascade_righteye_2splits.xml	Праве око
14	haarcascade_smile.xml	Посмішка
15	haarcascade_upperbody.xml	Верхня частина тіла

У мові програмування Python їх можна підключати з використанням наступної інструкції:

classifier = cv2.CascadeClassifier(

cv2.data.haarcascades + “haarcascade_frontalface_default.xml”),

де “haarcascade_frontalface_default.xml” - ім'я відповідного каскаду Хаара.

Програму виявлення облич наведено на Рис. 2.2.

```
import cv2
img = cv2.imread(r'C:/Foto/face1.jpg')
cv2.imshow('Input photo', img)
classifier = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
gray =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
boxes= classifier.detectMultiScale(gray,  scaleFactor=1.1, minNeighbors=5,  minSize=(30, 30))

for (x,y,w,h) in boxes:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

cv2.imshow('Face detection', img)
cv2.imwrite(r'C:/Foto/face1_detection.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.2. Програма виявлення облич за допомогою бібліотеки OpenCV

У першому рядку підключено бібліотеку OpenCV. Далі у змінну img завантажено тестове зображення face1.jpg і це зображення виведено користувачеві, а у змінну classifier - навчений класифікатор виявлення облич. Далі завантажене зображення конвертовано з кольорового в чорно-біле зображення (у градаціях сірого), і цей результат записано у змінну gray. Після цього ініційовано процес виявлення облич, і його результат записано у змінну boxes. На наступному кроці запущено процес формування прямокутника навколо кожного виявленого обличчя. Потім оброблене зображення показано у вікні та збережено у файлі face1_detection.jpg. Останні два рядки програми закривають усі вікна.

Для перевірки роботи програми було взято зображення, представлене на Рис. 2.3.

У результаті роботи програми отримано такий результат (Рис. 2.4). Як можна бачити, метод на основі каскадів Хаара haarcascade_frontalface_default.xml цілком успішно впорався з поставленим завданням.

Рис. 2.3. Зображення для тестування роботи програми виявлення облич

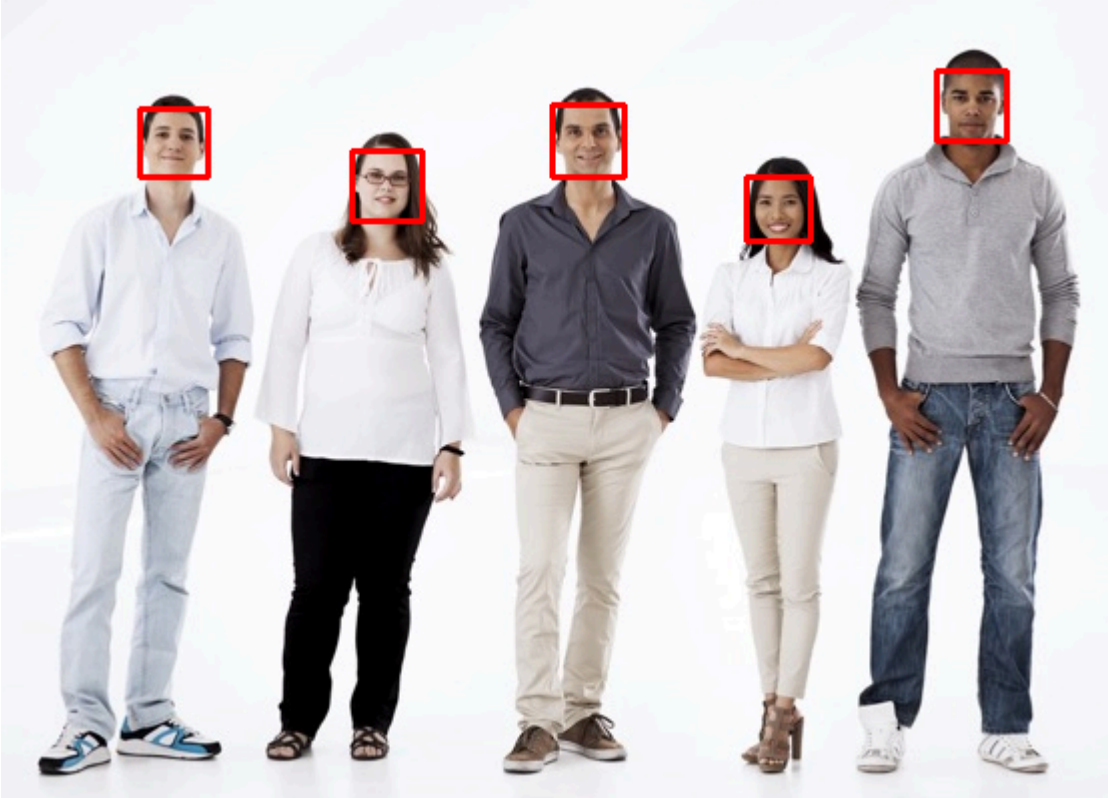


Рис. 2.4. Зображення, оброблене програмою виявлення облич

Слід зауважити, що бібліотека OpenCV зчитує файл зображення у форматі BGR (синій, зелений, червоний), а не у загальновживаному форматі RGB

(червоний, зелений, синій), що може спричинити проблеми при використанні різних бібліотек обробки зображень.

Іноді можливе неправильне виявлення обличчя. Щоб розв'язати цю проблему, функція виявлення облич в OpenCV (classifier.detectMultiScale()) дає змогу налаштувати його параметри, такі як scaleFactor, minNeighbors і minSize:

- параметр scaleFactor визначає те, наскільки зменшується розмір зображення на кожній ітерації алгоритму під час пошуку облич. Зазвичай він встановлюється у величину більшу за 1. Наприклад, якщо scaleFactor = 1.1, то зображення буде зменшуватися на 10% на кожній ітерації.
- параметр minNeighbors визначає, скільки “сусідів” має мати кожна зона обличчя для того, щоб бути визнаною як обличчя. Чим більше значення minNeighbors, тим більше обличчевих областей буде відфільтровано як неправильні. Це допомагає уникнути ложнопозитивних результатів.
- параметр minSize визначає найменший можливий розмір обличчя. Обличчя, менші за цей розмір, не будуть розпізнані. Встановлення цього параметру може допомогти уникнути виявлення дрібних артефактів або шумів як обличчя.

Цей метод також працює і для виявлення облич у відеопотоці з відеокамер. Програму наведено на Рис. 2.5.

```
import cv2
classifier = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
camera = cv2.VideoCapture(0)
while camera.isOpened():
    open, frame = camera.read()
    if not open:
        break
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    boxes= classifier.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

    for (x,y,w,h) in boxes:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

    cv2.imshow('Face detection', frame)
    key = cv2.waitKey(30)
    if key == 27:
        break
camera.release()
cv2.destroyAllWindows()
```

Рис. 2.5. Програма виявлення облич у відеопотоці

Спочатку підключено бібліотеку OpenCV і завантажено класифікатор haarcascade_frontalface_default.xml, який навчений на розпізнавання облич. Далі йде команда активації відеокамери:

camera = cv2.VideoCapture(0)

У наступних рядках програми:

- організовано цикл послідовного захоплення кадрів;
- читання кожного кадру;
- перетворення кольорового зображення кадру в чорно-біле зображення (у градаціях сірого);
- виявлення обличчя в поточному кадрі.

Далі виконується малювання прямокутної рамки навколо знайденого обличчя і показ обробленого кадру. Цей процес триває доти, доки користувач не натисне клавішу “q” для завершення процесу обробки кадрів і звільнення ресурсів.

У результаті роботи програми отримано такий результат (Рис. 2.6).

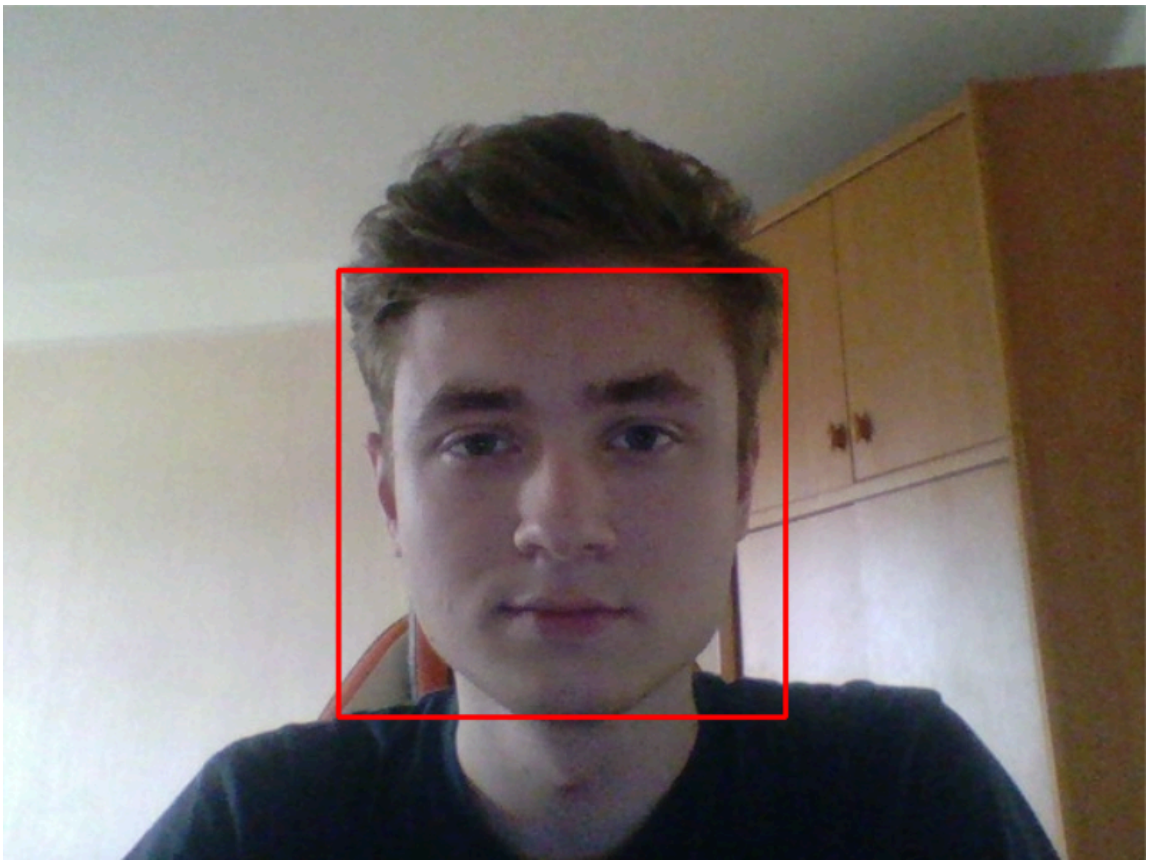


Рис. 2.6. Відеокадр, оброблений програмою виявлення облич

Крім OpenCV існує багато інших бібліотек для обробки зображень. Наприклад бібліотека face_recognition. Вона надає простий і зручний інтерфейс, легко інтегрується з іншими популярними бібліотеками Python, як-от OpenCV і PIL, що дає змогу створювати складніші та функціональніші програми. Бібліотека спроектована так, щоб бути максимально простою у використанні, надаючи розробникам високорівневі функції для виконання складних завдань без необхідності глибоко занурюватися у технічні деталі.

Бібліотека face_recognition дає змогу легко виявляти усі обличчя на зображенні або у відеопотоці.

Програму виявлення облич за допомогою бібліотеки face_recognition наведено на Рис. 2.7.

```
from PIL import Image
import face_recognition

img = face_recognition.load_image_file(r'C:/Foto/face1_2.jpg')
face_locations = face_recognition.face_locations(img)

count = 0
for face_location in face_locations:
    count += 1
    top, right, bottom, left = face_location
    print("Face {} Top: {}, Left: {}, Bottom: {}, Right: {}".format(count, top, left, bottom, right))

    face_image = img[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)
    pil_image.show()
```

Рис. 2.7. Програма виявлення облич за допомогою бібліотеки face_recognition

У програмі спочатку підключаються необхідні бібліотеки. Image з PIL (Python Imaging Library) для роботи із зображеннями та face_recognition для виявлення облич на зображеннях. Далі у змінну img завантажено тестове зображення face1.jpg. Функція face_recognition.face_locations виявляє усі обличчя на завантаженому зображенні та повертає список координат кожного виявленого обличчя. Ці координати зберігаються у змінній face_locations.

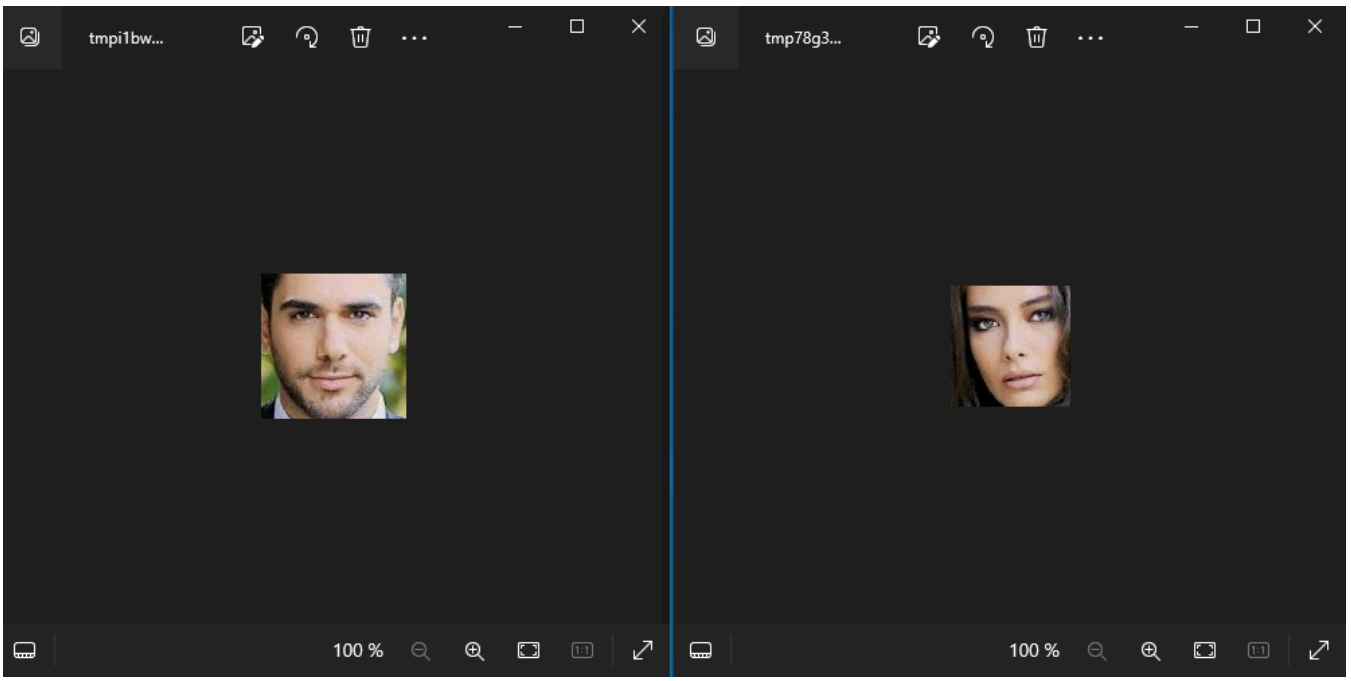
У циклі програма проходить по кожному елементу в face_locations. Витягує координати обличчя (верх, право, низ, ліворуч) з поточного елемента.

Друкує інформацію про поточне обличчя, включно з його номером і координатами. Далі кожне обличчя відображається в окремому вікні.

Для перевірки роботи програми було взято зображення, представлене на Рис. 2.8.

Рис. 2.8. Зображення для тестування роботи програми виявлення облич з використанням бібліотеки face_recognition

У результаті роботи програми отримано такий результат (Рис. 2.9).




```
Face 1 Top: 20, Left: 139, Bottom: 128, Right: 247
Face 2 Top: 86, Left: 315, Bottom: 176, Right: 404
```

Рис. 2.9. Зображення, оброблене програмою виявлення облич з використанням бібліотеки face_recognition

2.3 Автоматичне виявлення частин обличчя (Automatic Facial Features Detection)

На зображенні можна розпізнати не тільки обличчя, а й різні елементи на обличчях, наприклад, очі. Для вирішення таких завдань у бібліотеці OpenCV є вже готові каскади Хаара:

- haarcascade_eye.xml – очі;
- haarcascade_lefteye_2splits.xml – ліве око;
- haarcascade_righteye_2splits.xml – праве око.

Програму виявлення очей на зображенні наведено на Рис. 2.10.

```
import cv2
img = cv2.imread(r'C:/Foto/face2.jpg')
cv2.imshow('Input photo', img)
classifier_face = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
classifier_eye = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_eye.xml")
gray =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
boxes_faces = classifier_face.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

for (x,y,w,h) in boxes_faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    face_gray = gray[y:y+h, x:x+w]
    face_color = img[y:y+h, x:x+w]
    boxes_eyes = classifier_eye.detectMultiScale(face_gray, scaleFactor= 1.1, minNeighbors=10,minSize=(10, 10))
    for (ex, ey, ew, eh) in boxes_eyes:
        cv2.rectangle(face_color, (ex, ey), (ex + ew, ey + eh), (255, 255, 255), 2)

cv2.imshow('Faces and eyes detections', img)
cv2.imwrite(r'C:/Foto/face2_eyes_detection.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.10. Програма виявлення очей за допомогою бібліотеки OpenCV

У першому рядку підключено бібліотеку OpenCV. Далі у змінну img завантажено тестове зображення face2.jpg, і це зображення виведено користувачеві. Потім у змінну classifier_face завантажено класифікатор виявлення облич, а у змінну classifier_eye - класифікатор виявлення очей. Далі завантажене зображення конвертовано з кольорового в чорно-біле зображення (у градаціях сірого), і цей результат записано у змінну gray. Далі ініційовано процес виявлення облич, а на обличчі - виявлення очей і формування прямокутника навколо кожного виявленого

елемента. Потім оброблене зображення показано у вікні та збережено у файлі face2_eyes_detection.jpg. Останні два рядки програми закривають усі вікна.

Для перевірки роботи програми було взято зображення, представлене на Рис. 2.11.

Рис. 2.11. Зображення для тестування роботи програми виявлення очей

У результаті роботи програми отримано такий результат (Рис. 2.12).

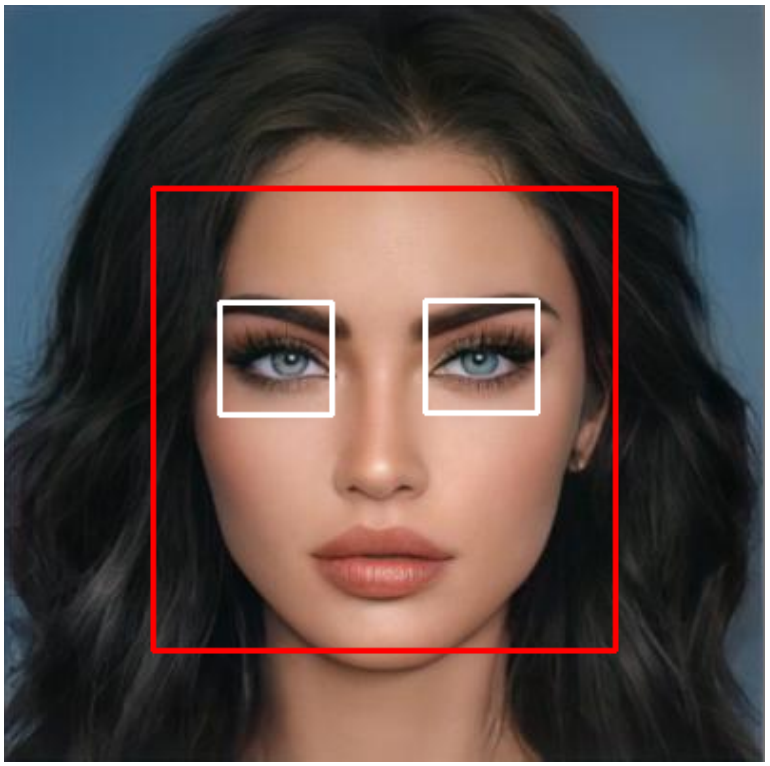


Рис. 2.12. Зображення, оброблене програмою виявлення очей

За допомогою бібліотеки face_recognition можна не тільки виявляти обличчя, але й ідентифікувати дев'ять орієнтирів з 72 ключовими точками на обличчі:

- підборіддя;
- ліва_брова;
- права_брова;
- перенісся;
- кінчик_носа;
- ліве_око;
- праве_око;
- верхня_губа;
- нижня_губа.

Програму виявлення частин обличчя на зображенні наведено на Рис. 2.13.

```
import cv2
import face_recognition
img = cv2.imread(r'C:/Foto/face2.jpg')
cv2.imshow('Input photo', img)
rgb = img[:, :, ::-1]
landmarks = face_recognition.face_landmarks(rgb)
count = 0
for landmark in landmarks:
    for part in landmark.keys():
        print(part)
        a = landmark[part]
        for idx, item in enumerate(a):
            if idx == len(a) -1:
                break
            cv2.line(img, item, a[idx + 1], [255, 255, 255], 2)
        for point in landmark[part]:
            img = cv2.circle(img, point, 2, (0, 0, 255), 2)
            count = count +1
            print(point)
print(count)
cv2.imshow('Face Landmarks', img)
cv2.imwrite(r'C:/Foto/face2_landmarks.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.13. Програма виявлення частин обличчя за допомогою бібліотеки face_recognition

У програмі спочатку підключаються необхідні бібліотеки: OpenCV та face_recognition. Далі у змінну img завантажено тестове зображення face2.jpg і це зображення виведено користувачеві. Потім зображення перетворюється з формату BGR, що використовується в OpenCV, у формат RGB, який використовується в бібліотеці face_recognition. Для виявлення облич і вилучення їхніх ключових точок використовується функція face_recognition.face_landmarks. Далі перебираються і відзначаються ключові точки і малюються лінії між ними. Друкуються координати кожної ключової точки та її назва. Потім оброблене зображення показано у вікні та збережено у файлі face2_landmarks.jpg. Останні два рядки програми закривають усі вікна.

У результаті роботи програми отримано такий результат (Рис. 2.14).

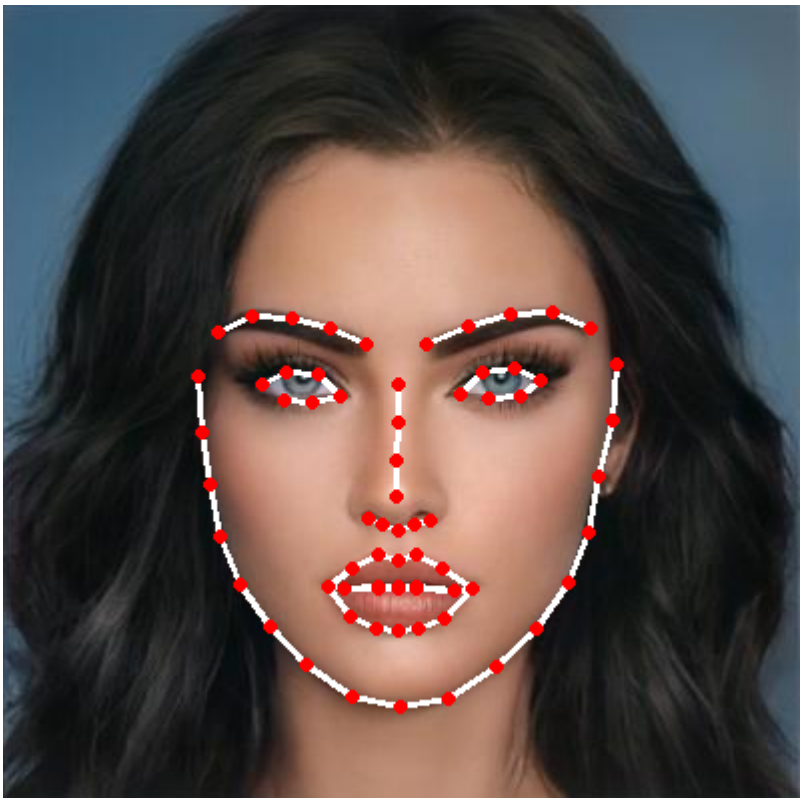


Рис. 2.14. Зображення, оброблене програмою виявлення частин обличчя за допомогою бібліотеки face_recognition

2.4 Визначення характеристик людини за зображенням її обличчя (Objectís Features Determination)

Бібліотека DeepFace дозволяє використовувати різні моделі глибокого навчання для виявлення, розпізнавання, верифікації та аналізу облич, а також отримувати інформацію про вік, стать, расу та емоції. Вона підтримує кілька глибоких нейронних мереж, включно з TensorFlow і Keras, що робить її гнучкою і сумісною з різними проектами. DeepFace написана на мові Python і легко інтегрується з іншими бібліотеками для комп'ютерного зору і глибокого навчання, такими як OpenCV і TensorFlow.

Програму визначення ознак людини за зображенням її обличчя наведено на Рис. 2.15.

```
import cv2
from deepface import DeepFace

foto=r'C:/Foto/face3.jpg'
img_show = cv2.imread(foto)
img = DeepFace.analyze(foto)[0]

print("Age: ", int(img["age"]))
print("Gender: ", max(img["gender"],key=img["gender"].get))
print("Race: ", img["dominant_race"])
cv2.imshow('Input photo', img_show)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.15. Програма визначення ознак людини за зображенням її обличчя

У програмі спочатку підключаються необхідні бібліотеки: OpenCV та deepface. Далі завантажено тестове зображення face3.jpg. Бібліотека OpenCV використовується для відображення тестового зображення у вікні. Зображення аналізується за допомогою функції DeepFace.analyze(), яка повертає список із результатами аналізу. Оскільки повертається список, [0] бере перший елемент списку, який містить результати аналізу особи на зображенні.

Далі відбувається виведення результатів:

- перетворюється значення віку в ціле число і виводиться;
- визначається стать за максимальним значенням ймовірності серед можливих значень і обирається ключ із найбільшим значенням ймовірності;
- виводиться домінуюча раса.

У результаті роботи програми отримано такий результат (Рис. 2.16).

Рис. 2.16. Зображення, оброблене програмою виявлення ознак людини

2.5 Аналіз виразів обличчя (Facial Expression Analysis)

На зображенні можна розпізнати не тільки окремі елементи, а й різні емоції, наприклад, посмішку. Для вирішення таких завдань у бібліотеці OpenCV є вже готові каскади Хаара – haarcascade_smile.xml.

Програму виявлення посмішки на обличчі наведено на Рис. 2.17.

```
import cv2
img = cv2.imread(r'C:/Foto/face4.jpg')
cv2.imshow('Input photo', img)
classifier_face = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
classifier_smile = cv2.CascadeClassifier(cv2.data.haarcascades +'haarcascade_smile.xml')
gray =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
boxes_faces = classifier_face.detectMultiScale(gray, scaleFactor=1.1,  minNeighbors=5, minSize=(30, 30))

for (x,y,w,h) in boxes_faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    face_gray = gray[y:y+h, x:x+w]
    face_color = img[y:y+h, x:x+w]
    boxes_smile = classifier_smile.detectMultiScale(face_gray, scaleFactor= 1.7, minNeighbors=5, minSize=(10, 10))
    for (xx, yy, ww, hh) in boxes_smile:
        cv2.rectangle(face_color, (xx, yy), (xx + ww, yy + hh), (255, 255, 255), 2)

cv2.imshow('Faces and smile detections', img)
cv2.imwrite(r'C:/Foto/face4_smile_detection.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.17. Програма виявлення посмішки за допомогою бібліотеки OpenCV

У першому рядку підключено бібліотеку OpenCV. Далі у змінну img завантажено тестове зображення face4.jpg, і це зображення виведено користувачеві. Потім у змінну classifier_face завантажено класифікатор виявлення облич, а у змінну classifier_smile - класифікатор виявлення посмішки. Далі завантажене зображення конвертовано з кольорового в чорно-біле зображення (у градаціях сірого), і цей результат записано у змінну gray. Далі ініційовано процес виявлення облич, а на обличчі - виявлення посмішки і формування прямокутника навколо кожного виявленого елемента. Потім оброблене зображення показано у вікні та збережено у файлі face4_smile_detection.jpg. Останні два рядки програми закривають усі вікна.

Для перевірки роботи програми було взято зображення, представлене на Рис. 2.18.

Рис. 2.18. Зображення для тестування роботи програми виявлення посмішки

У результаті роботи програми отримано такий результат (Рис. 2.19).

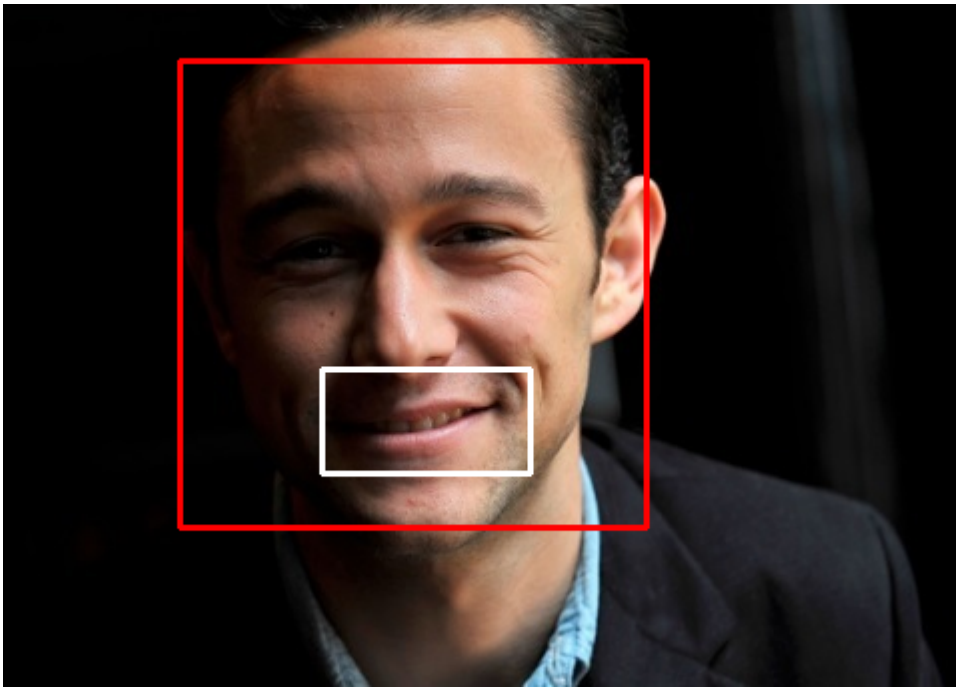


Рис. 2.19. Зображення, оброблене програмою виявлення посмішки

За допомогою бібліотеки DeepFace можна також отримувати інформацію про емоції людини.

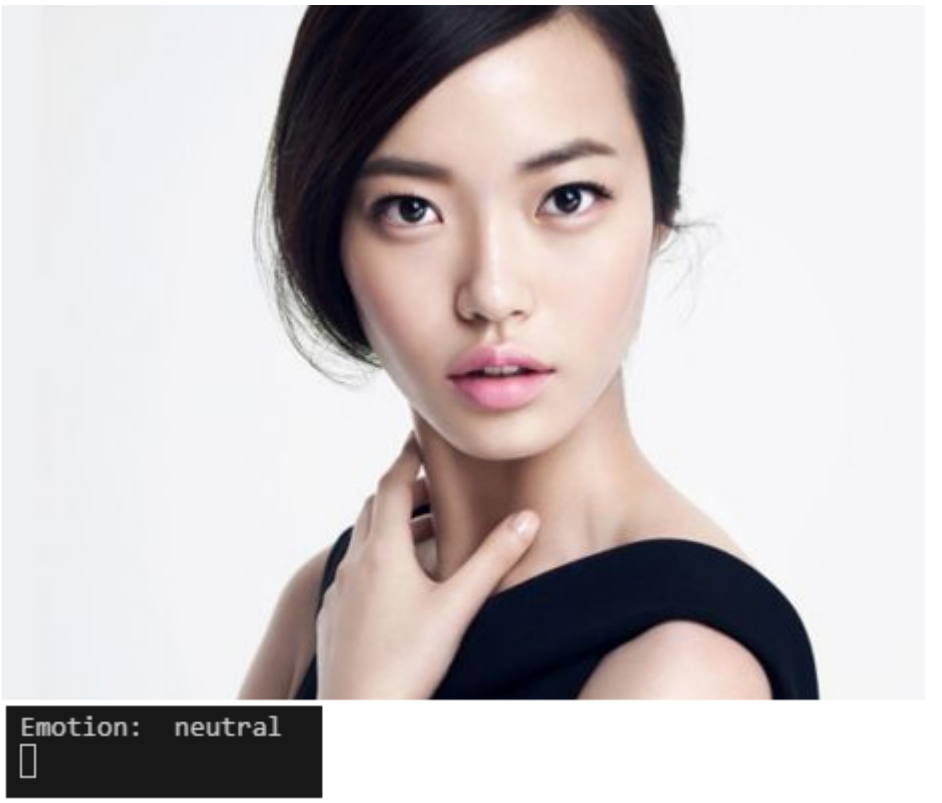
Програму розпізнавання емоції людини наведено на Рис. 2.20.

```
import cv2
from deepface import DeepFace

foto=r'C:/Foto/face3.jpg'
img_show = cv2.imread(foto)
img = DeepFace.analyze(foto,['emotion'])[0]

print("Emotion: ", img["dominant_emotion"])
cv2.imshow('Input photo', img_show)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.20. Програма розпізнавання емоції людини за зображенням її обличчя

У програмі спочатку підключаються необхідні бібліотеки: OpenCV та deepface. Далі завантажено тестове зображення. Бібліотека OpenCV використовується для відображення тестового зображення у вікні. Зображення аналізується за допомогою функції DeepFace.analyze(), яка повертає список із результатами аналізу. Оскільки повертається список, [0] бере перший елемент списку, який містить результати аналізу особи на зображенні. Далі відбувається виведення результатів – домінуюча емоція людини.

У результаті роботи програми отримано такий результат (Рис. 2.21).



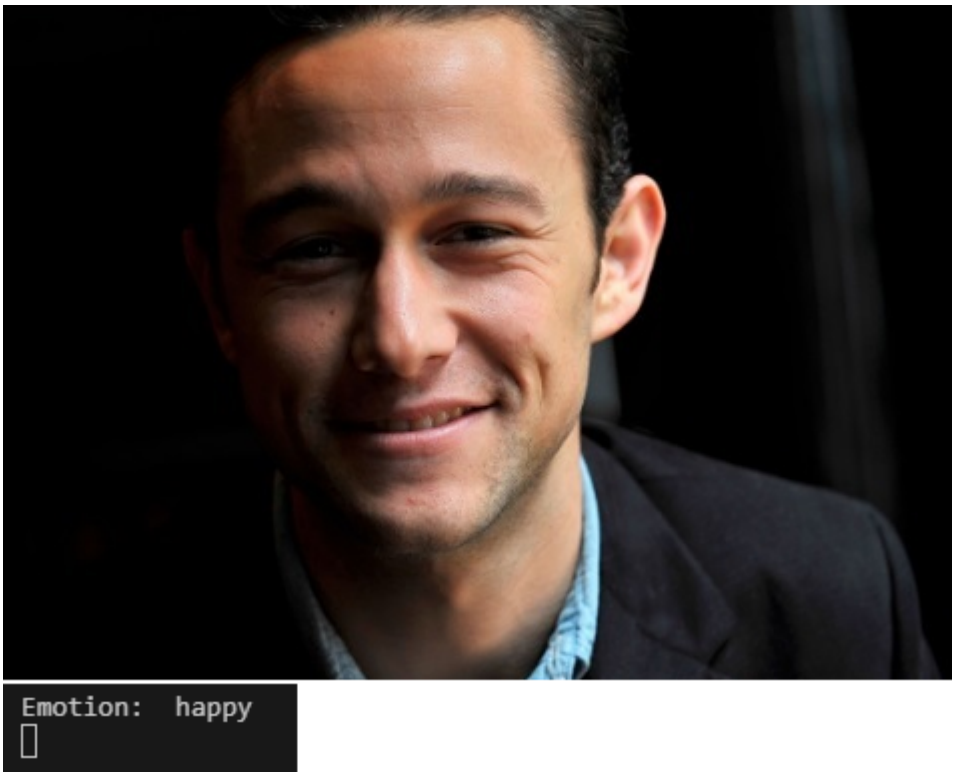


Рис. 2.21. Зображення, оброблене програмою розпізнавання емоції

2.6 Розпізнавання облич (Face Recognition)

Розпізнавання облич означає ідентифікацію людини за її обличчям. Спочатку потрібно виявити обличчя, а потім виконати ідентифікацію.

Програму розпізнавання облич наведено на Рис. 2.22.

```
import face_recognition, os, cv2
def recognition(directory):
    known_face_encodings = []
    known_face_names = []
    for f in os.listdir(directory):
        if f.endswith(".jpg") or f.endswith(".jpeg"):
            image_path = os.path.join(directory, f)
            face_image = face_recognition.load_image_file(image_path)
            face_encoding = face_recognition.face_encodings(face_image)[0]
            name = os.path.splitext(f)[0].split('.')[0]
            known_face_encodings.append(face_encoding)
            known_face_names.append(name)
    return known_face_encodings, known_face_names
known_face_encodings, known_face_names = recognition("C:/Foto/Faces")
test_img = face_recognition.load_image_file("C:/Foto/Faces/Test/face2.jpg")
face_locations = face_recognition.face_locations(test_img)
face_encodings = face_recognition.face_encodings(test_img, face_locations)
face_names = []
for face_encoding in face_encodings:
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"
    if True in matches:
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]
    face_names.append(name)
for (top, right, bottom, left), name in zip(face_locations, face_names):
    cv2.rectangle(test_img, (left, top), (right, bottom), (255, 0, 0), 2)
    cv2.putText(test_img, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 255), 1)
cv2.imshow('Test Image', cv2.cvtColor(test_img, cv2.COLOR_RGB2BGR))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рис. 2.22. Програма розпізнавання облич

У програмі спочатку підключаються необхідні бібліотеки: face_recognition, OpenCV, які необхідні для обробки зображень, виявлення і розпізнавання облич, та модуль os, який у мові програмування Python надає функції для взаємодії з операційною системою. Далі функція recognition(directory) завантажує зображення облич із зазначеної директорії, кодує обличчя у вектори ознак і повертає список відомих векторів ознак known_face_encodings і відповідні імена known_face_names.

На наступному кроці відбувається завантаження та кодування відомих облич та завантаження тестового зображення. Потім програма знаходить розташування облич на тестовому зображенні за допомогою функції face_recognition.face_locations(test_img). Після цього програма кодує обличчя на тестовому зображенні у вектори ознак з використанням функції face_recognition.face_encodings(test_img, face_locations). Потім вона порівнює кожен вектор ознак з відомими векторами ознак за допомогою функції face_recognition.compare_faces(known_face_encodings, face_encoding). Якщо знайдено збіг, програма визначає ім'я відомої особи, що відповідає поточному вектору ознак. Наприкінці програма відображає тестове зображення з позначеними обличчями та їхніми іменами.

Для перевірки роботи програми було взято зображення, представлене на Рис. 2.23.

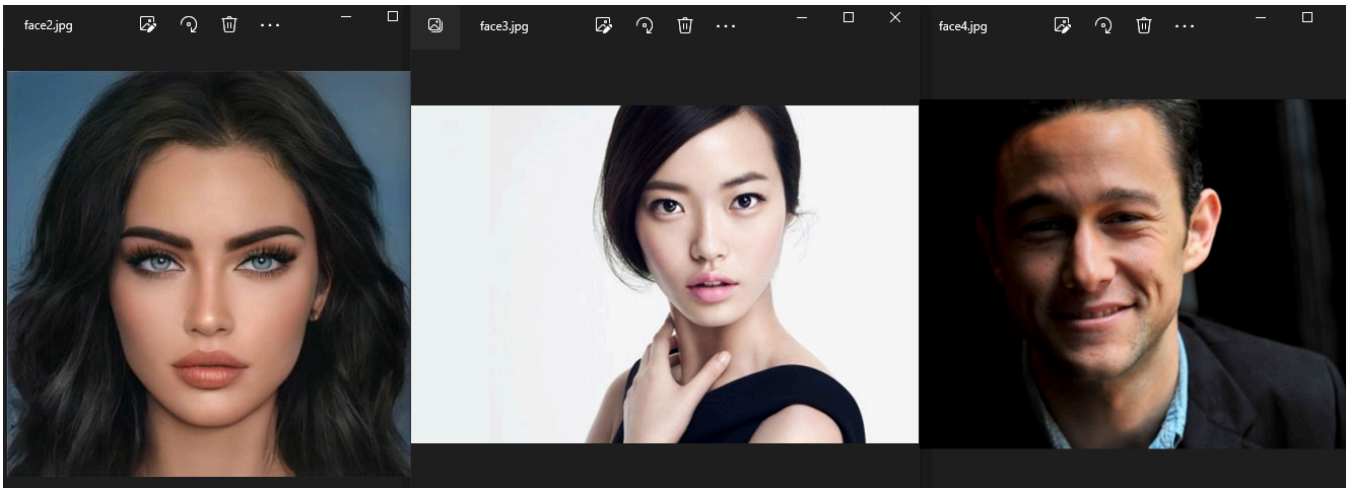


Рис. 2.23. Зображення відомих облич (face2, face3, face4)

У результаті роботи програми отримано такий результат (Рис. 2.24 - Рис. 2.25).

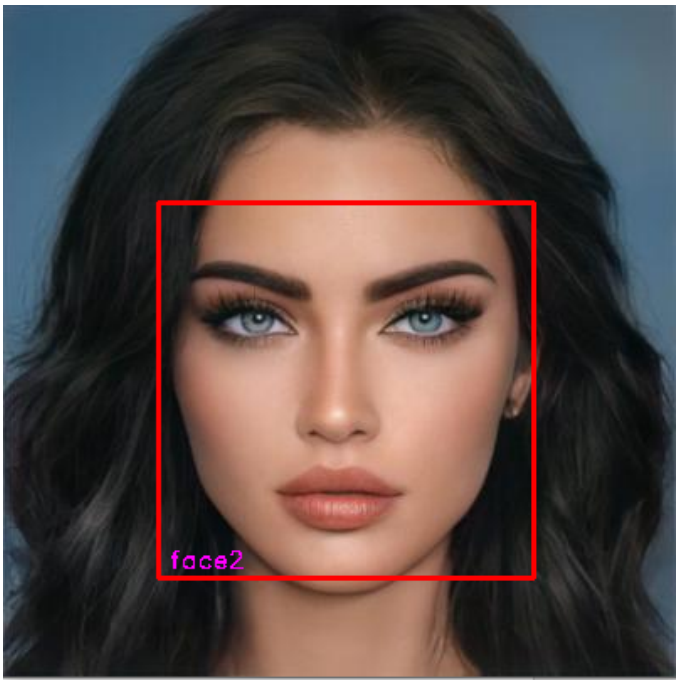


Рис. 2.24. Зображення, оброблене програмою розпізнавання облич (відоме обличчя)

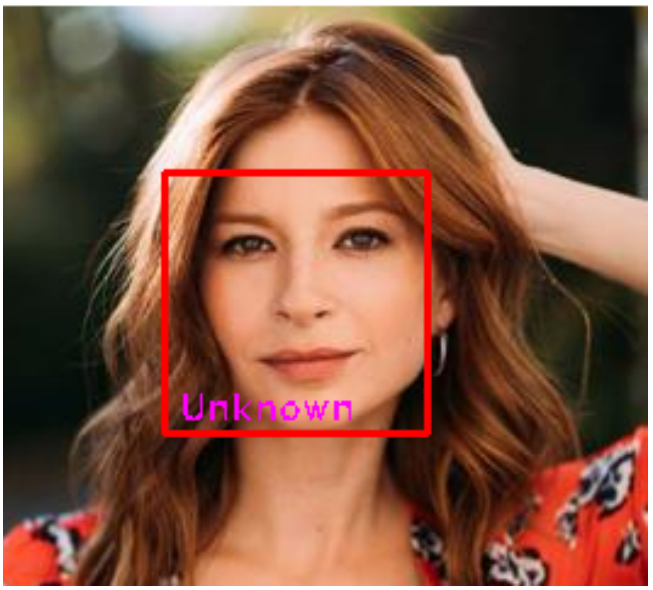


Рис. 2.25. Зображення, оброблене програмою розпізнавання облич (невідоме обличчя)

3. РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ

3.1 Середовище та платформа розробки

Автоматизована система виявлення та розпізнавання облич реалізована на мові програмування Python з використанням бібліотек комп'ютерного зору OpenCV, face_recognition, PIL, бібліотеки для роботи з числовими даними – NumPy, аналізу даних – Pandas, бібліотеки для створення графічних інтерфейсів – tkinter.

Python (Рис. 3.1) - це високорівнева мова програмування загального призначення, що характеризується динамічною суворою типізацією та автоматичним керуванням пам'яттю. Вона спрямована на підвищення продуктивності розробника, покращення читаності коду та його якості, а також забезпечення переносимості написаних на ній програм. Ця мова повністю об'єктно-орієнтована: усе є об'єктами [16, 17]. Унікальною особливістю є використання пробільних відступів для виділення блоків коду. Мінімалістичний синтаксис скорочує необхідність частого звернення до документації. Мова відома як інтерпретована і широко використовується для написання скриптів [18]. Недоліками є здебільшого нижча швидкість виконання та вище споживання пам'яті програм, написаних на ній, у порівнянні з аналогічним кодом, створеним компільованими мовами, такими як C або C++.

Рис. 3.1. Мова програмування Python

Python підтримує кілька парадигм програмування, таких як імперативне, процедурне, структурне, об'єктно-орієнтоване, метапрограмування і функціональне програмування [19, 20]. Завдання узагальненого програмування вирішуються за рахунок динамічної типізації. Аспектно-орієнтоване програмування частково підтримується через декоратори, більш повноцінна підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси — динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопотокових обчислень із глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується поділ програм на модулі, які можна об'єднувати в пакети [21, 22].

Стандартна бібліотека включає великий набір корисних функцій, що переносяться, починаючи від функціоналу для роботи з текстом і закінчуючи засобами для написання мережевих додатків. Додаткові можливості, такі як математичне моделювання, робота з обладнанням, написання веб-додатків або розробка ігор можуть реалізовуватися за допомогою великої кількості сторонніх бібліотек, а також інтеграцією бібліотек, написаних на C або C++, при цьому і сам інтерпретатор Python може інтегруватися в проекти, написані цими мовами [23, 24]. Існує і спеціалізований репозиторій програмного забезпечення, написаного на Python - PyPI. Цей репозиторій надає засоби для простої установки пакетів в операційну систему та став стандартом де-факто для Python.

Python став однією з найпопулярніших мов програмування. Її використовують в аналізі даних, машинному навчанні, DevOps, веб-розробці та інших галузях. Завдяки своїй читабельності, простому синтаксису і відсутності необхідності в компіляції, Python ідеально підходить для навчання програмуванню, дозволяючи зосередитися на вивченні алгоритмів, концепцій і парадигм. Процес налагодження та експериментування значно спрощується, оскільки мова є інтерпретованою. Багато великих компаній, як-от Google і Facebook, активно використовують Python у своїй роботі.

OpenCV (Open Source Computer Vision Library) - бібліотека з відкритим вихідним кодом, призначена для задач комп'ютерного зору і машинного навчання [25, 26]. OpenCV надає багатий набір інструментів для обробки зображень і відео, а також для розробки додатків, пов'язаних з аналізом візуальної інформації (Рис. 3.2).

Рис. 3.2. Бібліотека OpenCV

Основні можливості бібліотеки OpenCV:

1) обробка зображень і відео:
   - фільтрація і перетворення зображень (розмиття, згладжування, фільтри Собеля і Канні);
   - кольорові перетворення (конвертація між колірними просторами, такими як BGR, RGB, HSV та інших);
   - геометричні перетворення (масштабування, повороти, трансформації);
   - виявлення і розпізнавання граней і об'єктів;

2) аналіз форм і контурів:
   - виявлення контурів та їх аналіз;
   - апроксимація контурів і визначення їхніх характеристик (площа, периметр, момент тощо);
   - виявлення і зіставлення ключових точок (наприклад, детектори SURF, SIFT);
3) машинне навчання:
   - підтримка різних алгоритмів машинного навчання, таких як K-найближчих сусідів (KNN), дерево рішень, наївний байєсівський класифікатор тощо;
   - класифікація об'єктів і розпізнавання образів
4) комп'ютерний зір:
   - виявлення і розпізнавання облич;
   - виявлення рухів і відстеження об'єктів у відео;
   - калібрування камери та відновлення 3D-інформації;
5) інтеграція з іншими бібліотеками:
   - підтримка інтеграції з бібліотеками глибокого навчання, такими як TensorFlow і PyTorch;
   - можливість роботи з бібліотеками NumPy і SciPy для чисельних обчислень.

OpenCV є потужним інструментом для розробників і дослідників у галузі комп'ютерного зору та машинного навчання. З її допомогою можна розв'язувати широкий спектр завдань, від простих перетворень зображень до складних алгоритмів розпізнавання та аналізу. Завдяки своїй універсальності та доступності, OpenCV є однією з найпопулярніших бібліотек для обробки зображень і відео [27-29].

Face_recognition - бібліотека з відкритим вихідним кодом, призначена для простого й ефективного розпізнавання облич. Вона заснована на потужних алгоритмах глибокого навчання [30, 31]. Бібліотека надає зручний інтерфейс для виконання різноманітних завдань, пов'язаних із розпізнаванням облич, як-от виявлення облич, вилучення їхніх ознак і зіставлення облич (Рис. 3.3).

Рис. 3.3. Бібліотека face_recognition

Основні можливості бібліотеки face_recognition:

- виявлення облич: використовуючи алгоритми на основі гістограми орієнтованих градієнтів (HOG) і згорткових нейронних мереж (CNN), бібліотека здатна точно виявляти обличчя на зображеннях і у відео;
- розпізнавання облич: бібліотека face_recognition може визначати, хто зображений на фотографії, порівнюючи обличчя з відомими обличчями з бази даних. Алгоритми бібліотеки дають змогу витягувати унікальні ознаки облич і зіставляти їх із високою точністю;
- вилучення ознак облич: бібліотека дає змогу витягувати ключові ознаки обличчя, які являють собою багаторозмірні вектори. Ці ознаки використовуються для подальшого зіставлення і розпізнавання;
- зіставлення облич: порівняння облич виконується з використанням евклідової відстані між векторами ознак. Бібліотека підтримує як один-до-одного, так і один-до-багатьох режими зіставлення;

- виявлення частин облич: бібліотека надає можливість виявлення частин обличчя, таких як очі, ніс, рот і щелепа, що корисно для різних додатків, таких як анімація і фільтри.

Переваги бібліотеки face_recognition:

- простота використання: інтуїтивно зрозумілий API дає змогу швидко інтегрувати функціональність розпізнавання облич у проєкти без необхідності глибоких знань у галузі машинного навчання та комп'ютерного зору;
- точність і продуктивність: використовуючи передові алгоритми і моделі, бібліотека забезпечує високу точність розпізнавання при порівняно низькому обчислювальному навантаженні;
- кросплатформеність: бібліотека працює на різних операційних системах, включно з Windows, macOS і Linux;
- спільнота і документація: бібліотека має активне співтовариство розробників і користувачів, а також хорошу документацію і безліч прикладів використання.

Бібліотека face_recognition є потужним інструментом для розв'язання завдань розпізнавання облич, надаючи розробникам зручний і ефективний спосіб інтеграції цієї функціональності у свої додатки. Будь то аналіз зображень, безпека, управління доступом або розважальні програми, face_recognition пропонує широкий спектр можливостей для роботи з обличчями.

PIL (Python Imaging Library) - бібліотека для роботи із зображеннями, яка надає широкий набір інструментів для відкриття, обробки та збереження різних форматів зображень [32, 33]. PIL швидко стала однією з найпопулярніших бібліотек для обробки зображень у Python. Хоча PIL більше не оновлюється, її клон під назвою Pillow продовжує активно розвиватися і підтримувати сумісність із сучасними версіями Python (Рис. 3.4).

Рис. 3.4. Бібліотека Pillow

Основні можливості бібліотеки PIL/Pillow:

- відкриття та збереження зображень: бібліотека підтримує безліч форматів зображень, таких як JPEG, PNG, BMP, GIF, TIFF та інші. Це дає змогу легко відкривати і зберігати зображення в потрібних форматах;
- обробка зображень: PIL/Pillow надає функції для зміни розміру, обрізки, повороту, дзеркального відображення і перетворення зображень. Це уможливлює виконання різних операцій з обробки зображень з мінімальними зусиллями;
- робота з кольорами: бібліотека дає змогу змінювати колірні простори зображень, конвертувати зображення у відтінки сірого, застосовувати колірні фільтри та змінювати яскравість, контраст і насиченість;
- текст на зображеннях: PIL/Pillow включає інструменти для додавання тексту на зображення з використанням різних шрифтів і стилів, що корисно для створення водяних знаків, анотацій та інших цілей;
- фільтри та ефекти: бібліотека підтримує застосування різних фільтрів і ефектів, таких як розмиття, контурування, різкість і багато іншого, що дає змогу покращувати і змінювати зображення по-різному;

- обробка пікселів: PIL/Pillow надає доступ до окремих пікселів зображення, що дає змогу виконувати низькорівневе опрацювання та маніпуляції із зображеннями.

Переваги бібліотеки PIL/Pillow:

- простота використання: інтуїтивно зрозумілий API дає змогу швидко починати роботу із зображеннями навіть розробникам-початківцям;
- широкий набір інструментів: PIL/Pillow надає безліч функцій для виконання різних завдань з обробки зображень;
- кросплатформеність: бібліотека працює на різних операційних системах, включно з Windows, macOS і Linux;
- підтримка форматів: PIL/Pillow підтримує роботу з широким спектром форматів зображень, що робить її універсальним інструментом для різних додатків.

PIL/Pillow є потужною і гнучкою бібліотекою для роботи із зображеннями в мові програмування Python. Завдяки простоті використання, багатому набору функцій і широкій підтримці форматів, вона залишається популярним вибором серед розробників, які працюють із зображеннями в різних додатках, від наукових досліджень до веб-розробки та створення графічного контенту.

Бібліотека NumPy (Numerical Python) є одним з основних інструментів для роботи з числовими даними в Python (Рис. 3.5).

Ключові особливості та можливості бібліотеки NumPy [34-36]:

- основний об'єкт у NumPy - це багатовимірний масив, який являє собою таблицю елементів одного типу (зазвичай числових). Масиви NumPy надають ефективні методи для зберігання і маніпулювання даними, що робить їх ідеальним інструментом для наукових обчислень.
- NumPy надає велику кількість математичних функцій та операцій для роботи з масивами. Універсальні функції дають змогу виконувати елементарні операції (наприклад, додавання, множення) поелементно над масивами, що робить код більш компактним і ефективним.

- NumPy підтримує потужні можливості для індексації та зрізів масивів, даючи змогу вибирати та модифікувати елементи масиву за певними критеріями, а також виконувати операції з підмасивами.
- NumPy може виконувати операції між масивами різних форм і розмірів без явного копіювання даних. Це робить код більш читабельним і ефективним.
- NumPy має функції для виконання основних операцій лінійної алгебри, таких як множення матриць, знаходження визначника, розв'язування систем лінійних рівнянь тощо.
- бібліотека NumPy містить генератори псевдовипадкових чисел, що дають змогу створювати випадкові дані та розподіли.
- NumPy часто використовується разом з іншими бібліотеками для наукових обчислень у Python, такими як SciPy, Matplotlib, Pandas та іншими.

Завдяки своїй простоті використання, ефективності та широким можливостям NumPy є невід'ємним компонентом в екосистемі наукових обчислень і аналізу даних у Python.

Рис. 3.5. Бібліотека NumPy

Бібліотека Pandas - потужний інструмент для аналізу даних і маніпуляції з ними в Python. Вона надає високорівневі структури даних і функції, які полегшують роботу з табличними даними (Рис. 3.6).

Серед ключових можливостей Pandas [37-39]:

- основна структура даних у Pandas - DataFrame, являє собою двовимірну таблицю з індексами рядків і стовпців. DataFrame дає змогу легко читати, записувати і маніпулювати даними;
- одновимірний масив з мітками - Series, який можна використовувати для зберігання даних різних типів. Series часто використовується для представлення одного стовпця в DataFrame;
- потужні функції для читання і запису даних з різних джерел, таких як CSV, Excel, SQL баз даних, JSON і багатьох інших форматів;
- можливість виконання різних операцій з даними, таких як сортування, фільтрація, об'єднання, групування, агрегація та багато іншого;
- широкий набір методів для обробки пропущених значень і очищення даних;
- інтеграція з іншими бібліотеками Python, як-от NumPy і Matplotlib, для ефективнішої роботи з даними та візуалізації результатів.

Pandas є одним із найпопулярніших інструментів для аналізу даних у Python завдяки своїй простоті використання, гнучкості та високій продуктивності.

Рис. 3.6. Бібліотека Pandas

Tkinter - це стандартна бібліотека Python, призначена для створення графічних користувацьких інтерфейсів. Вона надає набір інструментів і віджетів для розроблення додатків із графічним інтерфейсом, таких як вікна, кнопки, текстові поля та інші елементи керування (Рис. 3.7).

Основні особливості бібліотеки tkinter [40, 41]:

- простота використання: tkinter вбудований у стандартну бібліотеку Python, що робить його доступним без необхідності встановлення додаткових пакетів. Інтерфейс програмування tkinter інтуїтивно зрозумілий і легко освоюється навіть для початківців-розробників.
- широкий набір віджетів: tkinter надає різні віджети, такі як кнопки (Button), мітки (Label), текстові поля (Entry), багаторядкові текстові поля (Text), прапорці (Checkbutton), радіокнопки (Radiobutton), списки, що випадають (OptionMenu). Також доступні контейнерні віджети, такі як Frame, LabelFrame і PanedWindow, які дають змогу групувати й організовувати інші віджети.
- гнучкість і потужність: tkinter дає змогу створювати складні та багаторівневі інтерфейси. Підтримує роботу з різними шрифтами, кольорами та стилями для кастомізації зовнішнього вигляду додатків. Є підтримка подієвого програмування що дає змогу легко обробляти користувацькі дії, такі як натискання кнопок або введення тексту.
- крос-платформеність: додатки, створені з використанням бібліотеки tkinter, працюють на різних платформах, включно з Windows, macOS і Linux, без необхідності внесення змін до коду.
- інтеграція з іншими бібліотеками: tkinter можна використовувати в поєднанні з іншими бібліотеками Python для створення багатофункціональних додатків.

Рис. 3.7. Бібліотека tkinter

Бібліотека tkinter є потужним і простим інструментом для створення графічних інтерфейсів у Python. Завдяки своїй інтеграції в стандартну бібліотеку і широкій підтримці віджетів, вона дає змогу розробникам швидко і легко створювати користувацькі інтерфейси для своїх додатків.

При створенні системи виявлення та розпізнавання облич було використано редактор коду Visual Studio Code.

Visual Studio Code - потужний редактор вихідного коду, який працює на різних операційних системах: Windows, macOS, Linux. Він має вбудовану підтримку таких мов програмування як JavaScript, TypeScript і серверної платформи для виконання JavaScript коду - Node.js, а також велику систему розширень для інших мов і середовищ виконання: C++, C#, Java, Python, PHP, Go, .NET.

Visual Studio Code вирізняється своєю легкістю і швидкістю роботи, а також можливістю налаштовувати його під індивідуальні потреби розробника.

Серед ключових можливостей Visual Studio Code:

- інтегрована підтримка налагодження коду;
- автодоповнення коду та підсвічування синтаксису;
- гнучка система налаштування і наявність тем оформлення;
- можливість роботи з розширеннями для розширення функціональності;
- інтеграція з системами керування версіями, такими як Git;
- підтримка розробки веб-додатків, включно з HTML, CSS і JavaScript;

- інструменти для роботи з контейнерами і хмарами, такі як Docker і Azure.

Основну частину екрана редактора (Рис. 3.8) займає вікно з текстовим вмістом файлу. У лівій частині екрана розташовані вкладки активного меню, в якому знаходяться головні функції редактора.

При запуску програми, за замовчуванням відкривається вкладка провідника. У неї виводяться список відкритих файлів і каталог відкритої папки.

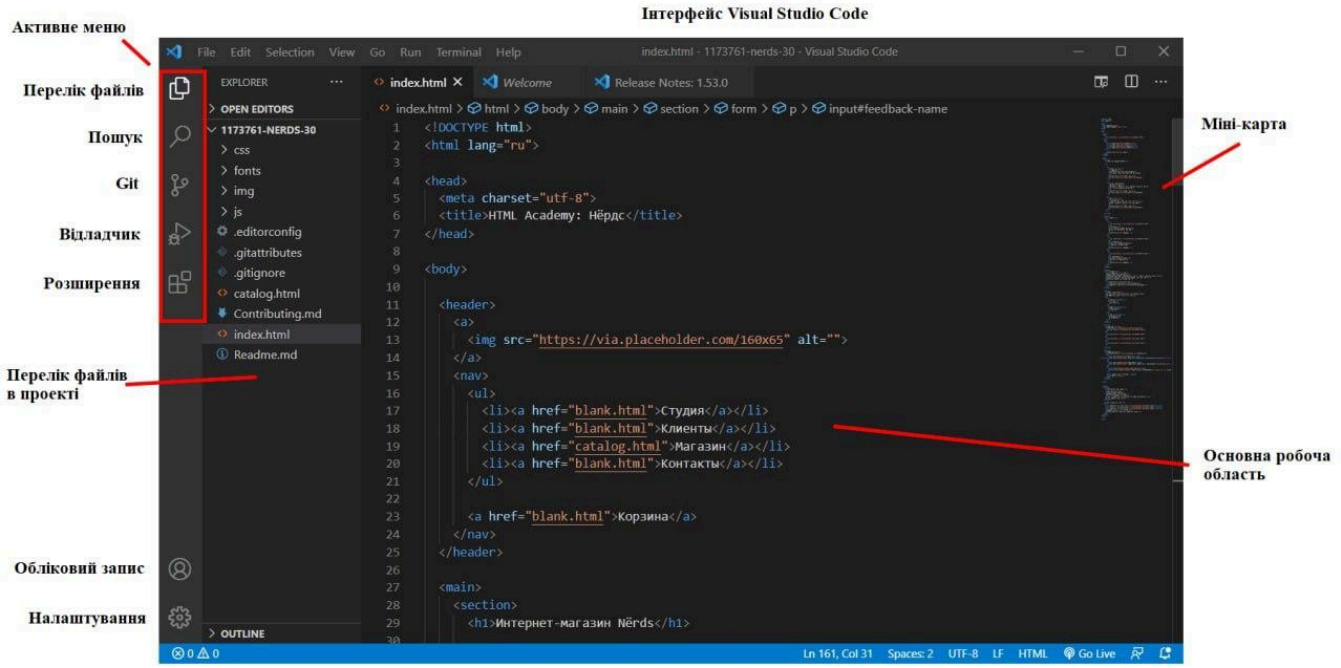


Рис. 3.8. Інтерфейс редактора Visual Studio Code

3.2. Загальна структура проекту

Система розпізнавання облич має наступну файлову структуру (Рис. 3.9). Каталог даних data - призначений для зберігання файлів зображень облич. Файл database.csv - призначений для зберігання інформацію про кожну особу, таку як ідентифікатор, ім'я та статус. Файл FaceRecognationSystem.py - це основна програма. Лістинг коду наведено у доданку Б. Файл train_data.npy використовується для зберігання даних, які необхідні для розпізнавання облич. Зокрема, він містить навчені дані облич та їхні ідентифікатори.

Рис. 3.9. Файлова структура системи розпізнавання облич

Основна програма - файл FaceRecognationSystem.py, має наступну структуру:

- імпорт бібліотек;
- ініціалізація глобальних змінних;
- визначення допоміжних функцій для роботи із зображеннями та файлами;
- створення графічного інтерфейсу користувача;
- основний цикл, який обробляє кадри з камери та оновлює інтерфейс користувача.

3.2.1 Імпорт бібліотек

У програмі спочатку підключаються необхідні бібліотеки (Рис. 3.10) для роботи із зображеннями, розпізнаванням облич, створення інтерфейсу користувача та роботи з файлами:

- OpenCV (cv2) - для захоплення відео з камери;

- numpy - бібліотека для роботи з масивами і матрицями, що містить математичні функції високого рівня для їх обробки;
- face_recognition - для розпізнавання облич;
- tkinter - для створення графічного інтерфейсу користувача;
- os - бібліотека для взаємодії з файловою та операційною системою;
- Pillow (PIL) - бібліотека для роботи із зображеннями, що дає змогу відкривати, змінювати і зберігати різні формати зображень;
- pandas - бібліотека для роботи з даними, що надає структури даних та інструменти для аналізу;
- CSV - для роботи з CSV (Comma-Separated Values) файлами.

```
import cv2
import numpy as np
import face_recognition
import tkinter as tk
from tkinter import ttk, messagebox
import os
from PIL import Image, ImageTk
import pandas as pd
import csv
```

Рис. 3.10. Імпорт бібліотек

3.2.2 Ініціалізація глобальних змінних

Змінні (Рис. 3.11), які використовуються для зберігання шляхів до даних; стану реєстрації; кількості зображень облич, які система використовує для навчання; даних користувача та змінні для зберігання облич і їхніх Id.

```
dataPath = r"C:/Foto/FaceRecognitionSystem/data"
databaseFile = r"C:/Foto/FaceRecognitionSystem/database.csv"

register = False
recognizeFrame = False
sampleNum = 20
name = ""
Id = ""
status = ""

faces = []
Ids = []
```

Рис. 3.11. Глобальні змінні

3.2.3 Визначення допоміжних функцій для роботи із зображеннями та файлами

Функція createImages(frame, count) (Рис. 3.12):

- захоплює обличчя з кадру і зберігає його в папку даних;
- малює прямокутник навколо обличчя на кадрі.

```
def createImages(frame, count):
    global dataPath, name, Id
    if not name or not Id:
        return frame
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_frame)
    for (top, right, bottom, left) in face_locations:
        face_image = rgb_frame[top:bottom, left:right]
        pil_image = Image.fromarray(face_image)
        pil_image.save(os.path.join(dataPath, f"{name}.{Id}.{count}.jpg"))
        cv2.rectangle(frame, (left, top), (right, bottom), (255, 0, 0), 2)
    return frame
```

Рис. 3.12. Функція createImages

Функція writeDatabase(databaseFile, row) записує інформацію про користувача в CSV файл (Рис. 3.13).

```
def writeDatabase(databaseFile, row):
    if not os.path.exists(databaseFile):
        with open(databaseFile, 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(["Id", "Name", "Status"])
    with open(databaseFile, 'a+') as csvFile:
        writer = csv.writer(csvFile)
        writer.writerow(row)
    return "Збережено в базу даних"
```

Рис. 3.13. Функція writeDatabase

Функція getImagesAndLabels(path) завантажує зображення облич і їхні мітки (ID) із зазначеної папки (Рис. 3.14).

```
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    Ids = []
    for imagePath in imagePaths:
        extension = os.path.splitext(imagePath)[1]
        if extension != '.jpg':
            continue
        image = face_recognition.load_image_file(imagePath)
        face_encodings = face_recognition.face_encodings(image)
        if face_encodings:
            face_encoding = face_encodings[0]
            Id = int(os.path.split(imagePath)[-1].split(".")[1])
            faces.append(face_encoding)
            Ids.append(Id)
    return faces, Ids
```

Рис. 3.14. Функція getImagesAndLabels

Функція Train(path) тренує систему розпізнавання облич, створюючи масив облич та їхніх міток і зберігаючи їх у файл train_data.npy (Рис. 3.15).

```python
def Train(path):
    faces, Ids = getImagesAndLabels(path)
    data = {"faces": faces, "Ids": Ids}
    project_dir = os.path.dirname(os.path.abspath(__file__))
    save_path = os.path.join(project_dir, "train_data.npy")
    np.save(save_path, data)
    return "Навчання завершено..."
```

Рис. 3.15. Функція Train

Функція TrackImages(frame) розпізнає обличчя в кадрі, порівнюючи їх із навченими даними, і відображає інформацію про розпізнані обличчя на екрані (Рис. 3.16).

```python
def TrackImages(frame):
    project_dir = os.path.dirname(os.path.abspath(__file__))
    save_path = os.path.join(project_dir, "train_data.npy")
    if not os.path.exists(save_path):
        return frame, "Немає доступних даних для навчання"
    data = np.load(save_path, allow_pickle=True).item()
    known_face_encodings = data["faces"]
    known_face_ids = data["Ids"]
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)
    df = pd.read_csv(databaseFile, delimiter=',', encoding='cp1251')
    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            Id = known_face_ids[best_match_index]
            person = df.loc[df['Id'] == Id]['Name'].values[0]
            status = df.loc[df['Id'] == Id]['Status'].values[0]
            person_info = f"{Id}-{person}-{status} {int((1 - face_distances[best_match_index]) * 100)}%"
        else:
            person_info = "Невідомо"
        cv2.rectangle(frame, (left, top), (right, bottom), (255, 0, 0), 2)
        cv2.putText(frame, person_info, (left, bottom + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)
    return frame, "Розпізнавання завершено"
```

Рис. 3.16. Функція TrackImages

3.2.4 Створення графічного інтерфейсу користувача

Створення головного вікна програми за допомогою бібліотеки tkinter. Встановлюються стилі для кнопок і колір фону вікна (Рис. 3.17).

```
def main():
    global register, sampleNum, dataPath, name, Id, status, recognizeFrame
    root = tk.Tk()
    root.title("Система розпізнавання облич")
    root.configure(bg="#ccffcc")

    style = ttk.Style()
    style.configure("TButton", font=("Helvetica", 12), background="#007ACC", foreground="blue")
    style.map("TButton", background=[('active', '#005C99')], foreground=[('active', 'blue')])
```

Рис. 3.17. Створення головного вікна програми

Функції обробки подій натискання кнопок реєстрації, тренування і розпізнавання (Рис. 3.18).

```
def register_action():
    global register, count
    register = True
    count = 0

def train_action():
    info = Train(dataPath)
    messagebox.showinfo("Інформація", info)

def recognize_action():
    global recognizeFrame
    recognizeFrame = True
```

Рис. 3.18. Функції обробки подій натискання кнопок реєстрації, тренування і розпізнавання.

Створення панелей і віджетів для введення даних користувача, а також кнопок реєстрації, тренування і розпізнавання (Рис. 3.19).

```
left_panel = tk.Frame(root, bg="#ccffcc")
left_panel.grid(row=0, column=0, padx=10, pady=10)
title_label = ttk.Label(left_panel, text="Розпізнавання обличчя", font=("Helvetica", 20), background="#ccffcc")
title_label.grid(row=0, column=0)
image_label = ttk.Label(left_panel)
image_label.grid(row=1, column=0)

right_panel = tk.Frame(root, bg="#ccffcc", padx=10, pady=10)
right_panel.grid(row=0, column=1, padx=10, pady=10, sticky="n")
id_label = ttk.Label(right_panel, text="Ідентифікатор:", font=("Helvetica", 11), background="#ccffcc")
id_label.grid(row=0, column=0, pady=5,sticky="w")
id_entry = ttk.Entry(right_panel)
id_entry.grid(row=0, column=1, pady=5)
name_label = ttk.Label(right_panel, text="Ім'я:", font=("Helvetica", 11), background="#ccffcc")
name_label.grid(row=1, column=0, pady=5,sticky="w")
name_entry = ttk.Entry(right_panel)
name_entry.grid(row=1, column=1, pady=5)
status_label = ttk.Label(right_panel, text="Статус:", font=("Helvetica", 11), background="#ccffcc")
status_label.grid(row=2, column=0, pady=5,sticky="w")
status_entry = ttk.Entry(right_panel)
status_entry.grid(row=2, column=1, pady=5)

register_button = ttk.Button(right_panel, text="1. Зареєструватися", command=register_action)
register_button.grid(row=3, column=1, columnspan=2, pady=10, sticky="w")
train_button = ttk.Button(right_panel, text="2. Тренувати", command=train_action)
train_button.grid(row=4, column=1, columnspan=2, pady=10, sticky="w")
recognize_button = ttk.Button(right_panel, text="3. Розпізнати", command=recognize_action)
recognize_button.grid(row=5, column=1, columnspan=2, pady=10, sticky="w")
```

Рис. 3.19. Створення панелей, віджетів, кнопок.

3.2.5 Основний цикл оновлення кадру

Запускається основний цикл оновлення кадру. У ньому виконується читання кадру з камери, реєстрація облич, тренування моделі та розпізнавання облич (Рис. 3.20).

```
cap = cv2.VideoCapture(0)
def update_frame():
    global register, recognizeFrame, count, name, Id, status
    ret, frame = cap.read()
    if not ret:
        return
    name = name_entry.get()
    Id = id_entry.get()
    status = status_entry.get()
    if register:
        frame = createImages(frame, count)
        info = "Збереження " + str(count)
        count += 1
        if count > sampleNum:
            row = [Id, name, status]
            info = writeDatabase(databaseFile, row)
            register = False
            messagebox.showinfo("Інформація", info)
    if recognizeFrame:
        frame, info = TrackImages(frame)
        title_label.config(text=info)
    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(img)
    imgtk = ImageTk.PhotoImage(image=img)
    image_label.imgtk = imgtk
    image_label.configure(image=imgtk)
    root.after(20, update_frame)
root.after(0, update_frame)
root.mainloop()
cap.release()
cv2.destroyAllWindows()
```

Рис. 3.20. Основний цикл оновлення кадру.

3.3. Розпізнавання облич. Результат роботи програми

Програма запускається в три кроки за допомогою трьох кнопок у вікні:

1) **Зареєструватися**: це дозволить виявляти обличчя на зображеннях з веб-камери, зберігати зображення обличчя у файли, а також зберігати відповідну інформацію про особу, а саме, ідентифікатор, ім'я та статус у файлі database.csv.
2) **Тренувати**: це потрібно для тренування розпізнавача обличчя бібліотеки face_recognition. Бібліотека використовує файли зображень обличчя та ідентифікатори для створення моделі розпізнавання обличчя і зберігає навчену модель у файлі train_data.npy.
3) **Розпізнати**: це завантаження навченої моделі з файлу train_data.npy, розпізнавання обличчя та відображення відповідної інформації.

У результаті роботи програми отримано такий результат (Рис. 3.21 - Рис. 3.24).

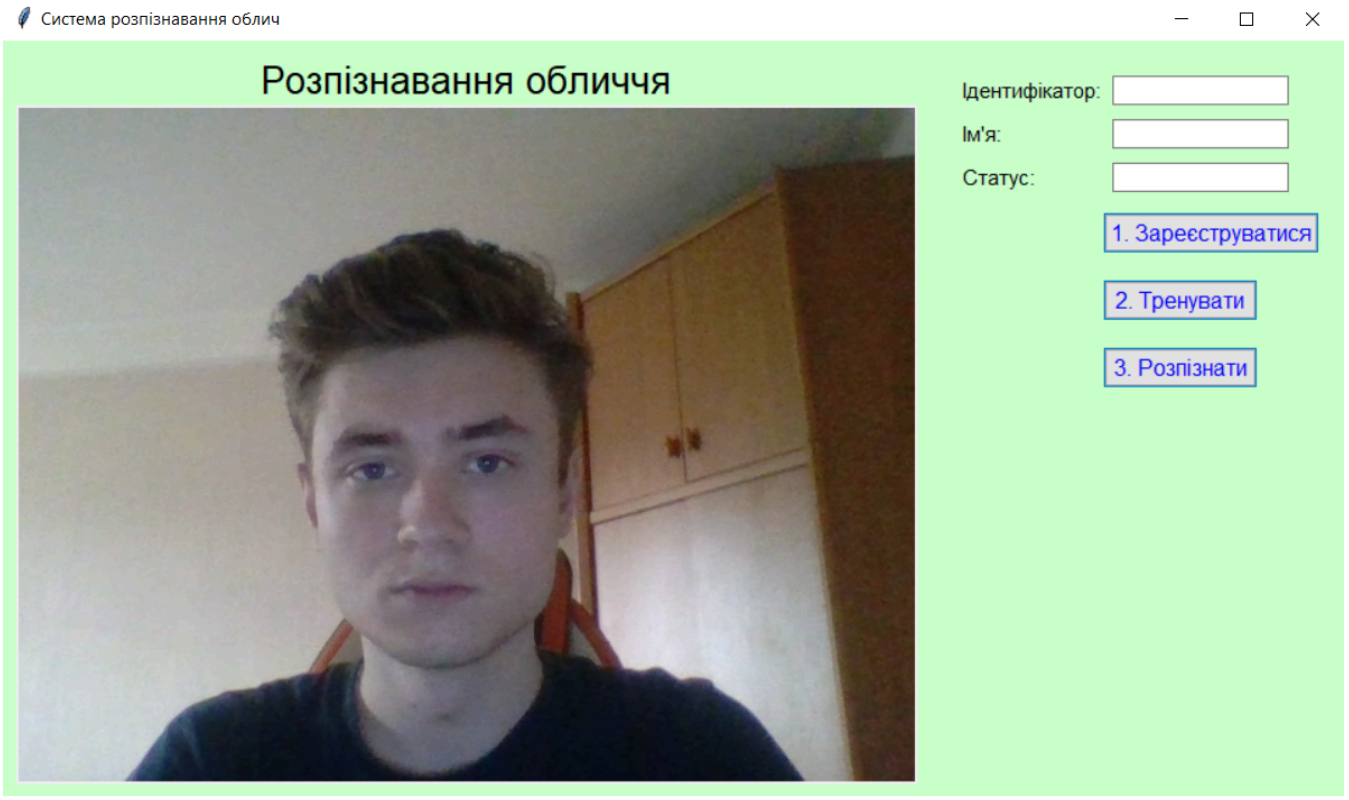


Рис. 3.21. Графічний інтерфейс користувача

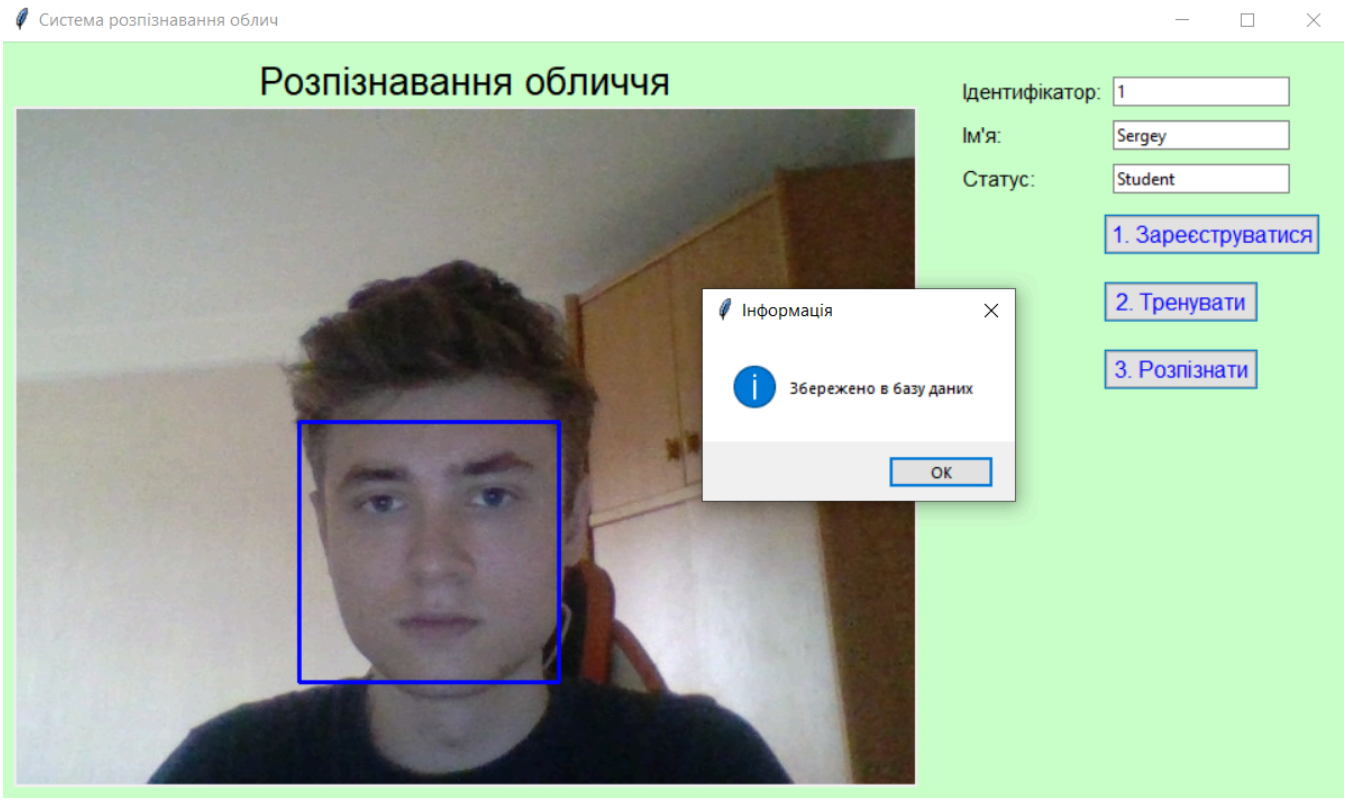


Рис. 3.22. Реєстрація користувача

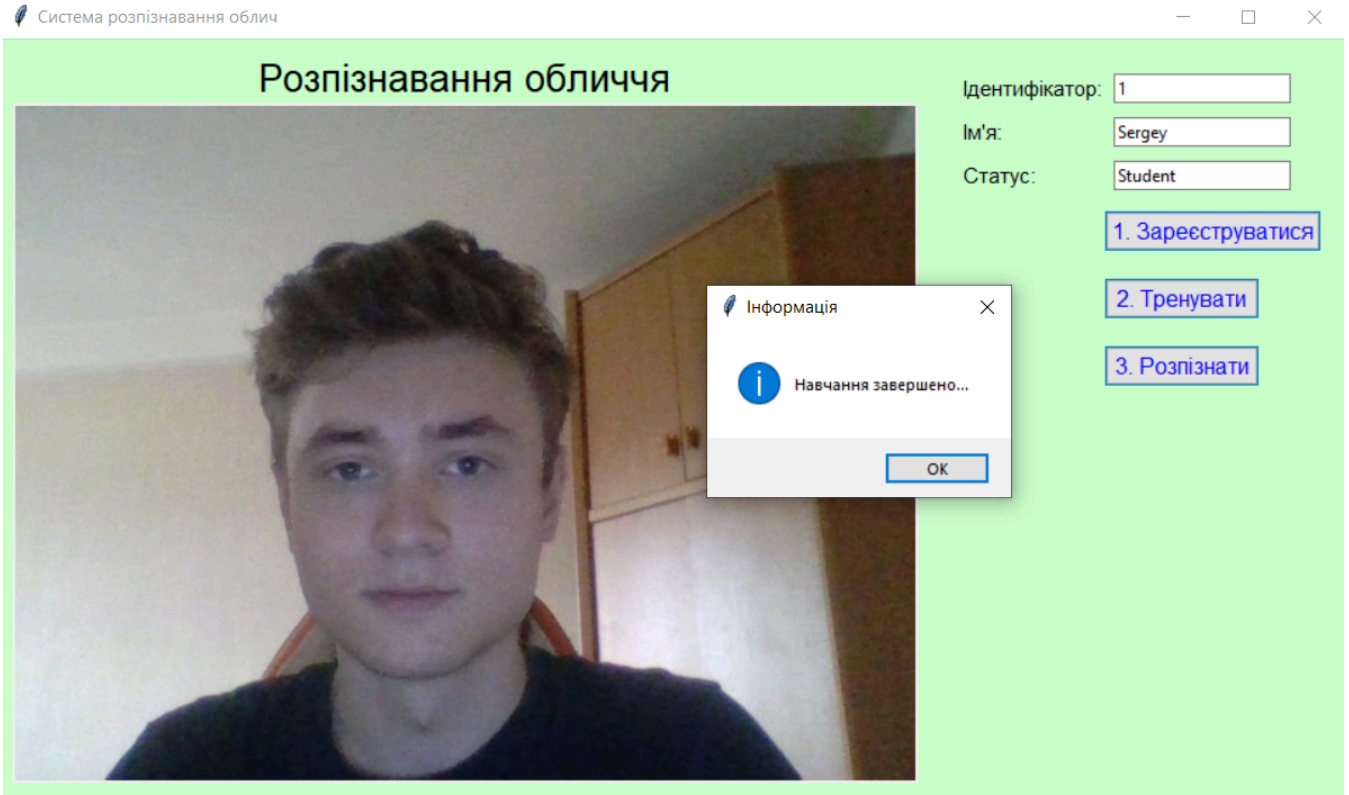


Рис. 3.23. Тренування розпізнавача обличчя

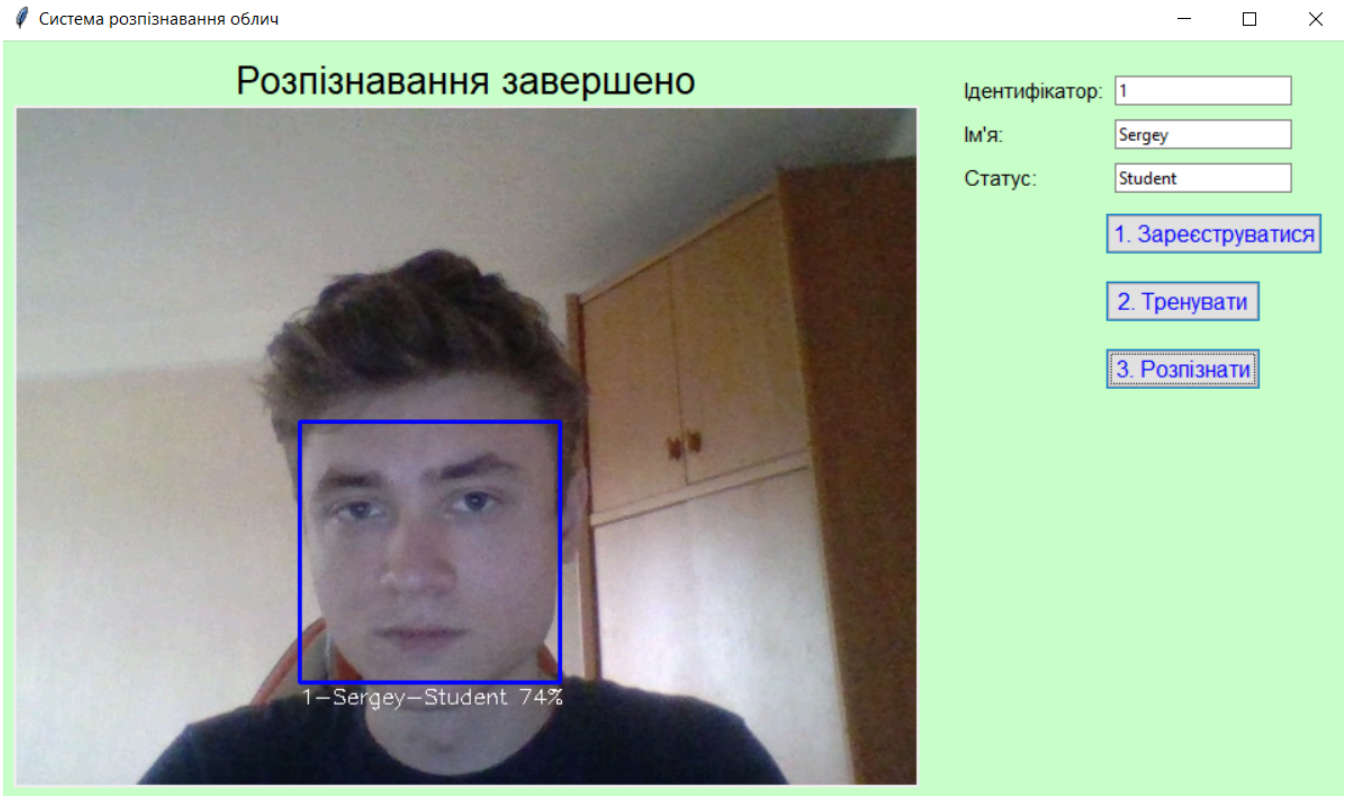


Рис. 3.24. Розпізнавання обличчя

4. ЕРГОНОМІКА СИСТЕМИ ВИЯВЛЕННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ

4.1 Ергономічні цілі програмного продукту

Ергономіка - це багатогранна наука, що досліджує взаємодію людини в робочих умовах з метою підвищення їхньої ефективності, безпеки та комфорту. Також вона являє собою сукупність міжнародних стандартів, що регулюють умови праці.

Ергономіка програмного забезпечення розглядає питання застосування ергономіки до програмних аспектів інтерактивних систем.

Комп'ютерні інформаційні системи розробляються для забезпечення роботи користувача, тобто для того, щоб він за допомогою таких систем швидше і якісніше вирішував свої виробничі завдання.

Принципи, рекомендації та вимоги ергономіки програмного забезпечення допомагають запобігти виникненню у користувачів проблем із придатністю використання програмного забезпечення, таких як:

- додаткові дії, не потрібні для виконання завдання;
- інформація, що вводить в оману;
- користувацькі інтерфейси з недостатньою інформацією;
- несподівана реакція інтерактивної системи;
- навігаційні обмеження під час використання системи;
- неефективне відновлення після помилок.

З погляду ергономіки, найважливіше в програмному забезпеченні - створити такий користувацький інтерфейс, який зробить роботу ефективною і продуктивною, а також забезпечить задоволеність користувача від роботи з програмою.

Ефективність роботи передбачає досягнення точності, функціональної повноти і завершеності під час виконання виробничих завдань користувачем на його робочому місці.

Створення програмного продукту має бути націлене на показники ефективності:

- точність роботи - визначається тим, якою мірою результат роботи користувача відповідає пред'явленим до нього вимогам. Показник точності включає відсоток помилок, яких припустився користувач: кількість помилок набору, кількість неправильних звернень до даних, запитів тощо;
- функціональна повнота - відображає ступінь використання первинних і оброблених даних, кількість пропущених технологічних операцій при виконанні поставленого користувачеві завдання;
- завершеність роботи - описує ступінь виконання виробничого завдання середнім користувачем за певний термін або період, частку необроблених заявок, а також кількість користувачів, які виконали завдання у фіксовані терміни.

4.2 Вимоги до програмного забезпечення та основні підходи до його проектування з точки зору користувача

Функціональні вимоги системи виявлення та розпізнавання облич (Рис. 4.1.):

- **реєстрація користувачів:** користувач повинен мати можливість зареєструватися, надавши своє ім'я, ідентифікатор і статус;
- **навчання моделі:** система має вміти навчати модель на основі завантажених зображень облич;
- **розпізнавання облич:** система має розпізнавати обличчя в реальному часі, використовуючи веб-камеру.

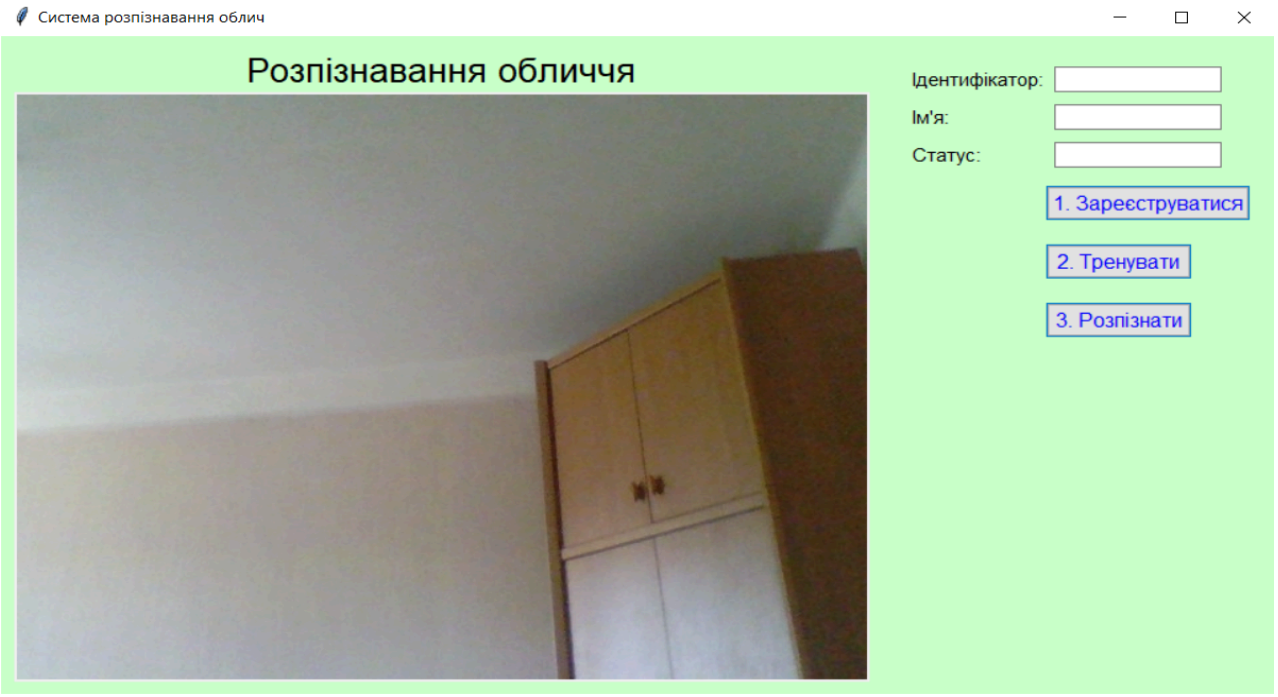


Рис. 4.1. Функціональні вимоги системи виявлення та розпізнавання облич

Нефункціональні вимоги:

- **продуктивність:** додаток має швидко реагувати на дії користувача;
- **надійність:** система має коректно опрацьовувати помилки та виняткові ситуації;
- **безпека:** дані користувачів мають бути захищені від несанкціонованого доступу.

Основні підходи до проектування:

- **модульність:** поділ функціоналу на окремі частини для спрощення розробки та супроводу;
- **інтерфейс користувача:** використання бібліотеки Tkinter для створення інтуїтивно зрозумілого інтерфейсу.

4.3 Аспекти які необхідно враховувати під час розроблення інтерфейсу користувача

Мета створення ергономічного інтерфейсу полягає в тому, щоб відобразити інформацію настільки ефективно, наскільки це можливо для людського сприйняття, і структурувати відображення на екрані таким чином, щоб привернути увагу до найважливіших одиниць інформації. Основна ж мета полягає в тому, щоб мінімізувати загальну інформацію на екрані та представити тільки те, що є необхідним для користувача.

Основні принципи створення інтерфейсу:

- **інтуїтивність**. Робота з системою не повинна викликати у користувача складнощів у пошуку необхідних елементів інтерфейсу для управління процесом вирішення поставленого завдання;
- **несуперечливість**. Якщо в процесі роботи з системою користувач використав деякі прийоми роботи з деякою частиною системи, то в іншій частині системи прийоми роботи мають бути ідентичні. Також робота із системою через інтерфейс має відповідати встановленим, звичним нормам;
- **не надмірність**. Користувач має вводити тільки мінімальну інформацію для роботи або управління системою;
- **гнучкість**. Для користувачів інтерфейс може бути організований як команди, комбінації натискань кнопок.

Необхідно враховувати розміщення інформації на екрані. Кількість інформації, що відображається на екрані, називається екранною щільністю. Якщо екранна щільність невелика, то відображувана інформація найбільш доступна та

зрозуміла для користувача, і навпаки, якщо екранна щільність велика, це може спричинити труднощі в засвоєнні інформації та її ясному розумінні.

Інформація на екрані може бути організована і структурована в значущі сегменти. Це може бути досягнуто з використанням кадрів (фреймів), методів на кшталт колірного кодування, рамок, або інших методів для привернення уваги.

Також необхідно враховувати, що дані на екрані слід розташовувати таким чином, щоб користувач знав, де знайти і де очікувати виведення необхідної інформації:

- інформація, на яку слід негайно звернути увагу, повинна завжди відображатися на видному місці, щоб захопити увагу користувача (наприклад, попереджувальні повідомлення та повідомлення про помилки);
- менш термінова, або менш необхідна інформація не повинна весь час перебувати перед користувачем, але має бути доступна, коли знадобиться.

Форми є основним елементом інтерфейсу. Їхнє призначення - забезпечити зручне введення та перегляд даних, стану та повідомлень автоматизованої системи.

Основні принципи проектування форм:

- форма проектується для зручнішого, зрозумілішого і якнайшвидшого досягнення розв'язання поставленого завдання;
- розміщення інформаційних одиниць на формі має відповідати логіці їх майбутнього використання; необхідно враховувати послідовність доступу до них, частоту їх використання та відносну важливість елементів;
- важливо використовувати порожній простір для створення балансу та симетрії серед інформаційних елементів форми, спрямовуючи увагу користувача в потрібне русло;
- логічні групи елементів слід відокремлювати пробілами, лініями, кольорами або іншими візуальними засобами;

- взаємопов'язані елементи повинні відображатися в одному вікні;
- для окремих полів заголовок має бути вирівняний по лівому краю; для полів списків, заголовок має бути вищим та лівішим по відношенню до основного поля, числові поля вирівнюються по правому полю;
- довгі стовпчикові поля, або довгі стовпчики інформаційних одиниць з поодинокими полями необхідно об'єднувати в групи, які розділяються порожнім рядком - це допомагає користувачеві подумки опрацьовувати інформацію за виділеними групами;
- у формах з великою кількістю інформації необхідно використовувати назви розділів, які однозначно свідчать про характер інформації, що їм належить;
- важливо чітко розділяти заголовки і поля введення, щоб уникнути плутанини і дискомфорту у користувача;
- заголовки мають бути стислими, знайомими та змістовними.

4.4 Основні ергономічні показники, що враховувалися під час розробки інтерфейсу користувача

Ергономіка програмного забезпечення є критично важливою складовою процесу розробки, оскільки вона сприяє створенню продукту, який є не лише функціональним, але й зручним та приємним у використанні.

Ергономічні показники системи виявлення та розпізнавання облич:

- **зручність використання:** інтерфейс простий та інтуїтивно зрозумілий;
- **простота навігації:** легкий доступ до всіх основних функцій;
- **консистентність:** єдиний стиль і форматування в усьому інтерфейсі;
- **інтерактивність:** оперативна реакція інтерфейсу на дії користувача;
- **адаптивність:** можливість використання на різних пристроях і екранах;
- **ефективність:** користувач швидко і легко досягає своїх цілей з мінімальними зусиллями.

Візуальний дизайн:

- **колірна палітра:** використання приємних для очей кольорів;
- **шрифти:** чіткі шрифти, які легко читаються;
- **розміщення елементів:** логічне та інтуїтивно зрозуміле розміщення кнопок, полів вводу та інших елементів управління.

Функціональність

- **підказки та повідомлення**: наявність інформативних повідомлень і підказок для користувача.
- **елементи управління:** зручні та чуйні елементи управління.

ВИСНОВКИ

Під час дослідження, у кваліфікаційній роботі було виконано такі завдання:

- проаналізовано існуючи методи та алгоритми для розпізнавання облич;
- досліджено задачу автоматичного виявлення обличчя;
- досліджено задачу автоматичного виявлення частин обличчя;
- досліджено задачу визначення характеристик людини за зображенням її обличчя;
- досліджено задачу аналізу виразів обличчя;
- досліджено задачу розпізнавання облич.

Апробацію результатів досліджень наведено у доданку А.

Було розроблено автоматизовану систему виявлення та розпізнавання облич. Основні результати роботи системи включають:

1) **реєстрація облич**:
   - **створення зображень облич**: під час реєстрації програма захоплює зображення облич із веб-камери і зберігає їх у вказану директорію. Ці зображення асоціюються з ім'ям та ідентифікатором, введеними користувачем;
   - **збереження даних у базі даних**: ідентифікатори, імена та статус зберігаються в CSV-файлі бази даних.
2) **навчання моделі**:
   - **збір даних для навчання**: програма витягує обличчя та їхні ідентифікатори зі збережених зображень;
   - **створення моделі розпізнавання**: витягнуті дані використовуються для створення моделі розпізнавання облич, яка потім зберігається у файл.
3) **розпізнавання облич**:
   - **виявлення облич на відео**: програма обробляє відеопотік із веб-камери, виявляє обличчя в кожному кадрі та порівнює їх з обличчями в навченій моделі;

- **порівняння облич**: для кожного виявленого обличчя обчислюється вектор ознак, який порівнюється з векторами, збереженими в навченій моделі;
- **ідентифікація облич**: якщо виявлена особа збігається з однією з відомих осіб у моделі, програма виводить відповідну інформацію (наприклад, ім'я і статус) поруч з особою на відео.

4) **візуалізація результатів**:

- **відображення розпізнаних облич**: програма відображає відео з веб-камери, на якому розпізнані обличчя обведені рамками, а поруч із ними зазначені ідентифікаційні дані (ім'я, статус і ступінь упевненості в розпізнаванні).

Список використаних джерел

1. Venetsanopoulos A. Face recognition using kernel direct discriminant analysis algorithms // IEEE Transactions On Neural Networks. – 2003. – Vol.14. – №1. – P.117–126.
2. Lawrence S., Giles C. L., Tsoiта A. C. Face Recognition: A Convolutional Neural Network Approach// IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition. – 1997. – Vol. 8. – №1. – P.98–113.
3. P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume: 19, Issue: 7, Jul 1997, pp. 711 – 720.
4. Cootes T. F., Edwards G. J., Taylor C. J. Active appearance models. // IEEE Trans. on Pattern Recognition and Machine Intelligence. – 2001. – №23 (6) – P. 681–685.
5. Haar A. Z. Theorie der orthogonalen Funktionensysteme. Mathematische Annalen / Published: in: C. Heil and D.F. Walnut (eds.)// Fundamental Papers in Wavelet Theory. – Princeton University Press, Princeton 2006. – P. 155-188.
6. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features [Electronic resource].- 2001. – Available at : http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-1.pdf.
7. Viola P. Rapid object detection using a boosted cascade of simple features // IEEE Conf. on Computer Vision and Pattern Recognition. – 2001. – V. 1. – Kauai, Hawaii, USA. – P. 511–518.
8. Viola P. Robust realtime face detection // International Journal of Computer Vision. – 2004. – V. 57. – № 2. – P. 137–154.
9. Lawrence S., Giles C.L., Tsoiта C. Face Recognition: A Convolutional Neural Network Approach // IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition. – 1997. – Vol. 8. – №1. – P.98–113.
10. Taigman Y., Yang M., Ranzato M. DeepFace: Closing the Gap to Human-Level Performance in Face Verification [Electronic resource] / [Yaniv Taigman, Ming

Yang, Marc'Aurelio Ranzato та ін] . – Available at : https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf.

11. Meng Joo Er ; Chen W.; Wu Shiqian High-speed face recognition based on discrete cosine transform and RBF neural networks // IEEE Transactions on Neural Networks. – 2005. –Vol. 16. – Issue: 3. – P. 679 – 691.

12. Rowley H.A.; Baluja S.; Kanade T. Rotation invariant neural network-based face detection // Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference. – 1998. – Available at : http://ieeexplore.ieee.org/abstract/document/698585/ http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

13. Mäenpää T. The local binary pattern approach to texture analysis – extensions and applications — Finland ,Oulu University Press, 2003. – 80 p.

14. Ahonen T.; Hadid A.; Pietikainen M. Face Description with Local Binary Patterns: Application to Face Recognition//IEEE Transactions on Pattern Analysis and Machine Intelligence. -2006. – Vol. 28. – Issue 12.- P. 2037 – 2041.

15. Lifshic Ju. Metody raspoznavanija lic // [Elektronnij resurs]. / Rezhim dostupu: http://yury.name/modern/08modernnote.pdf.

16. Mark Lutz, Learning Python, 5th Edition, O'Reilly, 2013 – 1643p.

17. Luciano Ramalho, Fluent Python, 1st Edition, O'Reilly, 2015 – 792p.
18. Al Sweigart, Automate the Boring Stuff with Python, 2nd Edition, No Starch Press, 2019 – 592p.
19. David Beazley, Brian K. Jones, Python Cookbook, 3rd Edition, O'Reilly, 2013 – 706p.
20. Eric Matthes, Python Crash Course, 2nd Edition, No Starch Press, 2019 – 544p.
21. Zed A. Shaw, Learn Python 3 the Hard Way, 1st Edition, Addison-Wesley, 2017 – 320p.
22. Allen B. Downey, Think Python, 2nd Edition, O'Reilly, 2015 – 292p.
23. Raymond Hettinger, Effective Python: 59 Specific Ways to Write Better Python, 1st Edition, Addison-Wesley, 2015 – 256p.
24. Paul Barry, Head First Python, 2nd Edition, O'Reilly, 2016 – 624p.

25. Bradski G. Learning OpenCV – O'Reilly / Kaehler A. – O'Reilly Media, 2008 – 580c.

26. Joseph Howse, Learning OpenCV 4 Computer Vision with Python, 3rd Edition, Packt Publishing, 2020 – 402p.

27. David Millán Escrivá, Vinícius G. Mendonça, Prateek Joshi, OpenCV 4 Computer Vision Application Programming Cookbook, 4th Edition, Packt Publishing, 2019 – 442p.

28. Michael Beyeler, Machine Learning for OpenCV: Intelligent Image Processing with Python, 1st Edition, Packt Publishing, 2017 – 382p.

29. Robert Laganière, OpenCV 3 Computer Vision Application Programming Cookbook, 2nd Edition, Packt Publishing, 2014 – 382p.

30. Sandipan Dey, Hands-On Image Processing with Python: Expert techniques for advanced image analysis and effective interpretation of image data, Packt Publishing, 2018 – 480p.

31. Adrian Rosebrock, Deep Learning for Computer Vision with Python: Master Deep Learning and Computer Vision in Python, 1st Edition, PyImageSearch, 2017 – 900p.

32. Matthew Wilkes, PIL Handbook, Python Imaging Library, Pythonware, 2010 – 112p.

33. Jan Erik Solem, Programming Computer Vision with Python: Tools and algorithms for analyzing images, O'Reilly Media, 2012 – 264p.

34. Travis E. Oliphant, Guide to NumPy, CreateSpace Independent Publishing Platform, 2006 – 378p.

35. Ivan Idris, NumPy Beginner's Guide, 3rd Edition, Packt Publishing, 2015 – 348p.

36. Wes McKinney, Python for Data Analysis, O'Reilly Media, 2017 – 544p.

37. Daniel Y. Chen, Pandas for Everyone: Python Data Analysis, Addison-Wesley Professional, 2018 – 544p.

38. Michael Heydt, Learning Pandas - Second Edition: High-performance data manipulation and analysis in Python, Packt Publishing, 2017 – 538p.

39. Matt Harrison, Pandas in Action, Manning Publications, 2020 – 424p.

40. Paul and Gail Grinstead, Tkinter GUI Application Development Blueprints: Master GUI programming in Tkinter as you design, implement, and deliver ten real-world applications from start to finish, Packt Publishing, 2018 – 422p.

41. Alan D. Moore, Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter, Packt Publishing, 2018 – 400p.

Додаток А

Апробація результатів

BMC-2023 – International Scientific-Practical Conference of young scientists "Build-Master-Class-2023" December 2023, Kyiv, Ukraine

Використання комп'ютерного зору в системах доповненої реальності

Максим Жураковський, студент[1]. Сергій Позднякович, студент[1]. Тетяна Гончаренко, к.т.н., доцент, в.о. завідувача кафедри ІТ[1]

[1] *Київський національний університет будівництва і архітектури, Київ, Україна*

АНОТАЦІЯ

Представлено результати аналізу взаємодії комп'ютерного зору із системами доповненої реальності. Виділені основні складові механік доповненої реальності, задачі комп'ютерного зору та галузі його використання.

Ключові слова: конференція з інформаційних технологій, комп'ютерний зір, доповнена реальність.

1. ВСТУП

Тематика доповненої реальності активно розвивається у наш час. Вона вже частково використовується під час будівництва, медицини, навчання, розваг, а також у розвитку штучного інтелекту. Одну з ключових ролей у створенні доповненої реальності грає комп'ютерний зір.

Взаємозв'язок двох згаданих технологій може у перспективі значно покращити стан багатьох сфер діяльності.

2. МЕТА РОБОТИ

Звернення уваги на комп'ютерний зір та технології доповненої реальності, а також їх взаємозв'язок у сфері інформаційних технологій.

3. КОМП'ЮТЕРНИЙ ЗОР

Комп'ютерний зір – інноваційна технологія, яка може виявити різні об'єкти, класифікувати їх та відстежувати їхні дії. Незважаючи на те, що основи вивчення даної технології були закладені наприкінці 20 століття, вона досі залишається актуальною та затребуваною.

Класичними завданнями комп'ютерного зору є: розпізнавання, відновлення зображення, руху, відновлення сцени.

Для першої задачі характерні розпізнавання конкретних об'єктів: людських осіб, геометричних фігур та різноманітних машин.

Суть завдання відновлення зображень є видалення різних шумів, розмитість об'єктів, що знаходяться в русі. Для виконання цього завдання використовуються фільтри низьких та середніх частот.

До завдання руху входить обробка відео для знаходження оцінки швидкості кожної точки зображення. З основних прикладів вирішення цього завдання є слідування за пересуванням різноманітних об'єктів, тобто тварин, людей, і навіть машин.

Для завдання відновлення сцени характерним є відтворення тривимірної моделі сцени, а в складних варіантах ще й повну тривимірну модель.[1]

На рис. 1 можна побачити різні сфери застосування комп'ютерного зору.

Однією з найважливіших сфер є медична галузь. Вона характеризується отриманням різної інформації з відео для постановки діагнозу, навчанням студентів людської анатомії тощо.[2]

Іншою галуззю, що потребує комп'ютерного зору, є військова сфера. Адже в ній він використовується для виявлення ворожих солдатів та техніки, керування сучасними ракетами.

Ще однією дуже важливою галуззю є промисловість. У ній визначається контроль якості продукції, зниження браку, і навіть підтримка різноманітного виробництва.[3]

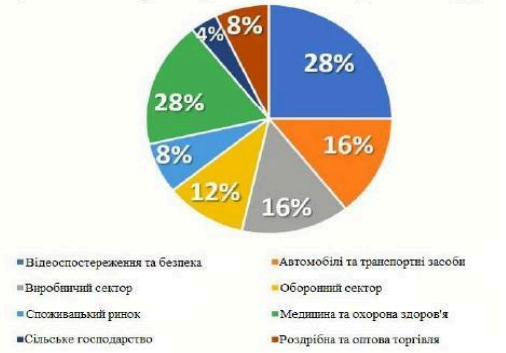


Рисунок 1. Найбільш перспективні індустрії комп'ютерного зору

4. ДОПОВНЕНА РЕАЛЬНІСТЬ

Доповнену реальність (augmented reality, AR) часто плутають з віртуальною, що, не дивлячись на схожі вихідні дані, не є вірним твердженням, адже всі віртуальні об'єкти, які ми бачимо в системах доповненої реальності, вводяться адаптивно в наш світ – це може бути схема нутрощів людського тіла, що стоїть перед вами або відновлені пам'ятники архітектури.

Доповнена реальність в більшості випадків створюється таким чином: спочатку проводиться зйомка фізичного об'єкту за допомогою оптичного сканеру (наприклад, камери телефону), після цього проходить ідентифікація, аналіз та обробка зображення через програмне забезпечення. Для віртуального об'єкту розраховується його місце розташування, а потім він об'єднується із нашою реальністю. Фінальний результат виводиться на дисплей пристрою (рис. 2).

Рисунок 2. Загальна схема створення доповненної реальності

Серед існуючих механік AR-технологій є різні прив'язки до місць-тригерів, наприклад до маркера або площини. У першому випадку віртуальні об'єкти з'являються при наведенні пристрою для зчитування на оригінал (втім, це може бути не тільки предмет, а й звук, логотип або картинка) (рис. 3), а в другому стають результатом попереднього сканування місцевості, щоб навіть при відведенні пристрою об'єкт займав виділений йому простір. Ще одна прив'язка до геолокації містить тригер на певних точках координат.

Рисунок 3. Приклад використання прив'язки до маркера

Механіка порталів дозволяє показувати графічні, фото- та відеоматеріали в режимі 360°, дає можливість поділитися контентом так, щоб і ви, і інші глядачі зможуть його побачити, поки стоять у різних точках простору.

AR-технології дозволяють накладати на існуючі фізичні об'єкти віртуальні доповнення. Для можливості такої взаємодії створюється копія об'єкта, що використовується в 3D-просторі.

На доповнену реальність можна не тільки дивитися – інтерактивні механіки відкривають можливості для натискання кнопок, ведення діалогів, переміщення у просторі та багато інших цікавих та зручних взаємодій, що максимально розкривають можливості сучасного стану технології.

Є й інші механіки, не такі основні, але не менш цікаві, на зразок масової участі в одній доповненій реальності або використання доповненої реальності в браузері.

На жаль, ця технологія все ще недоступна масовому користувачеві через проблеми з обробкою людським мозком інформації, що він отримує, громіздкістю пристроїв і, звичайно, їх вартістю, але технології все ще розвиваються і покращують свої показники з кожним роком.

5. ВЗАЄМОЗВ'ЯЗОК МІЖ КОМП'ЮТЕРНИМ ЗОРОМ ТА ДОПОВНЕНОЮ РЕАЛЬНІСТЮ

Системи доповненої реальності багатьом зобов'язані технології комп'ютерного зору, адже одна з головних гілок розвитку AR – це розпізнавання об'єктів у реальному часі. Зараз, наприклад, дана розробка допомагає сліпим і слабозорим дізнаватися, що відбувається навколо них, і які предмети можуть перешкодити їхньому руху. У майбутньому, з розвитком цієї галузі, можна буде визначати найдрібніші деталі оточення не тільки в полі зору користувача, а й повністю аналізувати архітектуру будівель, прокладати безпечні маршрути та багато іншого, що більше відповідає реалістичному 3D-скануванню з подальшим застосуванням отриманої інформації.

Сучасні системи доповненої реальності не завжди справляються з побудовою сітки навколишнього простору, але завдяки більшому впровадженню та розвитку технологій комп'ютерного зору, віртуальні об'єкти зможуть працювати не тільки на чистій площині, а й у «захараченому» просторі, вірно реагуючи як на вже присутні фізичні об'єкти, так і на їх динамічну зміну.

Звичайно, у майбутньому зв'язок між доповненою реальністю та комп'ютерним зором буде використовуватися в різних гібридних підходах, щоб запобігти можливим помилкам і вильотам при неможливості прорахувати навколишній простір; у психофізичних дослідженнях, як вивчення помітності затримок, залежності помилок при поворотах головою, зміні нахилу пристрою і його незапланованих рухах; зрештою, взаємозв'язок, що був описаний в цій роботі, може призвести до чогось більшого, наприклад, відчуття доповненої реальності через тактильність, нюх чи слух.[4]

6. ВИСНОВКИ

1) У даній роботі ми розглянули технології комп'ютерного зору і доповненої реальності, їх взаємозв'язок, і навіть виділили перспективні галузі для комп'ютерного зору.
2) Провели аналіз зв'язку між комп'ютерним зором та доповненою реальністю. Знайшли важливі аспекти розвитку цих галузей у майбутньому.
3) Розібрали як дві технології працюють і яку функцію виконують та виконуватимуть.

Список літератури

[1] А. А. Лукьяница, А. Г. Шишкин. Цифровая обработка видеоизображений. — М.: «Ай-Эс-Эс Пресс», 2009.

[2] Матеріал з вільного сайту «HoloAnatomy» [Електронний ресурс] Режим доступу: https://case.edu/holoanatomy/

[2] Матеріал з вільного сайту «TAdviser» [Електронний ресурс] Режим доступу: https://tadviser.com/index.php/Article:Computer_vision_(machine_vision)

[3] R. Azuma, A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, pp. 355—385, August 1997.

Додаток Б

Лістинг коду системи виявлення та розпізнавання облич

FaceRecognationSystem.py

```
import cv2
import numpy as np
import face_recognition
import tkinter as tk
from tkinter import ttk, messagebox
import os
from PIL import Image, ImageTk
import pandas as pd
import csv

dataPath = r"C:/Foto/FaceRecognitionSystem/data"
databaseFile = r"C:/Foto/FaceRecognitionSystem/database.csv"

register = False
recognizeFrame = False
sampleNum = 20
name = ""
Id = ""
status = ""

faces = []
Ids = []

def createImages(frame, count):
```

```
    global dataPath, name, Id
    if not name or not Id:
        return frame
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_frame)
    for (top, right, bottom, left) in face_locations:
        face_image = rgb_frame[top:bottom, left:right]
        pil_image = Image.fromarray(face_image)
        pil_image.save(os.path.join(dataPath, f"{name}.{Id}.{count}.jpg"))
        cv2.rectangle(frame, (left, top), (right, bottom), (255, 0, 0), 2)
    return frame

def writeDatabase(databaseFile, row):
    if not os.path.exists(databaseFile):
        with open(databaseFile, 'a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(["Id", "Name", "Status"])
    with open(databaseFile, 'a+') as csvFile:
        writer = csv.writer(csvFile)
        writer.writerow(row)
    return "Збережено в базу даних"

def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    Ids = []
    for imagePath in imagePaths:
        extension = os.path.splitext(imagePath)[1]
        if extension != '.jpg':
            continue
```

```
            image = face_recognition.load_image_file(imagePath)
            face_encodings = face_recognition.face_encodings(image)
            if face_encodings:
                face_encoding = face_encodings[0]
                Id = int(os.path.split(imagePath)[-1].split(".")[1])
                faces.append(face_encoding)
                Ids.append(Id)
        return faces, Ids

    def Train(path):
        faces, Ids = getImagesAndLabels(path)
        data = {"faces": faces, "Ids": Ids}
         project_dir = os.path.dirname(os.path.abspath(__file__))  # Абсолютний шлях до поточного файлу
        save_path = os.path.join(project_dir, "train_data.npy")
        np.save(save_path, data)
        return "Навчання завершено..."

    def TrackImages(frame):
        project_dir = os.path.dirname(os.path.abspath(__file__))
        save_path = os.path.join(project_dir, "train_data.npy")
        if not os.path.exists(save_path):
            return frame, "Немає доступних даних для навчання"
        data = np.load(save_path, allow_pickle=True).item()
        known_face_encodings = data["faces"]
        known_face_ids = data["Ids"]
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)
        df = pd.read_csv(databaseFile, delimiter=',', encoding='cp1251')
```

```
            for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
                matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
                face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    Id = known_face_ids[best_match_index]
                    person = df.loc[df['Id'] == Id]['Name'].values[0]
                    status = df.loc[df['Id'] == Id]['Status'].values[0]
                    person_info = f"{Id}-{person}-{status} {int((1 - face_distances[best_match_index]) * 100)}%"
                else:
                    person_info = "Невідомо"
                cv2.rectangle(frame, (left, top), (right, bottom), (255, 0, 0), 2)
                cv2.putText(frame, person_info, (left, bottom + 15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)
            return frame, "Розпізнавання завершено"
        count = 0
        def main():
            global register, sampleNum, dataPath, name, Id, status, recognizeFrame
            root = tk.Tk()
            root.title("Система розпізнавання облич")
            root.configure(bg="#ccffcc")  # Колір фону світло-салатовий

            style = ttk.Style()
            style.configure("TButton", font=("Helvetica", 12), background="#007ACC", foreground="blue")
```

```
style.map("TButton", background=[('active', '#005C99')], foreground=[('active', 'blue')])

        def register_action():
            global register, count
            register = True
            count = 0

        def train_action():
            info = Train(dataPath)
            messagebox.showinfo("Інформація", info)

        def recognize_action():
            global recognizeFrame
            recognizeFrame = True

        left_panel = tk.Frame(root, bg="#ccffcc")
        left_panel.grid(row=0, column=0, padx=10, pady=10)
                title_label = ttk.Label(left_panel, text="Розпізнавання обличчя", font=("Helvetica", 20), background="#ccffcc")
        title_label.grid(row=0, column=0)
        image_label = ttk.Label(left_panel)
        image_label.grid(row=1, column=0)

        right_panel = tk.Frame(root, bg="#ccffcc", padx=10, pady=10)
        right_panel.grid(row=0, column=1, padx=10, pady=10, sticky="n")
        id_label = ttk.Label(right_panel, text="Ідентифікатор:", font=("Helvetica", 11), background="#ccffcc")
        id_label.grid(row=0, column=0, pady=5,sticky="w")
        id_entry = ttk.Entry(right_panel)
```

```
id_entry.grid(row=0, column=1, pady=5)
    name_label = ttk.Label(right_panel, text="Ім'я:", font=("Helvetica", 11), background="#ccffcc")
name_label.grid(row=1, column=0, pady=5,sticky="w")
name_entry = ttk.Entry(right_panel)
name_entry.grid(row=1, column=1, pady=5)
  status_label = ttk.Label(right_panel, text="Статус:", font=("Helvetica", 11), background="#ccffcc")
status_label.grid(row=2, column=0, pady=5,sticky="w")
status_entry = ttk.Entry(right_panel)
status_entry.grid(row=2, column=1, pady=5)

    register_button = ttk.Button(right_panel, text="1. Зареєструватися", command=register_action)
register_button.grid(row=3, column=1, columnspan=2, pady=10, sticky="w")
        train_button = ttk.Button(right_panel, text="2. Тренувати", command=train_action)
train_button.grid(row=4, column=1, columnspan=2, pady=10, sticky="w")
    recognize_button = ttk.Button(right_panel, text="3. Розпізнати", command=recognize_action)
recognize_button.grid(row=5, column=1, columnspan=2, pady=10, sticky="w")

cap = cv2.VideoCapture(0)

def update_frame():
    global register, recognizeFrame, count, name, Id, status
    ret, frame = cap.read()
    if not ret:
        return
```

```
        name = name_entry.get()
        Id = id_entry.get()
        status = status_entry.get()

        if register:
            frame = createImages(frame, count)
            info = "Збереження " + str(count)
            count += 1
            if count > sampleNum:
                row = [Id, name, status]
                info = writeDatabase(databaseFile, row)
                register = False
                messagebox.showinfo("Інформація", info)

        if recognizeFrame:
            frame, info = TrackImages(frame)
            title_label.config(text=info)

        img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(img)
        imgtk = ImageTk.PhotoImage(image=img)
        image_label.imgtk = imgtk
        image_label.configure(image=imgtk)
        root.after(20, update_frame)
    root.after(0, update_frame)
    root.mainloop()
    cap.release()
    cv2.destroyAllWindows()

main()
```

Додаток В

Презентація

Інформаційна технологія візуального сприйняття для виявлення та розпізнавання облич

Виконав:
Позднякович С. О.
гр. КН-20-1

Керівник:
к.т.н., доцент Гончаренко Т.А.

Аналіз предметної області та постановка задачі

Виявлення і розпізнавання облич — це безконтактні методи ідентифікації людей. З розвитком обчислювальних потужностей та алгоритмів використання великих баз даних, ці технології набули великої популярності.

Актуальність теми дослідження

Інтерес до розпізнавання облич продовжує зростати через його значення для безпеки, верифікації та криміналістичної експертизи.

Популярні бібліотеки компьютерного зору, такі як OpenCV, Face_recognition, дозволяють ефективно використовувати різні алгоритми розпізнавання. Готові рішення, такі як DeepFace від Facebook і FaceAPI від Google, мають певні обмеження, що робить подальші дослідження та удосконалення в цій області актуальними.

Аналіз предметної області та постановка задачі

3

Метою досліджень в роботі є створення автоматизованої системи виявлення та розпізнавання облич.

Реалізація мети здійснюється вирішенням наступних основних завдань:

- аналіз існуючих методів та алгоритмів розпізнавання облич;

Дослідження задач:

- автоматичного виявлення обличчя (Face Detection);
- автоматичного виявлення частин обличчя (Automatic Facial Features Detection);
- визначення характеристик людини за зображенням її обличчя (Objectis Features Determination);
- аналізу виразів обличчя (Facial Expression Analysis);
- розпізнавання облич (Face Recognition).

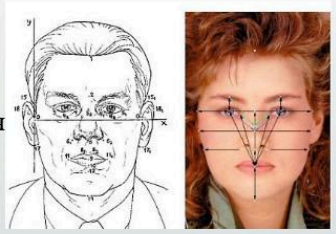
Об'єктом дослідження роботи є процес виявлення та розпізнавання облич.

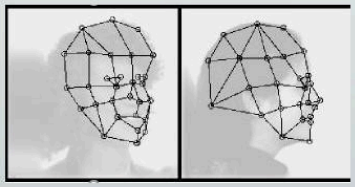
Предметом дослідження є методи виявлення та розпізнавання облич, за допомогою бібліотек OpenCV, PIL, DeepFace та Face_recognition мови програмування Python.

Огляд існуючих методів розпізнавання облич

4

- **Аналіз геометричних характеристик обличчя:**
  - виділяються вузлові точки (куточки очей, губ, кінчик носа);
  - прост в реалізації, але чутливий до освітлення та положення обличчя.
- **Метод гнучкого порівняння на графах:**
  - використовує графи для опису зображення обличчя;
  - міцний до змін освітлення та кута, але обчислювально складний.
- **Метод головних компонент (PCA):**
  - зменшує простір ознак для кращого порівняння зображень;
  - ефективний, але чутливий до освітлення та виразу обличчя.

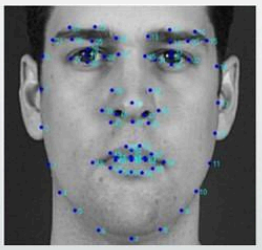




Огляд існуючих методів розпізнавання облич

5

- **Активні моделі зовнішнього вигляду (AAM):**
  - статистичні моделі, адаптовані до зображень через деформації;
  - потребує великого набору навчальних зображень.
- **Метод Віоли-Джонса:**
  - використовує вейвлети Хаара для виявлення об'єктів в реальному часі;
  - швидкий, але чутливий до кута обличчя.
- **Нейронні мережі (ЗНМ):**
  - найкращі результати у розпізнаванні, стійкі до змін масштабу, зсувів, поворотів.
  - потребує великої кількості даних для навчання, важко додавати нові еталони.



Завдання, пов'язані з аналізом зображень обличчя людини

6

Автоматичне виявлення обличчя людини

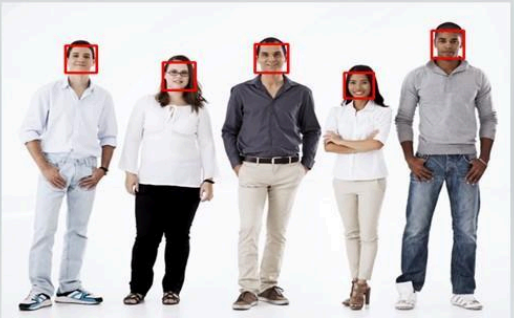
Бібліотека: OpenCV використовує каскадний класифікатор Хаара заснований на машинному навчанні.

Включає: набір вже навчених класифікаторів у вигляді XML-файлів.

Принцип роботи: використовує набір прямокутних областей (ознак Хаара) для виявлення деталей об'єкта на зображенні.

Застосування: виявлення об'єктів на зображенні та класифікація (об'єкт є / об'єкта немає).

```
import cv2
img = cv2.imread(r'C:/Foto/face1.jpg')
cv2.imshow('Input photo', img)
classifier = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
gray =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
boxes= classifier.detectMultiScale(gray,  scaleFactor=1.1, minNeighbors=5,  minSize=(30, 30))

for (x,y,w,h) in boxes:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

cv2.imshow('Face detection', img)
cv2.imwrite(r'C:/Foto/face1_detection.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Завдання, пов'язані з аналізом зображень обличчя людини

7

Автоматичне виявлення обличчя людини бібліотека Face_recognition

Переваги:

- проста у використанні;
- інтеграція з OpenCV та PIL.

Функції:

- виявлення облич на зображеннях та відео;
- ідентифікація ключових точок на обличчі.

```
from PIL import Image
import face_recognition

img = face_recognition.load_image_file(r'C:/Foto/face1_2.jpg')
face_locations = face_recognition.face_locations(img)

count = 0
for face_location in face_locations:
    count += 1
    top, right, bottom, left = face_location
    print("Face {} Top: {}, Left: {}, Bottom: {}, Right: {}".format(count, top, left, bottom, right))

    face_image = img[top:bottom, left:right]
    pil_image = Image.fromarray(face_image)
    pil_image.show()
```

```
Face 1 Top: 20, Left: 139, Bottom: 128, Right: 247
Face 2 Top: 86, Left: 315, Bottom: 176, Right: 404
```

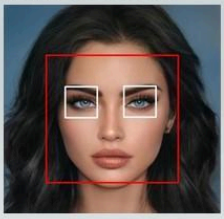
Завдання, пов'язані з аналізом зображень обличчя людини

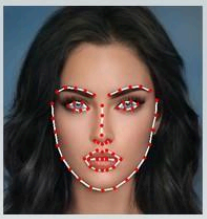
8

Автоматичне виявлення частин обличчя

На зображенні можна розпізнати не тільки обличчя, а й різні елементи на обличчях, наприклад, очі. Для вирішення таких завдань у бібліотеці OpenCV є вже готові каскади Хаара.

За допомогою бібліотеки Face_recognition можна не тільки виявляти обличчя, але й ідентифікувати частини обличчя.





```
import cv2
import face_recognition
img = cv2.imread(r'C:/Foto/face2.jpg')
cv2.imshow('Input photo', img)
rgb = img[:, :, ::-1]
landmarks = face_recognition.face_landmarks(rgb)
count = 0
for landmark in landmarks:
    for part in landmark.keys():
        print(part)
        a = landmark[part]
        for idx, item in enumerate(a):
            if idx == len(a) -1:
                break
            cv2.line(img, item, a[idx + 1], [255, 255, 255], 2)
        for point in landmark[part]:
            img = cv2.circle(img, point, 2, (0, 0, 255), 2)
            count = count +1
            print(point)
print(count)
cv2.imshow('Face Landmarks', img)
cv2.imwrite(r'C:/Foto/face2_landmarks.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Завдання, пов'язані з аналізом зображень обличчя людини

9

Визначення характеристик обличчя з DeepFace

Можливості: визначення віку, статі, раси

Підтримка TensorFlow та Keras.

```
import cv2
from deepface import DeepFace

foto=r'C:/Foto/face3.jpg'
img_show = cv2.imread(foto)
img = DeepFace.analyze(foto)[0]

print("Age: ", int(img["age"]))
print("Gender: ", max(img["gender"],key=img["gender"].get))
print("Race: ", img["dominant_race"])
cv2.imshow('Input photo', img_show)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Action: race: 100%|████████████████████████████████████████| 4/4 [00:50<00:00, 12.58s/it]
Age:  26
Gender:  Woman
Race:  asian
```

Завдання, пов'язані з аналізом зображень обличчя людини

10

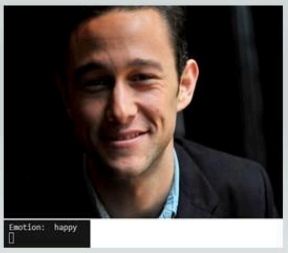
Аналіз виразів обличчя

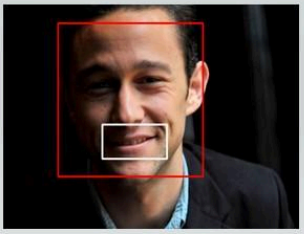
Методи:

- використання haarcascade_smile.xml - класифікатора Хаара бібліотекі OpenCV для виявлення посмішки;
- використання DeepFace для аналізу емоцій.

```
import cv2
from deepface import DeepFace

foto=r'C:/Foto/face3.jpg'
img_show = cv2.imread(foto)
img = DeepFace.analyze(foto,['emotion'])[0]

print("Emotion: ", img["dominant_emotion"])
cv2.imshow('Input photo', img_show)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
import cv2
img = cv2.imread(r'C:/Foto/face4.jpg')
cv2.imshow('Input photo', img)
classifier_face = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
classifier_smile = cv2.CascadeClassifier(cv2.data.haarcascades +'haarcascade_smile.xml')
gray =cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
boxes_faces = classifier_face.detectMultiScale(gray, scaleFactor=1.1,  minNeighbors=5, minSize=(30, 30))

for (x,y,w,h) in boxes_faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    face_gray = gray[y:y+h, x:x+w]
    face_color = img[y:y+h, x:x+w]
    boxes_smile = classifier_smile.detectMultiScale(face_gray, scaleFactor= 1.7, minNeighbors=5, minSize=(10, 10))
    for (xx, yy, ww, hh) in boxes_smile:
        cv2.rectangle(face_color, (xx, yy), (xx + ww, yy + hh), (255, 255, 255), 2)

cv2.imshow('Faces and smile detections', img)
cv2.imwrite(r'C:/Foto/face4_smile_detection.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





Завдання, пов'язані з аналізом зображень обличчя людини

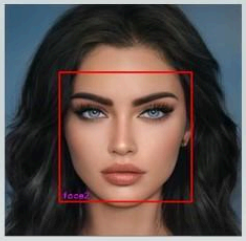
11

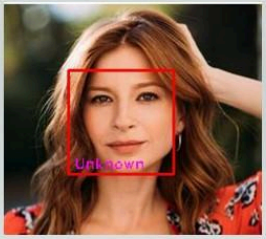
Розпізнавання облич

Процес розпізнавання облич включає:

- виявлення обличчя;
- вирівнювання обличчя;
- виділення ознак обличчя;
- порівняння обличчя за ознаками.

Ці етапи формують основу сучасних систем розпізнавання облич.





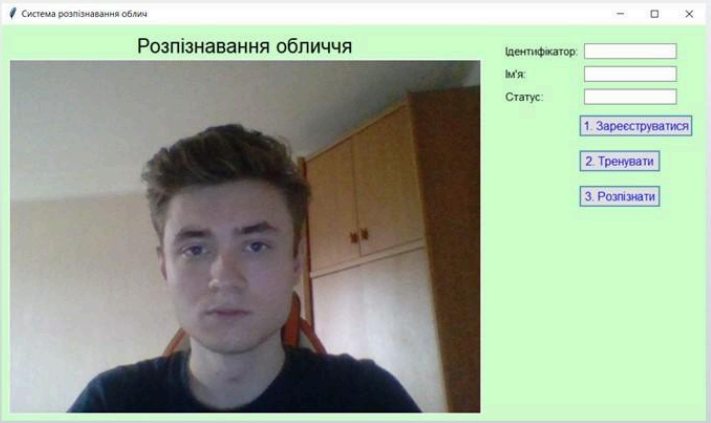

```python
import face_recognition, os, cv2
def recognition(directory):
    known_face_encodings = []
    known_face_names = []
    for f in os.listdir(directory):
        if f.endswith(".jpg") or f.endswith(".jpeg"):
            image_path = os.path.join(directory, f)
            face_image = face_recognition.load_image_file(image_path)
            face_encoding = face_recognition.face_encodings(face_image)[0]
            name = os.path.splitext(f)[0].split('.')[0]
            known_face_encodings.append(face_encoding)
            known_face_names.append(name)
    return known_face_encodings, known_face_names
known_face_encodings, known_face_names = recognition("C:/Foto/Faces")
test_img = face_recognition.load_image_file("C:/Foto/Faces/Test/face2.jpg")
face_locations = face_recognition.face_locations(test_img)
face_encodings = face_recognition.face_encodings(test_img, face_locations)
face_names = []
for face_encoding in face_encodings:
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = "Unknown"
    if True in matches:
        first_match_index = matches.index(True)
        name = known_face_names[first_match_index]
    face_names.append(name)
for (top, right, bottom, left), name in zip(face_locations, face_names):
    cv2.rectangle(test_img, (left, top), (right, bottom), (255, 0, 0), 2)
    cv2.putText(test_img, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 0, 255), 1)
cv2.imshow('Test Image', cv2.cvtColor(test_img, cv2.COLOR_RGB2BGR))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Система виявлення та розпізнавання облич

12

Програма запускається в три кроки за допомогою трьох кнопок у вікні:

1) **Зареєструватися**.
2) **Тренувати**.
3) **Розпізнати**.



13

Система виявлення та розпізнавання облич

Основні результати роботи системи включають:

1) **реєстрація облич**:

- **створення зображень облич**: під час реєстрації програма захоплює зображення облич із веб-камери і зберігає їх у вказану директорію. Ці зображення асоціюються з ім'ям та ідентифікатором, введеними користувачем;
- **збереження даних у базі даних**: ідентифікатори, імена та статус зберігаються в CSV-файлі бази даних.

2) **навчання моделі**:

- **збір даних для навчання**: програма витягує обличчя та їхні ідентифікатори зі збережених зображень;
- **створення моделі розпізнавання**: витягнуті дані використовуються для створення моделі розпізнавання облич, яка потім зберігається у файл.

14

Система виявлення та розпізнавання облич

Основні результати роботи системи включають:

3) **розпізнавання облич**:

- **виявлення облич на відео**: програма обробляє відеопотік із веб-камери, виявляє обличчя в кожному кадрі та порівнює їх з обличчями в навченій моделі;
- **порівняння облич**: для кожного виявленого обличчя обчислюється вектор ознак, який порівнюється з векторами, збереженими в навченій моделі;
- **ідентифікація облич**: якщо виявлена особа збігається з однією з відомих осіб у моделі, програма виводить відповідну інформацію (наприклад, ім'я і статус) поруч з особою на відео.

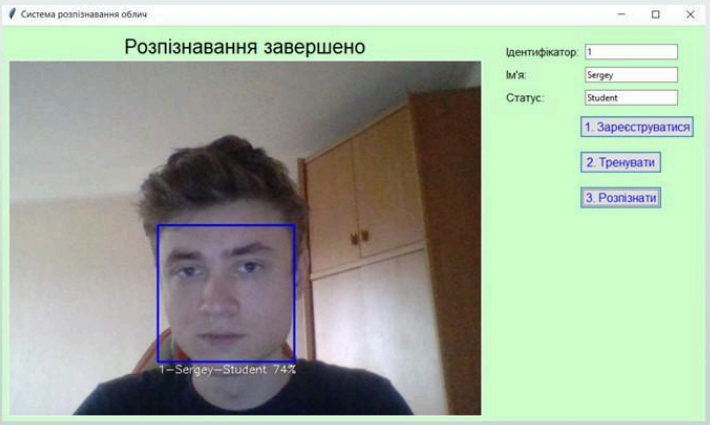
15

Система виявлення та розпізнавання облич

Основні результати роботи системи включають:

4) візуалізація результатів:

- **відображення розпізнаних облич**: програма відображає відео з веб-камери, на якому розпізнані обличчя обведені рамками, а поруч із ними зазначені ідентифікаційні дані (ім'я, статус і ступінь упевненості в розпізнаванні).



16

Ергономіка системи виявлення та розпізнавання облич

Основні ергономічні показники, що враховувалися під час розробки інтерфейсу користувача

- **зручність використання:** інтерфейс простий та інтуїтивно зрозумілий;
- **простота навігації:** легкий доступ до всіх основних функцій;
- **консистентність:** єдиний стиль і форматування в усьому інтерфейсі;
- **інтерактивність:** оперативна реакція інтерфейсу на дії користувача;
- **адаптивність:** можливість використання на різних пристроях і екранах;
- **ефективність:** користувач швидко і легко досягає своїх цілей з мінімальними зусиллями.

17

Ергономіка системи виявлення та розпізнавання облич

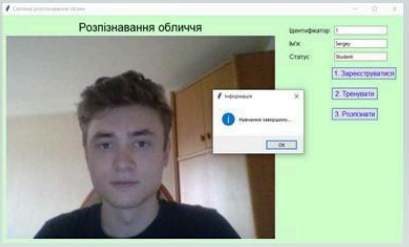
Основні ергономічні показники, що враховувалися під час розробки інтерфейсу користувача

Візуальний дизайн:

- **колірна палітра:** використання приємних для очей кольорів;
- **шрифти:** чіткі шрифти, які легко читаються;
- **розміщення елементів:** логічне та інтуїтивно зрозуміле розміщення кнопок, полів вводу та інших елементів управління.

Функціональність:

- **підказки та повідомлення**: наявність інформативних повідомлень і підказок для користувача.
- **елементи управління:** зручні та чуйні елементи управління.



18

ВИСНОВКИ

Під час дослідження, у кваліфікаційній роботі було виконано такі завдання:

- проаналізовано існуючи методи та алгоритми для розпізнавання облич;

Досліджено задачу:

- автоматичного виявлення обличчя;
- автоматичного виявлення частин обличчя;
- визначення характеристик людини за зображенням її обличчя;
- аналізу виразів обличчя;
- розпізнавання облич.

Розроблено автоматизовану систему виявлення та розпізнавання облич.