НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**УДК** 004.932.72'1

| **ПОГОДЖЕНО** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** |
|---|---|
| Декан факультету | Завідувач кафедри |
| Інформаційних технологій | Комп'ютерних систем, мереж та кібербезпеки |
| _____ Глазунова О.Г., д.пед.н, проф. | _____ Лахно В.А., д.т.н., проф. |
| підпис ПІБ, вчене звання і ступінь | підпис ПІБ, вчене звання і ступінь |
| «___»_____2021 р. | «___»_____2021 р. |

# МАГІСТЕРСЬКА РОБОТА

На тему: «Дослідження системи розпізнавання осіб з використанням комп'ютерного зору на базі IP камери»

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма _____

Орієнтація освітньої програми _____

Керівник магістерської роботи: _____ / Лахно В.А. /
підпис ПІБ

Виконав: _____ / Сміян І. А. /
підпис ПІБ

**КИЇВ-2021**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**«ЗАТВЕРДЖУЮ»**
**завідувач кафедри**
комп'ютерних систем, мереж та кібербезпеки
_____ / Лахно В.А., д.т.н., проф. /
підпис ПІБ, вчене звання і ступінь
«__» _____ 20__ р.

# З А В Д А Н Н Я

## ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Сміяну Іллі Андрійовичу
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): комп'ютерна інженерія
Освітня програма: комп'ютерні системи і мережі
Орієнтація освітньої програми: _____

Тема магістерської роботи: «Дослідження системи розпізнавання осіб з використанням комп'ютерного зору на базі IP камери»

Затверджена наказом ректора НУБіП України від "23" 10 2020 р. № 1578С
Термін подання завершеної роботи на кафедру _____
Вихідні дані до магістерської роботи _____

Перелік питань, що підлягають дослідженню:
1. Аналітичний огляд
2. Проектування
3. Реалізація та розробка компонентів системи

Перелік графічного матеріалу (за потреби) _____

**Дата видачі завдання** "23" 10 **2020 р.**

**Керівник магістерської роботи** _____ Лахно В.А., д.т.н., проф.
(підпис) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ Сміян І. А.
(підпис) (прізвище та ініціали студента)

**КАЛЕНДАРНИЙ ПЛАН**

| № з/п | Назва етапів магістерської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | | Виконано |
| 2 | Проектування системи | | Виконано |
| 3 | Реалізація системи | | Виконано |
| 4 | Тестування системи | | Виконано |
| 5 | Оформлення пояснювальної записки | | Виконано |
| 6 | Оформлення графічного матеріалу | | Виконано |

**Студент** _____ _____

(підпис) (ініціали та прізвище)

**Керівник проекту (роботи)** _____ _____

**РЕФЕРАТ**

Пояснювальна записка: 67 сторінок, 22 рисунків, 2 таблиці, 2 додатки, 10 джерел.

КОМП'ЮТЕРНА СИСТЕМА, КОМП'ЮТЕРНИЙ ЗІР ДОСТУПУ, ПРОЕКТУВАННЯ, BACK-END, FRONT-END, HTTP

Предметом дослідження є методи розпізнавання обличь за допомогою алгоритмів комп'ютерного зору.

Об'єктом дослідження виступає методи обробки та робота з зображення в реальному часі.

Мета магістерської роботи полягає в проектуванні та дослідженні системи розпізнавання осіб з використанням комп'ютерного зору яка приймає відеопотік з камери, навчається, ідентифікує та розпізнає обличчя.

Завданнями магістерської роботи є прискорення процесу розпізнавання облич у вхідному відеопотоці.

Апаратні та програмні засоби, що використовувались при проектуванні: python, Django, JetBrains PYcharm, draw.io, Microsoft Visio.

Результати досягнуті в процесі роботи – було розроблено систему розпізнавання осіб з використанням комп'ютерного зору та проведено тестування створеної системи, яке показало, що дана система є повністю працездатною та виконує усі вказані функції.

Одержані результати можуть бути використані у усіх підприємствах, адже дана система є універсально.

**ЗМІСТ**

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

Django – високорівневий відкритий Python-фреймворк для розробки веб-систем.

HTTP – протокол передачі гіпер-текстових документів, Hyper Text Transfer Protocol

API – програмний інтерфейс застосунку, Application Programing Interface

URL – уніфікований локатор ресурсів, Uniform Resource Locator

IDE – інтегроване середовище розробки, Integrated Development Environment

UML – уніфікована мова моделювання, Unified Modeling Language

БД – база даних

IP-адреса – це ідентифікатор (унікальний числовий номер) мережевого рівня, який використовується для адресації комп'ютерів чи пристроїв у мережах, Internet Protocol.

OpenCV – бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом, Open Computer Vision.

Бібліотека – у програмуванні збірник підпрограм або об'єктів, які використовуються для розробки програмного забезпечення.

# ВСТУП

Протягом багатьох десятиліть люди мріяли створити машини з характеристиками людського інтелекту, такі, які можуть мислити і діяти як люди. Однією з найбільш захоплюючих ідей було надати комп’ютерам можливість «бачити» та інтерпретувати навколишній світ. Вигадка вчорашнього дня стала фактом сьогодення.

Завдяки досягненню штучного інтелекту та обчислювальної потужності, технології комп’ютерного зору зробили величезний стрибок до інтеграції в наше повсякденне життя.

Використання систем для розпізнавання образів є одним із основоположних моментів промислової стратегії розвитку для більшості підприємств світу. Про це свідчить експоненційне зростання ринку споживання програмних рішень, що використовують технології розпізнавання зображень, що спостерігається останніми роками.

Можливості області, у яких можуть застосовуватися системи комп'ютерного зору, тривалий час отримували розвиток, переважно, як машинний аналог людських очей. Основною функцією та призначенням систем комп'ютерного зору є розпізнавання зовнішнього вигляду та розташування у просторі спостережуваних об'єктів, а також, перетворення оцифрування зображень, для подальшої обробки на персональному комп'ютері та передачі.

# 1 АНАЛІТИЧНИЙ ОГЛЯД

## 1.1 Комп'ютерний зір

Комп’ютерний зір — це область інформатики, яка зосереджується на створенні цифрових систем, які можуть обробляти, аналізувати й розуміти візуальні дані (зображення чи відео) так само, як це роблять люди.

Концепція комп’ютерного зору заснована на навчанні комп’ютерів обробляти зображення на рівні пікселів і розуміти його. Технічно машини намагаються отримати візуальну інформацію, обробляти її та інтерпретувати результати за допомогою спеціальних програмних алгоритмів.

Донедавна комп’ютерний зір працював лише в обмежених можливостях.

Завдяки досягненням у галузі штучного інтелекту та інноваціям у глибокому навчанні та нейронних мережах, ця галузь змогла зробити великі стрибки за останні роки та змогла перевершити людей у деяких завданнях, пов’язаних із виявленням та маркуванням об’єктів.

Одним із рушійних факторів зростання комп’ютерного зору є кількість даних, які ми генеруємо сьогодні, які потім використовуються для навчання та покращення комп’ютерного зору.

Поряд із величезною кількістю візуальних даних (понад 3 мільярди зображень щодня публікуються в Інтернеті) тепер доступні обчислювальні потужності, необхідні для аналізу даних. Оскільки область комп’ютерного зору розширюється з новим обладнанням та алгоритмами, точність ідентифікації об’єктів зростає. Менш ніж за десятиліття сучасні системи досягли 99% точності з 50%, що робить їх точнішими, ніж люди, у швидкому реагуванні на візуальні дані.

Ранні експерименти з комп’ютерним зором почалися в 1950-х роках, і вперше його почали використовувати в комерційних цілях для розрізнення набраного та рукописного тексту в 1970-х роках, сьогодні застосування комп’ютерного зору зросло в геометричній прогресії.

## 1.2 Як працює комп'ютерний зір

Одне з головних відкритих питань: як саме працює наш мозок? Реальність така, що існує дуже мало робочих і всеосяжних теорій обчислень мозку, тому, незважаючи на те, що нейронні мережі мають «імітувати роботу мозку», ніхто не впевнений, чи це насправді так.

Той самий парадокс справедливий і для комп'ютерного зору — оскільки ми не вирішили, як мозок і очі обробляють зображення, важко сказати, наскільки добре алгоритми, які використовуються у виробництві, схожі на наші власні внутрішні розумові процеси.

На певному рівні комп'ютерний зір – це все про розпізнавання образів. Таким чином, один із способів навчити комп'ютер розуміти візуальні дані – передати йому зображення, тисячі, мільйони зображень, якщо можливо, позначені, а потім піддати їх різним програмним методам або алгоритмам, які дозволяють комп'ютеру знаходити візерунки у всіх елементах.

Так, наприклад, якщо ви нагодуєте комп'ютеру мільйони зображень котів, він піддасть їх усіх алгоритмам, які дозволять їм аналізувати кольори на фотографії, форми, відстані між фігурами, де об'єкти межують один з одним, щоб визначити, що означає «кіт». Коли це буде завершено, комп'ютер (теоретично) зможе використовувати свій досвід, якщо йому дадуть інші зображення без міток, щоб знайти ті, які є кішкою.

Нижче наведена ілюстрація рис. 1.1. буфера зображення у відтінках сірого, в якому зберігається наше зображення. Яскравість кожного пікселя представлена одним 8-бітовим числом, діапазон якого становить від 0 (чорний) до 255 (білий).

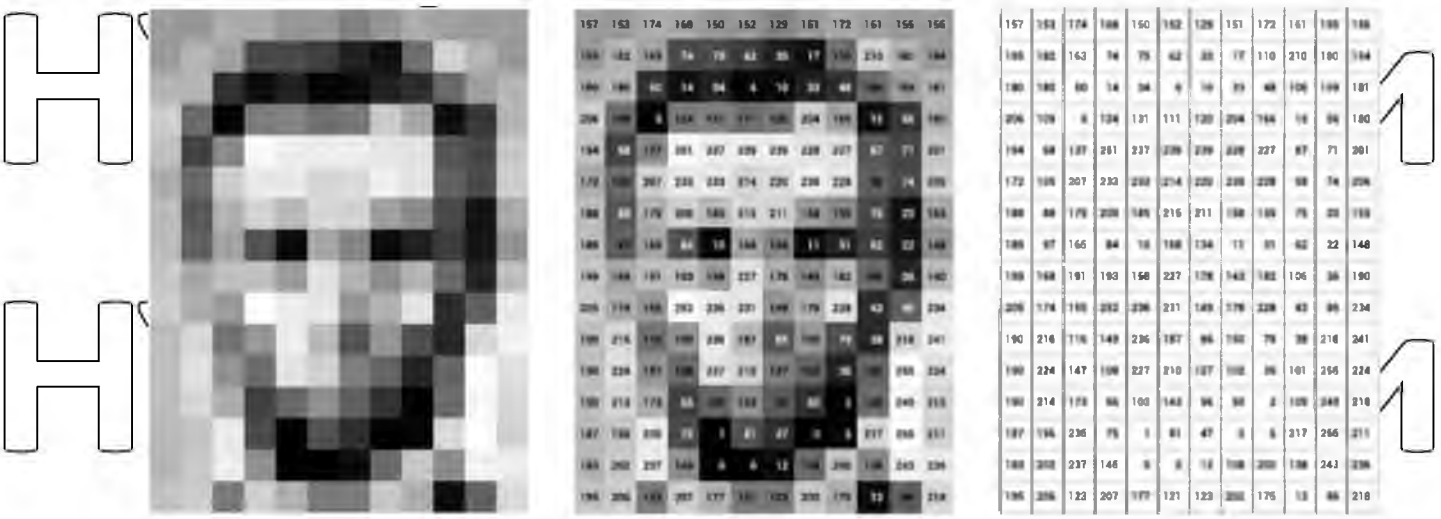


Рис. 1.1. Зображення у відтінках сірого

Такий спосіб зберігання даних зображення може суперечити вашим очікуванням, оскільки дані, безумовно, здаються двовимірними, коли вони відображаються. Однак це так, оскільки пам’ять комп’ютера складається просто з постійно зростаючого лінійного списку адресних просторів, приклад на рис. 1.2.


Рис. 1.2. Приклад збереження даних

Уявіть, що використовується кольорове зображення рис.1.3. Тепер все починає ускладнюватися.

Комп'ютери зазвичай зчитують колір як серію з 3 значень — червоного, зеленого та синього (RGB) — у тій самій шкалі 0–255. Тепер кожен піксель фактично має 3 значення, які комп'ютер зберігає на додаток до свого положення. Якби ми розфарбували рис.1.3., це призвело б до значень 12 x 16 x 3 або 576 чисел.

Рис. 1.3. Кольорове зображення

Для одного зображення потрібно багато пам'яті та багато пікселів, щоб алгоритм перебирав. Але щоб навчити модель із значущою точністю, особливо коли ви говорите про глибоке навчання, вам зазвичай потрібні десятки тисяч зображень, і чим більше, тим краще.

Комп'ютерний зір працює шляхом отримання, обробки та розуміння даних зображення або відео.

Хоча три кроки, що викладають основи комп'ютерного зору, здаються легкими, обробка та розуміння зображення за допомогою машинного зору досить складні.

Зображення складається з кількох пікселів, причому піксель є найменшим квантом, на який можна розділити зображення.

Комп'ютери обробляють зображення у вигляді масиву пікселів, де кожен піксель має набір значень, що представляють наявність та інтенсивність трьох основних кольорів: червоного, зеленого та синього.

Усі пікселі об'єднуються, утворюючи цифрове зображення.

Цифрове зображення, таким чином, стає матрицею, а комп'ютерний зір — дослідженням матриць. У той час як найпростіші алгоритми комп'ютерного зору використовують лінійну алгебру для маніпулювання цими матрицями, складні додатки включають такі операції, як згортки з ядрами, які можна вивчати, і зниження дискретизації через об'єднання.

## 1.3 Еволюція комп'ютерного зору

До появи глибокого навчання завдання, які міг виконувати комп'ютерний зір, були дуже обмеженими і вимагали багато ручного кодування та зусиль від розробників і людей-операторів. Наприклад, якщо ви хочете виконати розпізнавання обличчя, вам потрібно буде виконати такі дії:

1. Створити базу даних: вам потрібно було зафіксувати окремі зображення всіх предметів, які ви хотіли відстежувати, у певному форматі;
2. Анотувати зображення: тоді для кожного окремого зображення вам доведеться ввести кілька ключових точок даних, таких як відстань між очима, ширина перенісся, відстань між верхньою губою та носом, а також десятки інших вимірювань, які визначають унікальні характеристики кожної людини;
3. Зняти новий контент: далі вам потрібно буде зробити нові зображення, будь то фотографії чи відео. А потім потрібно було знову пройти процес вимірювання, позначивши ключові точки на зображенні. Ви також повинні були врахувати кут, під яким було зроблено зображення;

Після всієї цієї ручної роботи програма нарешті зможе порівняти вимірювання в новому зображенні з тими, що зберігаються в базі даних, і повідомити, чи відповідає воно будь-якому з профілів, які він відстежує. Насправді автоматизації було дуже мало, і більшість роботи виконувалася вручну. І похибка все ще була великою.

Машинне навчання забезпечило інший підхід до вирішення проблем комп'ютерного зору. Завдяки машинному навчанню розробникам більше не потрібно вручну кодувати кожне правило у своїх програмах бачення. Замість цього вони запрограмували «функції», менші програми, які могли виявляти певні закономірності в зображеннях. Потім вони використовували алгоритм статистичного навчання, такий як лінійна регресія, логістична регресія, дерева рішень або допоміжні векторні машини (SVM), щоб виявити шаблони та класифікувати зображення та виявити в них об'єкти.

Машинне навчання допомогло вирішити багато проблем, які історично були складними для класичних інструментів і підходів розробки програмного забезпечення. Наприклад, інженери машинного навчання змогли створити програмне забезпечення, яке могло б передбачити рак молочної залози краще, ніж люди-експерти. Однак створення функцій програмного забезпечення вимагало зусиль десятків інженерів і експертів з раку молочної залози і займало багато часу на розробку.

Глибоке навчання є принципово інший підхід до машинного навчання. Глибоке навчання спирається на нейронні мережі - функції загального призначення, які можуть вирішити будь-яку проблему на прикладах. Якщо надати нейронній мережі безліч анотованих прикладів певного типу даних, вона зможе отримати загальні закономірності між цими прикладами та перетворити їх на математичне рівняння, яке допоможе класифікувати майбутні фрагменти інформації.

Наприклад, створення додатка для розпізнавання осіб за допомогою глибокого навчання потребує лише розробки чи вибору готового алгоритму та його навчання на прикладах облич людей, яких він має розпізнати. При достатній кількості прикладів (багато прикладів) нейронна мережа зможе розпізнавати обличчя без додаткових інструкцій щодо його характеристик чи вимірів.

Глибоке навчання є дуже ефективним методом розвитку комп'ютерного зору. У більшості випадків створення хорошого алгоритму глибокого навчання зводиться до збору великої кількості позначених навчальних даних і налаштування таких параметрів, як тип і кількість шарів нейронних мереж і епохи навчання. Порівняно з попередніми типами машинного навчання, глибоке навчання легше та швидше розробляється та впроваджується.

Більшість сучасних програм комп'ютерного зору, таких як виявлення раку, безпілотні автомобілі та розпізнавання обличчя, використовують глибоке навчання. Глибоке навчання та глибокі нейронні мережі перейшли від концептуальної сфери до практичних застосувань завдяки доступності та прогресу в апаратних і хмарних обчислювальних ресурсах.

### 1.4 Історія комп'ютерного зору

Поле комп'ютерного зору з'явилося в 1950 році

Один ранній прорив стався в 1957 році у вигляді машини «Персептрон». Ця «гігантська машина, густо заплутана проводами» була винаходом психолога та піонера комп'ютерного зору Френка Розенблата.

Біологи досліджували, як навчання виникає внаслідок запуску нейронних клітин мозку. Загалом висновок полягав у тому, що навчання відбувається, коли зв'язки між нейронами стають міцнішими. І зв'язки стають міцнішими, коли нейрони з'єднуються частіше.

Розенблатт зрозумів, що той самий процес можна застосувати в комп'ютерах. Perceptron використовував дуже ранню «штучну нейронну

мережу і зміг сортувати зображення за дуже простими категоріями, такими як трикутник і квадрат. Хоча з сьогоднішньої точки зору ранні нейронні мережі Perceptron є надзвичайно рудиментарними, вони створюють важливу основу для подальших досліджень.

Один з головних кроків розвитку комп'ютерного зору вважається робота Лоуренса Робертса «Машинне сприйняття тривимірних тіл», яка була опубліковано в 1963 році і широко вважається одним із попередників сучасного комп'ютерного зору, зображення з його роботи рис. 1.4.

У науковій роботі Ларрі описав процес отримання тривимірної інформації про тверді об'єкти з 2D-фотографій. Він в основному зводив візуальний світ до простих геометричних фігур.

Метою програми, яку він розробив та описав у статті, було обробити 2D-фотографії в лінійні малюнки, потім створити 3D-уявлення з цих ліній і, нарешті, відобразити 3D-структури об'єктів з видаленими всіма прихованими лініями.

Ларрі писав, що процеси побудови 2D в 3D, а потім від 3D до 2D, були хорошою відправною точною для майбутніх досліджень комп'ютерних 3D-систем. Він був категорично правий.

Слід зазначити, що Лоуренс недовго пробув у сфері комп'ютерного зору. Натомість він приєднався до DARPA і тепер відомий як один з винахідників Інтернету.

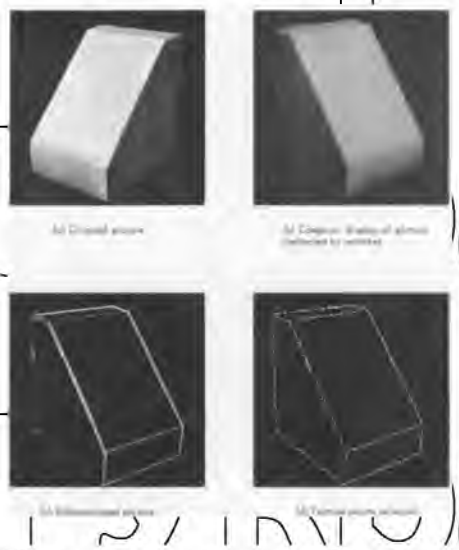


Рис. 1.4. Приклад алгоритму Лоуренса Робертса

У 1960-х роках штучний інтелект став академічною дисципліною, і деякі дослідники, надзвичайно оптимістично налаштовані щодо майбутнього цієї галузі, вважали, що для створення комп'ютера, такого розумного, як людина, знадобиться не більше 25 років. Це був період, коли Сеймур Пеперт, професор лабораторії штучного інтелекту Массачусетського технологічного інституту, вирішив запустити проект Summer Vision і за кілька місяців вирішити проблему машинного зору.

Ключовим моментом стало заснування лабораторії штучного інтелекту в Массачусетському технологічному інституті в 1959 році. Одним із співзасновників був Марвін Мінський, і в 1966 році він дав студенту бакалаврату конкретне завдання: «Проведіть літо, підключаючи камеру до комп'ютера. Потім змусьте комп'ютер описати те, що він бачить».

Він дотримувався думки, що невелика група студентів Массачусетського технологічного інституту здатна розробити значну частину зорової системи за одне літо. Студенти, координовані Сеймуром, повинні були розробити платформу, яка могла б автоматично виконувати сегментацію фону/переднього плану та витягувати об'єкти, що не перекриваються, із зображень реального світу.

Проект не мав успіху. Через п'ятдесят років ми все ще не близькі до вирішення проблеми комп'ютерного зору. Однак цей проект, на думку багатьох, став офіційним народженням CV як наукової галузі.

Розклад Мінського був занадто впевненим. Але був прогрес, наприклад, розуміння того, де знаходяться краї об'єкта на зображенні. У 70-х роках з'явилися перші комерційні застосування технології комп'ютерного зору. Оптичне розпізнавання символів робить набраний, рукописний або друкований текст зрозумілим для комп'ютерів. У 1978 році компанія Kurzweil Computer Products випустила свій перший продукт. Метою було дати можливість незрячим людям читати за допомогою комп'ютерних програм.

У 1980-х роках широко використовувалися штучні нейронні мережі, започатковані Френком Розенблатом у 50-х роках. Нові нейронні мережі були

більш складними в тому, що робота по інтерпретації зображення відбувалася на кількох «шарах».

У другій половині 90-х все почало загострюватися. У цьому десятилітті вперше було використано статистичні методи розпізнавання облич на зображеннях; і посилення взаємодії між комп'ютерною графікою та комп'ютерним зором. Прогрес ще більше прискорився завдяки Інтернету. Частково це сталося завдяки тому, що в Інтернеті стали доступні більші анотовані набори зображень. А за останнє десятиліття цей прогрес у сфері комп’ютерного зору почався. Дві основні причини спонукали до проривів: більше і краще обладнання; і багато, набагато більше даних.

Апаратне забезпечення. У цьому є дві частини. По-перше, обчислювальна потужність просто стала більшою і дешевшою. Наприклад, коли штучний інтелект Google навчився розпізнавати мордочки котів у відео YouTube, у нього було 16 000 комп’ютерів, які працювали разом, щоб зробити це.

Другий аспект полягає в тому, що обладнання стало краще розроблене для завдань комп'ютерного зору. Мій наступний пост буде зосереджено більш детально на ШІ та глибокому навчанні, але ось короткий дегустатор. Графічний процесор (GPU) — це чіп, спочатку розроблений для ігрової індустрії. Виявляється, це також дуже корисно в обчислювальній роботі. Графічні процесори розкрили потенціал нейронних мереж як підходу до комп’ютерного зору (і насправді до інших важких проблем ШІ).

Більше даних для подачі в систему. У тому ж прикладі з котами Google надав цим 16 000 комп’ютерам 10 мільйонів відео. Кількість доступних даних привела до якісного вдосконалення алгоритмів комп'ютерного зору.

Історія комп’ютерного зору – це багато в чому історія про штучні нейронні мережі. У надзвичайно рудиментарній формі вони лежать в основі раннього прогресу. Глибокі нейронні мережі, які працюють на основі графічних процесорів і великої кількості даних, сьогодні сприяють прогресу. Оскільки прогрес прискорився, так і захоплення інвесторів — за оцінками, ринок комп’ютерного зору досягне 50 мільярдів доларів до 2022 року.

## 1.5 Застосування комп'ютерного зору

Комп’ютерний зір є однією з областей машинного навчання, де основні концепції вже інтегровані в основні продукти, які ми використовуємо щодня.

### 1.5.1 Комп’ютерний зір в безпілотних автомобілях

Але не лише технологічні компанії використовують машинне навчання для графічних додатків.

Комп’ютерний зір дає можливість самокерованим автомобілям розуміти навколишнє середовище. Камери знімають відео з різних кутів навколо автомобіля і передають його в програмне забезпечення комп’ютерного зору, яке потім обробляє зображення в режимі реального часу, щоб знайти кінці доріг, зчитувати дорожні знаки, виявляти інші автомобілі, об’єкти та пішоходів. Тоді самокерований автомобіль може рухатися по вулицях і шосе, уникати наїзду на перешкоди та безпечно довозити пасажирів до місця призначення.

### 1.5.2 Комп’ютерний зір у доповненій та змішаній реальності

Комп’ютерний зір також відіграє важливу роль у доповненій та змішаній реальності, технології, яка дозволяє комп’ютерним пристроям, таким як смартфони, планшети та розумні окуляри, накладати та вбудовувати віртуальні об’єкти на зображення реального світу. Використовуючи комп’ютерний зір, обладнання AR виявляє об’єкти в реальному світі, щоб визначити розташування на дисплеї пристрою для розміщення віртуального об’єкта. Наприклад, алгоритми комп’ютерного зору можуть допомогти додаткам AR виявляти такі площини, як стільниці, стіни та підлоги, що є дуже важливою частиною встановлення глибини та розмірів та розміщення віртуальних об’єктів у фізичному світі, приклад додатку для планування інтер’єру для приміщення рис. 1.5.

Рис. 1.5. Приклад додатку

1.5.3. Комп'ютерний зір в охорон здоров'я

Інформація про зображення є ключовим елементом для діагностики в медицині, оскільки на неї припадає 90 відсотків усіх медичних даних. Багато діагнозів у здоров'ї ґрунтуються на обробці зображень — рентгенівських променів, МРТ та мамографії, і це лише деякі з них. А сегментація зображень довела свою ефективність під час аналізу медичних сканів. Наприклад, алгоритми комп'ютерного зору можуть виявити діабетичну ретинопатію, найбільш швидко зростаючу причину сліпоти. Комп'ютерний зір може обробляти зображення задньої частини ока рис. 1.6. і оцінювати їх наявність та тяжкість захворювання.

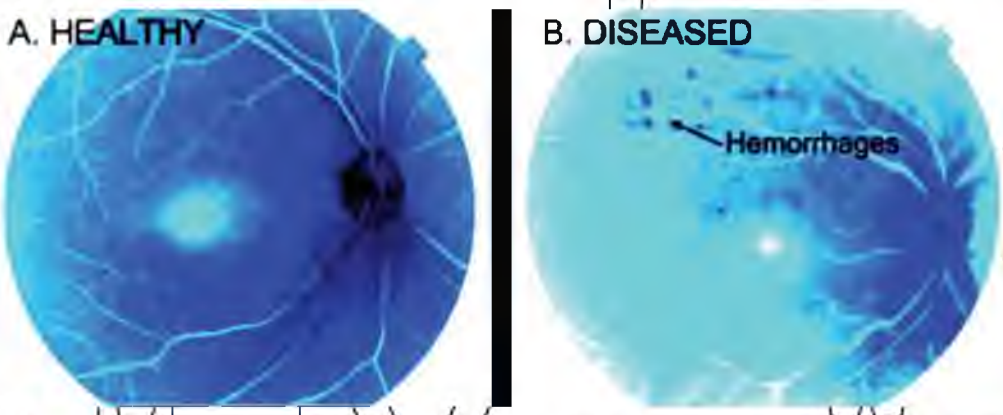



Рис. 1.6. Обстеження задньої ока комп'ютерним зором

Виявлення раку є ще одним яскравим прикладом. Точність у діагностиці різних форм раку життєво важлива. За даними Google, інструменти комп'ютерного зору допомагають виявляти метастази раку з набагато більшою точністю, ніж лікарі-люди. Нижче ви можете побачити крупний план біопсії лімфатичних вузлів рис. 1.7. Тканина містить метастази раку молочної залози, а також ділянки, схожі на пухлину, але є доброякісними. Алгоритм комп'ютерного зору успішно визначає область пухлини (яскраво-зелений) і не плутається з нормальними ділянками, які виглядають як пухлини.

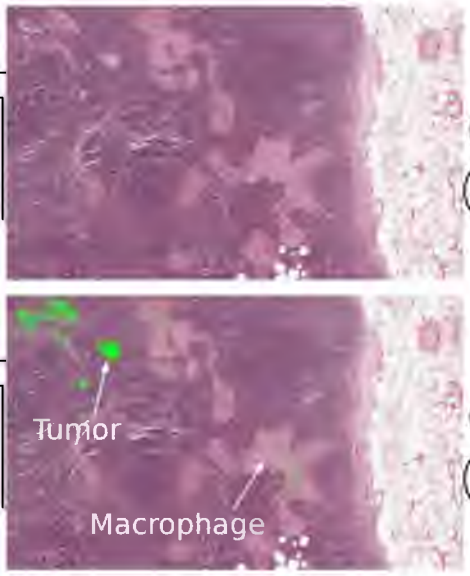



Рис. 1.7. Комп'ютерний зір допомагає виявляти метастази раку

### 1.5.4 Комп'ютерний зір у розпізнаванні облич

Комп'ютерний зір також відіграє важливу роль у програмах розпізнавання облич — технології, яка дозволяє комп'ютерам узгоджувати зображення облич людей з їхніми особами. Алгоритми комп'ютерного зору визначають риси обличчя на зображеннях і порівнюють їх з базами даних профілів обличчя. Пристрої використовують розпізнавання облич для автентифікації особи своїх власників. Додатки соціальних мереж використовують розпізнавання облич для виявлення та позначення користувачів. Правоохоронні органи також покладаються на технологію розпізнавання облич, щоб ідентифікувати злочинців у відеоканалах. Приклад розпізнавання обличчя рис. 1.8.

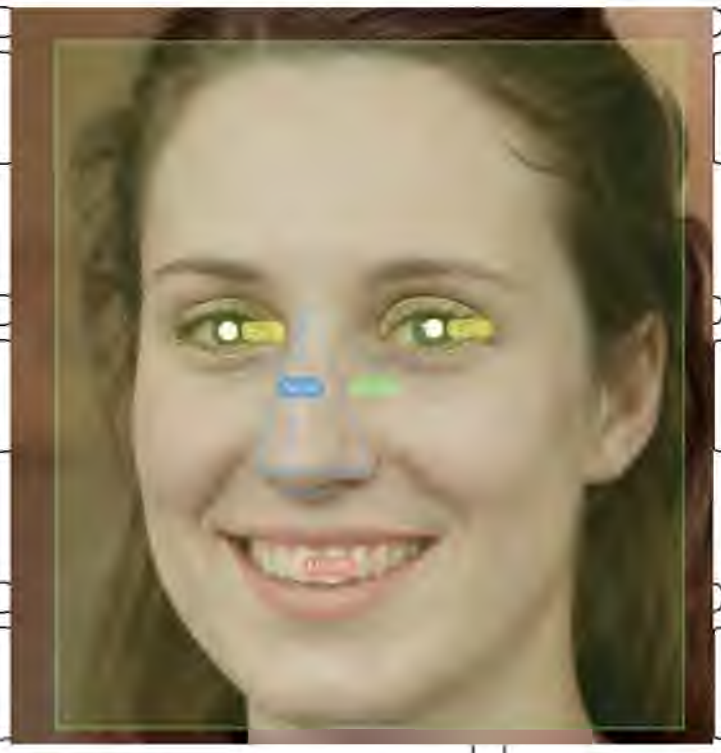


Рис. 1.8. Приклад розпізнавання обличчя рисунок

1.5.5 Комп'ютерний зір в сільському господарстві

Багато сільськогосподарських організацій використовують комп'ютерний зір для моніторингу врожаю та вирішення поширених сільськогосподарських проблем, таких як поява бур'янів або дефіцит поживних речовин. Системи комп'ютерного зору обробляють зображення з супутників, дронів або літаків і намагаються виявити проблеми на ранній стадії, що допомагає уникнути непотрібних фінансових втрат.

1.5.6 Комп'ютерний зір в роздрібній торгівлі

Amazon була першою компанією, яка відкрила магазин, який працює без каси або касових апаратів. Amazon Go оснащений сотнями камер комп'ютерного зору. Ці пристрої відстежують товари, які клієнти кладуть у свої кошики. Камери також можуть відстежувати, чи повертає клієнт товар на полицю та видаляє його з віртуального кошика. Клієнти оплачують через додаток Amazon Go, що виключає будь-яку необхідність залишатися в черзі. Камери також запобігають крадіжкам у магазинах та запобігають виходу товару.

## 1.6 Виклики комп'ютерного зору

Допомагати комп'ютерам бачити виявляється дуже важко.

Винайти машину, яка бачить так, як ми, є оманливо важким завданням, не тільки тому, що важко змусити комп'ютери робити це, але й тому, що ми не зовсім впевнені, як взагалі працює людський зір.

Вивчення біологічного зору вимагає розуміння органів сприйняття, таких як очі, а також інтерпретації сприйняття в мозку. Було досягнуто значного прогресу, як у складанні діаграм процесу, так і в тому, щоб виявити хитрощі та ярлики, які використовує система, хоча, як і будь-яке дослідження, що залучає мозок, попереду довгий шлях.

Багато популярних програм комп'ютерного зору включають спроби розпізнавати речі на фотографіях наприклад:

– Класифікація об'єктів: яка широка категорія об'єктів на цій фотографії?;

– Ідентифікація об'єкта: який тип даного об'єкта зображений на цій фотографії?;

– Перевірка об'єкта: чи є об'єкт на фотографії?;

– Виявлення об'єктів: де знаходяться об'єкти на фотографії? ;

– Виявлення орієнтирів об'єкта: які ключові моменти для об'єкта на фотографії?;

– Сегментація об'єкта: які пікселі належать об'єкту на зображенні?;

– Розпізнавання об'єктів: які об'єкти на цій фотографії і де вони?;

Крім простого визнання, інші методи аналізу включають:

– Аналіз руху відео використовує комп'ютерний зір для оцінки швидкості об'єктів у відео або самої камери;

– У сегментації зображень алгоритми розбивають зображення на кілька наборів переглядів;

– Реконструкція сцени створює 3D-модель сцени, введену через зображення або відео;

– Під час відновлення зображень шуми, наприклад розмиття, видаляються з фотографій за допомогою фільтрів на основі машинного навчання;

## 1.7 IP-камера

Це різновид відеокамери, що передає сигнал локальною або глобальною мережею. Свою назву вони отримали через те, що кожна камера в мережі є окремим клієнтом і має виділену IP-адресу.

Технічно IP-камери – суттєво складніші пристрої, ніж аналогові камери для відеоспостереження, оскільки вони включають не лише об'єктив та матрицю, на якій фіксується зображення, а й центральний процесор та мережевий інтерфейс. Процесор відповідає за обробку сигналу: він може передаватися на монітор «як є» або стискатися за стандартним алгоритмом (за допомогою так званого кодека, наприклад h.264) зниження навантаження на мережу. Мережевий інтерфейс використовується для підключення камери до центрального вузла системи відеоспостереження та передачі отриманих даних у вигляді стандартного мережного протоколу (наприклад, TCP).

## 1.8 Принцип роботи IP-камер

Як і у звичайних камерах, в IP-моделях об'єктив фокусує картинку на матриці, яка у свою чергу перетворює світло на електричний сигнал. Він передається на процесор, який обробляє кольори, яскравість та інші параметри зображення. Після цього відео надходить на компресор, який стискає дані передачі їх через мережевий контролер.

Кожній IP-камері при підключенні надається власна IP-адреса, як і іншим пристроям, які працюють через інтернет. Він необхідний, щоб камера могла

синхронізуватися з реєстратором, що відбувається за допомогою спеціальної програми або команди. Без IP-адреси забезпечити спільну роботу та доступ до камери з мобільних гаджетів буде неможливо.

Завдяки наявності цифрових компонентів функціональність IP-камери стає практично безмежною, дозволяючи отримувати доступ до її даних із будь-якої точки планети, де є інтернет.

## 1.9 Варіанти підключення до IP-камери

IP-камери забезпечують велику зручність та функціональність роботи завдяки можливості підключення до:

– реєстратора, персонального комп'ютера чи хмарного сховища даних. Користувач може вибрати найбільш підходящий варіант або використовувати кілька одночасно;

– роутера або комутатора за наявності, що підтримують мережевий протокол динамічного налаштування вузла. Завдяки цьому до реєстратора можна підключити кілька камер одночасно;

– адаптера, PoE-комутатора або реєстратора для отримання живлення. За допомогою PoE живлення та дані передаються одночасно з використанням крученої пари;

Спосіб підключення до персонального комп'ютера залежить від кількості IP-камер. Якщо камера одна, достатньо використання LAN-інтерфейсу мережної карти. Якщо більше – до LAN підключають комутатор, а до нього – камери з присвоєнням IP-адрес.

## 1.10 Технології передачі даних в ip-камерах

Камери цього типу працюють за стеком протоколів TCP/IP. Це модель мережного підключення, яка має чотири рівні передачі інформації:

– прикладний - HTTP, RTSP, FTP, DNS та інші;

– транспортний - TCP, UDP, SCTP, DCCP та ін. (RIP, протоколи маршрутизації типу OSPF, що працюють поверх IP, - частина мережевого рівня);

– мережевий - IP (допоміжні протоколи, наприклад ICMP і IGMP, працюють поверх мережевого протоколу, але відносяться до мережного рівня, а ARP - самостійний допоміжний протокол, що працює поверх канального рівня);

– мережевий доступ – Ethernet, IEEE 802.11 WLAN, SLIP, Token Ring, ATM та MPLS, фізичне середовище та принципи кодування інформації, T1, E1;

### 1.11 Транспортні протоколи

TCP – транспортний протокол, ефективність якого доведена випробуваннями. Вже перші досліди показали, що пакет даних пройшов TCP 150 тисяч кілометрів без найменших втрат. Протокол TCP за допомогою команд спочатку забезпечує підключення, а потім стартує трансляція. Має кілька важливих переваг – технологія перевіряє цілісність інформації та її послідовність, а також налаштовує швидкість передачі даних таким чином, щоб вони не відправлялися швидше, ніж їх можна прийняти, що запобігає несправностям та втраті трафіку. Більше того, цей протокол може виправляти помилки шляхом відправлення дубля пакета, якщо частина даних була втрачена або видаляє зайвий пакет при відправці двох однакових за однією і тією ж адресою.

UDP – це альтернатива протоколу TCP, але з меншим набором функцій. Перш за все він починає передачу даних без попередньої установки з'єднання, а також не контролює процес відправлення пакетів. Таким чином він працює швидше, але менш надійно.

RTP – протокол, який передає дані у режимі реального часу. З переваг – синхронізація інформації та виправлення послідовності передачі пакетів.

Таким чином, найбільш швидкими технологіями передачі в режимі реального часу є RTP або UDP. Але якщо мережа має проблеми, то найкращим вибором буде протокол TCP, який допомагає виправити помилки, приклад технологій передачі даних в ip-камерах рис. 1.9.

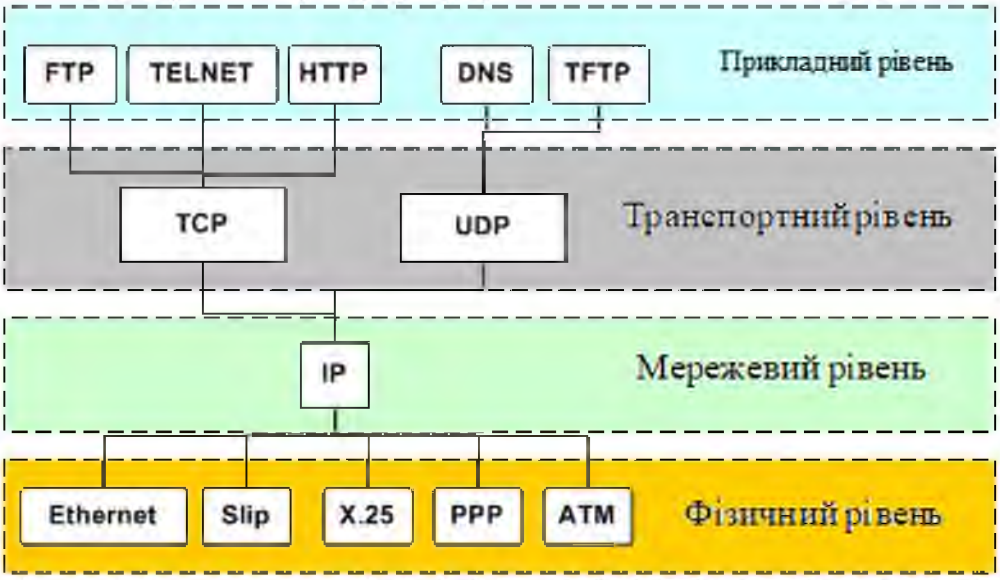



Рис. 1.9. Технології передачі даних в ip-камерах

## 1.12 Протоколи сумісності

Щоб пристрої без проблем обмінювалися даними, вони мають бути сумісні. Так, техніка одного бренду сумісна за умовчанням. Для того, щоб IP-камери могли працювати з іншими пристроями, потрібна підтримка прикладних протоколів. Це, як правило, RTSP та ONVIF.

RTSP – прикладний протокол реального часу, головна функція якого – віддалене керування камерою сервером за допомогою опису команд. Цей стандарт не виконує стиснення, не має відношення до пакетів та визначення транспортного протоколу. По суті, трансляція – не частина RTSP, оскільки займається окремий стандартний протокол. RTSP-команди передаються за допомогою спеціального порту окремо від основного потоку.

Запити:

- Announce – оновлення даних опису вмісту;
- Describe – опис вмісту;
- Options – методи, що підтримуються;
- Play – початок передачі вмісту;
- Pause – тимчасова зупинка передачі;
- Record – запис вмісту сервером;
- Redirect – перенаправлення на інший вміст;
- Setup – встановлення транспортного механізму;
- Get_parameter – запит зазначених параметрів сервера;
- Set_parameter – встановлення параметрів сервера;
- Teardown – зупинка потоку, визволення ресурсів;

ONVIF – один із сучасних протоколів, який поєднує ряд готових технологій та стандартів (у тому числі, RTSP), які були спеціально адаптовані для IP-відеоспостереження. Під цю специфікацію було створено чотири профілі для чотирьох класів пристроїв:

- Profile S 0 для відеоджерел;
- Profile C – для СКУД;
- Profile G – для записуючих відеопристроїв;
- Profile Q – для пристроїв, сумісних із коробки.

З HTTP працювали застарілі моделі відеокамер, у яких відеопотік розкладався на кадри у форматі JPEG і викладався на веб-сервері відеокамери, а клієнт забирав їх із певною частотою. Це не потокова передача даних, вона називалася JPEG over HTTP. Нині такий метод не використовується.

RTP (Real Time Transport Protocol) це ще один варіант стрімінгового протоколу, що використовується для передачі даних у режимі реального часу. RTP працює, як правило, поверх UDP і не використовує зарезервовані порти як RTSP, це може стати проблемою, якщо вам треба відправити відеопотік кудись за міжмережевий екран, в інший сегмент мережі або взагалі в іншу мережу.

### 1.13 Способи передачі сигналу ip-камерою

IP-камера може пропонувати три способи трансляції відеосигналу: дротовий, бездротовий та гібридний (об'єднує дротовий та бездротовий).

Найбільш стабільну передачу даних забезпечує провідне з'єднання, але має обмеження, пов'язані з довжиною кабелю. Віта пара – 100 метрів, коаксіальний кабель – 500 метрів. Найбільша відстань забезпечує оптоволокно – 100 кілометрів (без комутаторів та повторювачів).

Бездротову передачу даних в IP-камері інтегрований WiFi-модуль (як правило) та/або 3G/4G-модуль. Це пропонує можливість зручнішого розміщення камер, але дальність сигналу обмежуватиметься фізичними перешкодами, наприклад, стінами та електромагнітними перешкодами.

Гібридні моделі, які поєднують дротові та бездротові технології передачі даних, пропонують найвищий рівень надійності роботи локальної мережі.

### 1.14 Ethernet: локальна провідна мережа для роботи ip-камер

IP-камера підключається до мережі Ethernet, яка об'єднує всіх користувачів у локальну мережу (LAN), забезпечуючи пакетну трансляцію інформації. Таким чином, ви можете використовувати локальну мережу об'єкта, в яку підключено комп'ютери та інші пристрої.

Ethernet описується стандартами групи IEEE 802, що визначають формат кадрів та протоколи управління доступом до середовища на канальному рівні схеми взаємодії пристроїв один з одним.

# 2 Проектування

## 2.1 Вибір алгоритму розпізнавання облич

Виявлення облич — це комп'ютерна технологія на основі штучного інтелекту, яка використовується для вилучення та ідентифікації людських облич із цифрових зображень. При інтеграції з біометричними системами безпеки (зокрема, з розпізнаванням облич), така технологія дає змогу стежити за людьми в режимі реального часу. У програмах, які використовують відстеження, аналіз і розпізнавання обличчя, розпізнавання обличчя зазвичай працює як перший крок і має значний вплив на те, як будуть виконуватися послідовні операції в програмі.

Виявлення облич допомагає аналізувати обличчя, визначаючи частини відео чи зображення, на яких слід зосередитися при визначенні статі, віку та емоцій. Аналогічно, із системами розпізнавання облич (які створюють карти рис обличчя з «відбитком обличчя») дані розпізнавання облич включені в алгоритми системи. Розпізнавання обличчя допомагає визначити, які частини відео або зображення потрібні для створення відбитка обличчя.

### 2.1.1 Вибір методу виявлення облич

Спочатку комп'ютер розглядає фотографію або відеозображення і намагається відрізнити обличчя від будь-яких інших об'єктів на задньому плані. Існують методи, за допомогою яких комп'ютер може цього досягти, компенсуючи освітлення, орієнтацію або відстань до камери. Ці методи поділяються на чотири категорії, і алгоритми розпізнавання обличчя можуть належати до двох або більше груп.

**Виявлення обличчя на базі знань.**

Цей метод спирається на набір правил, розроблених людьми відповідно до наших знань. Ми знаємо, що обличчя повинно мати ніс, очі та рот на певній відстані

та положенні один від одного. Проблема цього методу полягає в тому, щоб створити відповідний набір правил. Якщо правила занадто загальні або надто детальні, система отримує багато помилкових спрацьовувань. Однак він працює не для всіх кольорів шкіри і залежить від умов освітлення, які можуть змінити точний відтінок шкіри людини на знімку.

**Відповідність шаблону.**

Метод відповідності шаблону використовує попередньо визначені або параметризовані шаблони облич, щоб знайти або виявити грані за кореляцією між попередньо визначеними або деформованими шаблонами та вхідними зображеннями. Модель обличчя може бути побудована за ребрами за допомогою методу виявлення країв.

Для цього підходу програмне забезпечення має кілька класифікаторів для виявлення різних типів облич, а також деякі для профільних облич, таких як детектори очей, носа, рота, а в деяких випадках навіть всього тіла. Хоча цей підхід простий у реалізації, він зазвичай недостатній для виявлення обличчя.

**Виявлення обличчя на основі зовнішнього вигляду.**

Метод заснований на зовнішньому вигляді, залежить від набору тренінгових зображень обличчя делегатів для визначення моделей обличчя. Він покладається на машинне навчання та статистичний аналіз, щоб знайти відповідні характеристики зображень обличчя та виділити з них особливості.

**Виявлення на основі характеристик обличчя**

Метод на основі ознак виділяє структурні особливості обличчя. Він навчається як класифікатор, а потім використовується для диференціації лицьових та нелицьових областей. Одним із прикладів цього методу є розпізнавання обличчя на основі кольору, яке сканує кольорові зображення або відео на предмет ділянок із типовим кольором шкіри, а потім шукає сегменти обличчя.

Одним з таких методів є техніка визначення обличчя Віоли-Джонс, відома як каскади Хаара.

Вибір ознак Хаара спирається на подібні властивості людських облич, щоб сформувати відповідності з рисами обличчя: розташування та розмір ока, рота, перенісся та орієнтовані градієнти інтенсивності пікселів.

Цей алгоритм виявлення об’єктів використовується для ідентифікації облич на зображенні або відео в реальному часі. Алгоритм використовує функції виявлення країв або ліній, запропоновані Віолою та Джонсом у їхній дослідницькій роботі «Швидке виявлення об’єктів з використанням посиленого каскаду простих ознак», опублікованій у 2001 році. Алгоритм надає багато позитивних зображень, що складаються з облич, і багато негативні зображення, які не складаються з будь-якого обличчя, щоб на них тренуватися.

Незважаючи на те, що Віола-Джонс є застарілою системою, вона досить потужна, і її застосування виявилося надзвичайно помітним у розпізнаванні облич у реальному часі. Цей алгоритм повільно тренується, але може виявляти обличчя в режимі реального часу з вражаючою швидкістю.

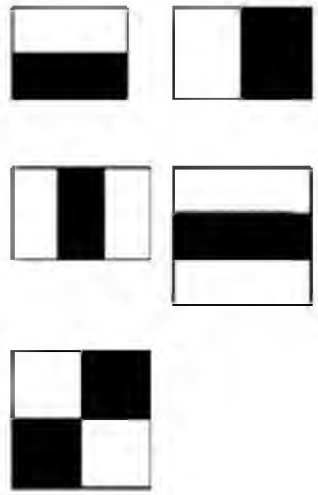
Дано зображення (цей алгоритм працює з зображенням у відтінках сірого), алгоритм розглядає багато менших субрегіонів і намагається знайти обличчя, шукаючи конкретні ознаки в кожному субрегіоні. Потрібно перевірити багато різних положень і масштабів, оскільки зображення може містити багато граней різних розмірів. Віола і Джонс використовували функції, подібні до Хаара, для виявлення облич у цьому алгоритмі.

Алгоритм Віоли Джонс має чотири основні кроки:

1. Вибір характеристика, подібних до Хаара;
2. Створення щілісного зображення;
3. Запуск навчання AdaBoost;
4. Каскадний класифікатор;

Характеристики, подібні до Хаара – це функції цифрового зображення, що використовуються для розпізнавання об’єктів. Усі людські обличчя мають деякі універсальні властивості людського обличчя, наприклад, область очей темніша за сусідні пікселі, а область носа світліша за область очей.

Простий спосіб дізнатися, яка область світліша чи темніша, — підсумувати значення пікселів обох регіонів і порівняти їх. Сума значень пікселів у темній області буде меншою за суму пікселів у світлішій області. Якщо одна сторона світліша за іншу, це може бути край брови або іноді середня частина може бути блискучішою за навколишні коробочки, що можна інтерпретувати як ніс. Це можна зробити за допомогою функцій, подібних до Хаара, і за допомогою них ми можемо інтерпретувати різні частини обличчя. Функції Хаара рис. 2.1.



Характеристика краю

Характеристика ліній

Характеристика чотирьох прямокутників

Рис. 2.1. Функції Хаара

Існує 3 типи ознак Хаара, які Віола і Джонс визначили у своєму дослідженні:

- Характеристика краю;
- Характеристика ліній;
- Характеристика чотирьох прямокутників;

Елементи країв і лінії ліній корисні для виявлення країв і ліній відповідно. Чотиристоронні об'єкти використовуються для пошуку діагональних об'єктів

Значення функції обчислюється як одне число: сума значень пікселів у чорній області мінус сума значень пікселів у білій області. Значення дорівнює нулю для простої поверхні, на якій всі пікселі мають однакове значення, і, таким чином, не надають корисної інформації.

Оскільки наші обличчя мають складні форми з темнішими та яскравішими плямами, функція, подібна до Хаара, дає вам велику кількість, коли області в чорно-білих прямокутниках дуже різні. Використовуючи це значення, ми отримуємо частину дійсної інформації із зображення.

Щоб бути корисною, функція, подібна до Хаара, повинна дати вам велике число, що означає, що області чорного та білого прямокутників дуже різні. Є відомі функції, які дуже добре розпізнають людські обличчя рис. 2.2

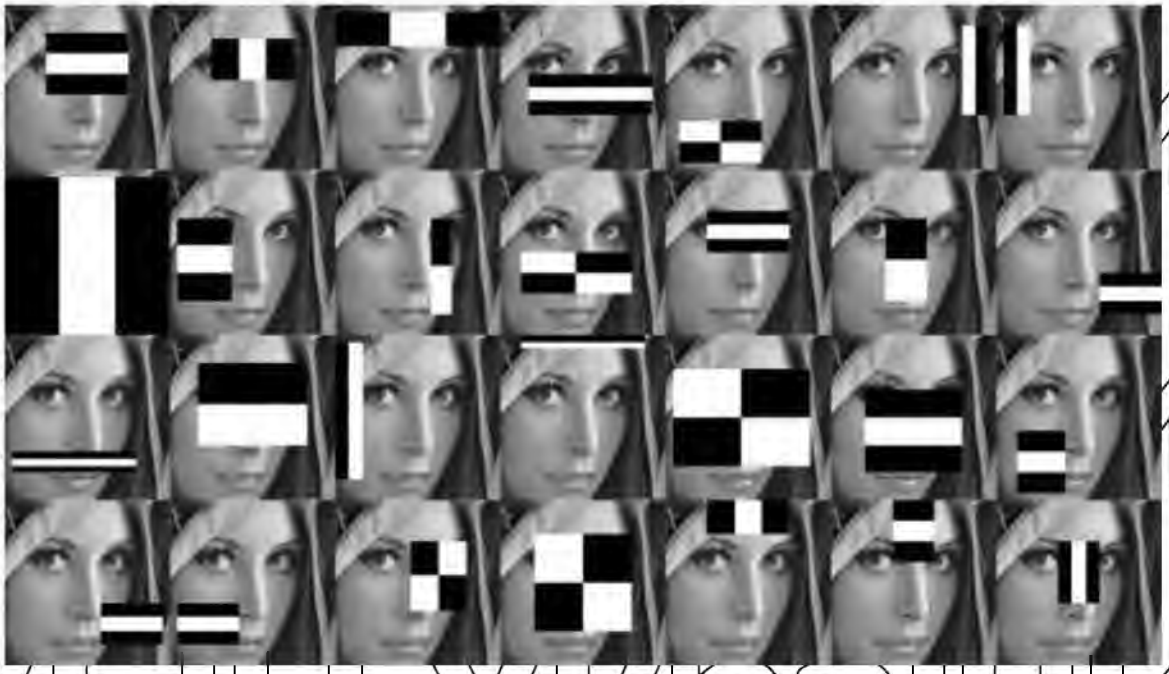


Рис. 2.2. Використання функцій Хаара на обличчі

Для чорно-білого зображення значення пікселів дорівнюють 0 або 1, але в реальних випадках ми маємо нормалізувати зображення у відтінках сірого, що містить значення пікселів, приклад на рис. 2.3.


Рис. 2.3. Приклад опрацювання зображення

Інтегральне зображення відіграє свою роль, дозволяючи нам швидко виконувати обчислення, щоб ми могли зрозуміти, чи відповідає певна ознака кількох ознак критеріям.

Цілісне зображення це назва як структури даних, так і алгоритму, що використовується для отримання цієї структури даних. Він використовується як швидкий та ефективний спосіб обчислення суми значень пікселів у зображенні або прямокутній частині зображення.

У ціленому зображенні значення кожної точки є сумою всіх пікселів зверху та зліва, включаючи цільовий піксель.

Використовуючи інтегральні зображення, ми заощаджуємо багато часу на обчислення підсумовування всіх пікселів у прямокутнику, оскільки нам потрібно виконувати обчислення лише на чотирьох краях прямокутника, приклад на рис. 2.4.


Рис. 2.4. Приклад обчислень

Коли ми додаємо пікселі в синьому квадраті, ми отримуємо 8 як суму всіх пікселів, і тут ми маємо шість елементів, задіяних у вашому обчисленні. Тепер, щоб обчислити суму цих самих пікселів за допомогою цього зображення, вам просто потрібно знайти кути прямокутника, а потім додати зелені вершини та відняти вершини в червоних квадратах.

Ми отримуємо ту саму відповідь. 8, а в обчисленнях беруть участь лише чотири числа. Незалежно від того, скільки пікселів знаходиться в прямокутній коробці, нам просто потрібно буде обчислити ці 4 вершини.

Тепер, щоб обчислити значення будь-якої функції, схожої на Хаар, є простий спосіб обчислити різницю між сумами значень пікселів двох прямокутників.

Далі ми використовуємо алгоритм машинного навчання, відомий як AdaBoost.

Кількість ознак, які присутні у вікні детектора 24×24, становить майже 160 000, але лише деякі з них важливі для ідентифікації обличчя. Тому ми використовуємо алгоритм AdaBoost для визначення найкращих функцій із 160 000 функцій. В алгоритмі Віоли-Джонса кожна ознака, подібна до Хаара, представляє слабкого учня. Щоб визначити тип і розмір функції, яка входить до остаточного

класифікатора, AdaBoost перевіряє продуктивність усіх класифікаторів, які ви йому надаєте.

Щоб розрахувати ефективність класифікатора, ви оцінюєте його для всіх субрегіонів усіх зображень, які використовуються для навчання. Деякі субрегіони дадуть сильний відгук у класифікаторі. Вони будуть класифіковані як позитивні, тобто класифікатор вважає, що на ньому є людське обличчя. Субрегіони, які не дають сильної реакції, не містять людського обличчя, на думку класифікаторів. Вони будуть класифіковані як негативні.

Класифікаторам, які показали хороші результати, надається більша важливість або вага. Кінцевим результатом є сильний класифікатор, також званий посиленим класифікатором, який містить найефективніші слабкі класифікатори.

Тому коли ми навчаємо AdaBoost визначати важливі функції, ми надаємо йому інформацію у вигляді навчальних даних, а потім навчаємо його вчитися на основі інформації для прогнозування. Таким чином, зрештою, алгоритм встановлює мінімальний поріг, щоб визначити, чи можна щось класифікувати як корисну функцію чи ні.

Каскадний класифікатор складається з серії етапів, де кожен етап є сукупністю слабких учнів. Слабкі учні навчаються за допомогою підвищення, що дозволяє отримати високоточний класифікатор із середнього прогнозу всіх слабких учнів.

На основі цього прогнозу класифікатор або вирішує вказати, що об’єкт знайдено (позитивний), або переходить до наступної області (негативний). Етапи призначені для максимально швидкого відкидання негативних зразків, оскільки більшість вікон не містять нічого цікавого.

Важливо максимізувати низький рівень помилкових негативних результатів, оскільки класифікація об’єкта як необ’єкта суттєво погіршить ваш алгоритм виявлення об’єктів. На рис. 2.5. показуються каскади Хаара в дії. Червоні квадратики позначають «позитивні результати» від слабких учнів.

Рис. 2.5. Результати знаходження функцій слабкими учнями

Переваги підходу:

– Цей алгоритм є популярним та розповсюдженим методом виявлення облич;
– Швидко виявляє обличчя за допомоги каскадного класифікатора;
– Має дуже високу точність;
– Надає низький відсоток помилкових знаходжень обличчя, якщо порівнювати с точність алгоритмів які повільніші;
– Має багато імплементацій в різних інструментах розробки програмного забезпечення;
– Не потребує високопродуктивного апаратного обладнання;

Недоліки підходу:

– Потребує великої вибірки зображень для кращого результату
– Навчання потребує багато часу
– Має обмеження на розташування обличчя під час знаходження обличчя

### 2.1.2 Короткий опис процесу розпізнавання осіб

В програму надходить зображення, його можна отримати будь-яким зручним шляхом. Це залежить від того, як буду спроектована система. За допомоги алгоритмів знаходиться обличчя на зображенні, визначаються його межі, це є етап виявлення обличчя на зображенні.

Наступний крок це етап розпізнавання. На цьому етапі зображення обличчя трансформується, змінюється параметри яскравості, відбувається процес вирівняння та масштабується. Чим краще зображення, тим краще буде проведено наступний етап.

Етап вилучення рис обличчя — це процес вилучення з зображення обличчя таких компонентів як очі, ніс, рот тощо. Вилучення рис обличчя дуже важливо для точного розпізнавання. Серед усіх рис обличчя важливе значення мають локалізація та виявлення очей, за якими можна визначити розташування всіх інших рис обличчя.

На ключовому етапі відбувається безпосереднє порівняння отриманого зображення з зображеннями облич осіб існуючими в базі даних.

Саме цей етап є завершальним, безпосередньо порівняння й називається ідентифікацією облич осіб. Приклад на рис. 2.6.

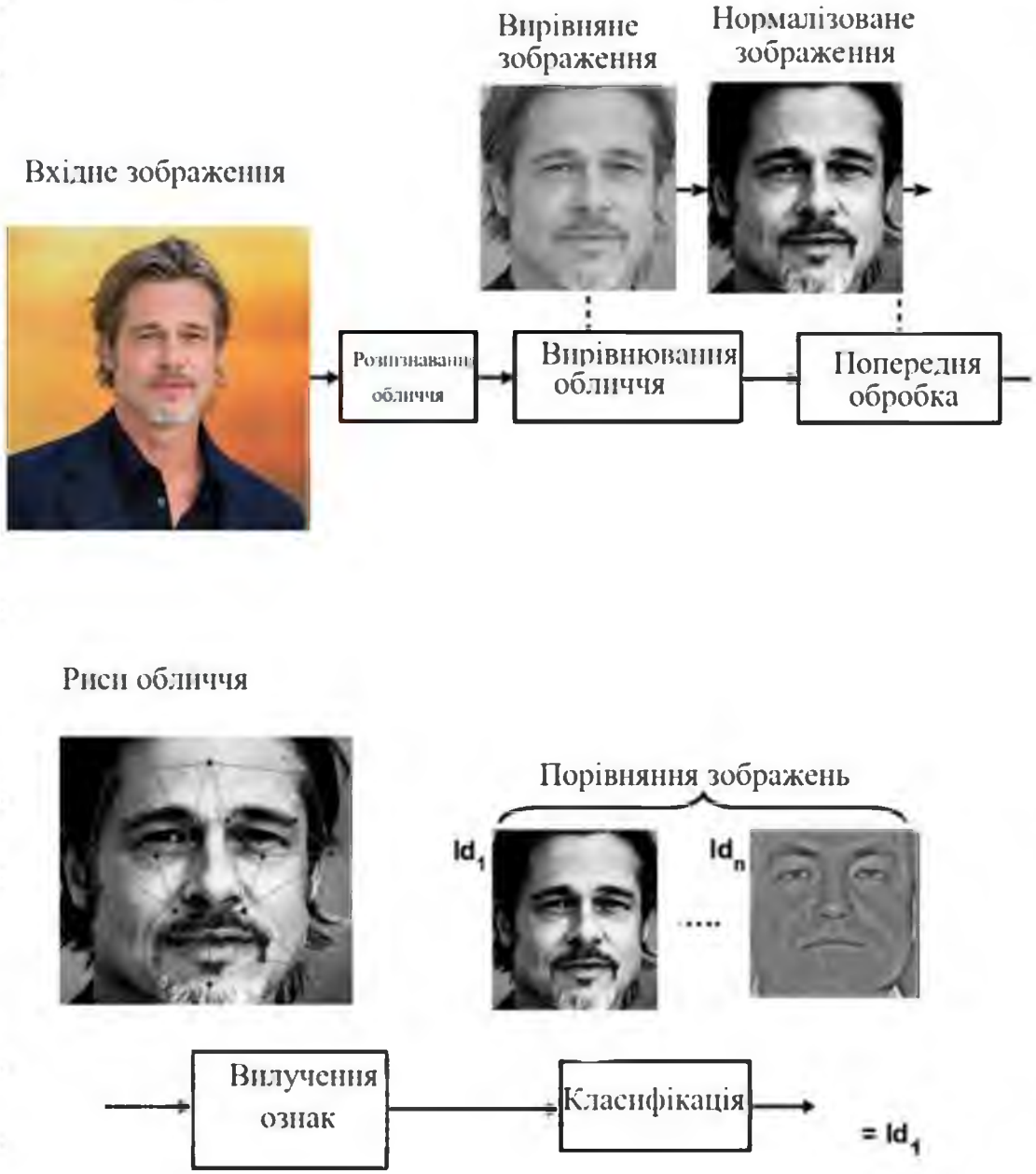



Рис. 2.6. Процес розпізнавання осіб

### 2.1.3. Алгоритми розпізнавання облич

Люди автоматично розпізнають обличчя щодня і практично без зусиль.

Хоча для нас це звучить як дуже просте завдання, воно виявилося складним завданням для комп'ютера, оскільки воно має багато змінних, які можуть погіршити точність методів, наприклад: зміна освітлення, низька роздільна здатність, оклюзія, серед іншого .

У інформатиці розпізнавання облич це в основному завдання розпізнавання людини на основі її зображення обличчя. За останні два десятиліття він став дуже популярним, в основному через нові методи та високу якість сучасних відео/камер.

Системи розпізнавання обличчя можуть працювати в основному в двох режимах.

– Перевірка або аутентифікація зображення обличчя: воно в основному порівнює введене зображення обличчя із зображенням особи, пов'язаним із користувачем, який потребує аутентифікації. В основному це порівняння 1x1.

– Ідентифікація або розпізнавання обличчя: воно в основному порівнює введене зображення обличчя з усіма зображеними обличчя з набору даних з метою знайти користувача, який відповідає цьому обличчю. В основному це порівняння 1xN.

Існують різні типи алгоритмів розпізнавання обличчя, наприклад:

– Eigenfaces;

– Гістограми локальних бінарних шаблонів (LBPH) ;

– Fisherfaces;

– Масштабно інваріантне перетворення ознак (SIFT) ;

– Прискорення надійних функцій (SURF) ;

Eigenfaces відноситься до підходу до розпізнавання облич на основі зовнішнього вигляду, який прагне зафіксувати варіації в колекції зображень облич і використовувати цю інформацію для кодування та порівняння зображень окремих облич у цілісному вигляді. Зокрема, власні грані є основними компонентами розподілу граней, або, що еквівалентно, власними векторами матриці коваріації набору зображень обличчя, де зображення з N пікселів вважається точкою (або вектором) у N-вимірному просторі. Ідея використання основних компонентів для зображення людських облич була розроблена Сіровічем і Кірбі у 1987 і використана Терком і Пентлендом у 1991, для виявлення та розпізнавання облич.

Багато хто вважає, що підхід Eigenface є першою працюючою технологією розпізнавання обличчя, і він послужив основою для однієї з найкращих комерційних технологій розпізнавання облич. З моменту його початкової розробки

та публікації було багато розширень оригінального методу та багато нових розробок у системах автоматичного розпізнавання обличчя. Власні грані досі часто розглядаються як базовий метод порівняння для демонстрації мінімальної очікуваної продуктивності такої системи.



Fisherfaces. Ключовою проблемою комп'ютерного зору, розпізнавання образів і машинного навчання є визначення відповідного представлення даних для поставленої задачі.



Одним із способів представлення вхідних даних є пошук підпростору, який представляє більшу частину дисперсії даних. Це можна отримати за допомогою аналізу основних компонентів (PCA). При застосуванні до зображень обличчя PCA дає набір власних граней. Ці власні грані є власними векторами, пов'язаними з найбільшими власними значеннями коваріаційної матриці навчальних даних. Знайдені таким чином власні вектори відповідають розв'язку методу найменших квадратів (LS). Це дійсно потужний спосіб представлення даних, оскільки він забезпечує збереження дисперсії даних, усуваючи непотрібні існуючі кореляції між вихідними ознаками (вимірами) у вибіркових векторах.



Коли метою є класифікація, а не представлення, рішення LS може не дати найбільш бажаних результатів. У таких випадках потрібно знайти підпростір, який відображає вибіркові вектори одного класу в одній точці представлення об'єктів і векторів різних класів якомога далі один від одного. Методи, розроблені для досягнення цієї мети, відомі як дискримінантний аналіз (DA).



Найвідомішим DA є лінійний дискримінантний аналіз (LDA), який можна вивести з ідеї, запропонованої Р.А. Фішером у 1936 році. Коли LDA використовується для пошуку представлення підпростору набору зображень обличчя, отримані базисні вектори, що визначають цей простір, відомі як Fisherfaces.



Масштабно інваріантне перетворення ознак (SIFT) — це дескриптор зображення для зіставлення та розпізнавання на основі зображень, розроблений

Девідом Лоу (1999, 2004). Цей дескриптор, а також пов'язані з ним дескриптори зображень використовуються для багатьох цілей комп'ютерного зору, пов'язаних із узгодженням точок між різними видами тривимірної сцени та розпізнаванням об'єктів на основі перегляду. Дескриптор SIFT є інваріантним до трансляцій, поворотів і перетворень масштабування в області зображення і є надійним або помірним перетворенням перспективи та змінами освітлення. Експериментально було доведено, що дескриптор SIFT дуже корисний на практиці для зіставлення зображень та розпізнавання об'єктів у реальних умовах.

Метод прискорення надійних функцій (SURF) — це швидкий і надійний алгоритм локального, інваріантного подібності представлення та порівняння зображень. Основний інтерес підходу SURF полягає в його швидкому обчисленні операторів за допомогою коробкових фільтрів, що дозволяє таким додаткам у реальному часі, як відстеження та розпізнавання об'єктів.

Гістограми локальних бінарних шаблонів (LBPH) використовує локальні двійкові шаблони (LBP), простий, ефективний оператор текстури в комп'ютерному баченні, який позначає пікселі на зображенні, встановлюючи поріг сусідства кожного пікселя та розглядаючи результат як двійкове число. На етапі навчання алгоритм LBPH створює гістограми для кожного зображення, яке позначено та класифіковано. Кожна гістограма представляє кожне зображення з навчального набору. Таким чином, фактичний процес розпізнавання передбачає порівняння гістограм будь-яких двох зображень.

Для розробки програмного забезпечення обрано гістограму локальних бінарних шаблонів (LBPH)

### 2.1.4 Гістограма локальних бінарних шаблонів (LBPH)

Локальний двійковий шаблон (LBP) — це простий, але дуже ефективний оператор текстури, який позначає пікселі зображення, обмежуючи оточення кожного пікселя, і розглядає результат як двійкове число.

Вперше він був описаний у 1994 році (LBP) і з тих пір було визнано потужною функцією для класифікації текстур. Крім того, було визначено, що коли

LBP поєднується з дескриптором гістограм орієнтованих градієнтів (HOG), це значно покращує ефективність виявлення деяких наборів даних.

Використовуючи LBP у поєднанні з гістограмами, ми можемо представити зображення обличчя за допомогою простого вектора даних.

Оскільки LBP є візуальним дескриптором, його також можна використовувати для завдань розпізнавання обличчя.

Кроки алгоритму:

Параметри: LBPH використовує 4 параметри:

– Радіус: радіус використовується для побудови кругового локального двійкового шаблону і представляє радіус навколо центрального пікселя. Зазвичай встановлюється на 1;

– Сусіди: кількість точок вибірки для побудови кругового локального двійкового шаблону. Майте на увазі: чим більше точок вибірки ви включите, тим вищі обчислювальні витрати. Зазвичай встановлюється на 8;

– Сітка X: кількість осередків у горизонтальному напрямку. Чим більше комірок, тим дрібніше сітка, тим вища розмірність результуючого вектора ознак. Зазвичай встановлюється на 8;

– Сітка Y: кількість осередків у вертикальному напрямку. Чим більше комірок, тим дрібніше сітка, тим вища розмірність результуючого вектора ознак. Зазвичай встановлюється на 8;

Навчання алгоритму: По-перше, нам потрібно навчити алгоритм. Для цього нам потрібно використовувати набір даних із зображеннями облич людей, яких ми хочемо впізнати. Нам також потрібно встановити ідентифікатор (це може бути число або ім’я людини) для кожного зображення, тому алгоритм використовуватиме цю інформацію, щоб розпізнати вхідне зображення та надати вам вихід. Зображення однієї і тієї ж особи повинні мати однаковий ідентифікатор.

Застосування операції LBP: Перший обчислювальний крок LBPH полягає у створенні проміжного зображення, яке краще описує вихідне зображення шляхом виділення характеристик обличчя. Для цього в алгоритмі використовується

концепція розсувного вікна, заснована на параметрах радіуса і сусідів. Приклад цього кроку на рис. 2.7.

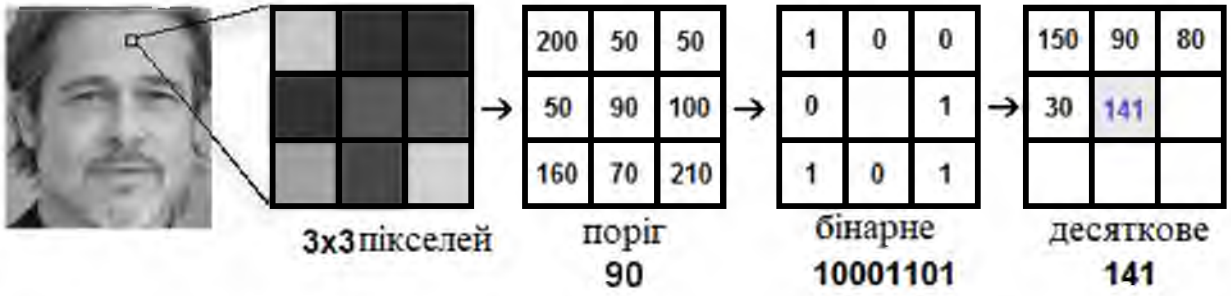



Рис. 2.7. Застосування операції LBP

Вилучення гістограм: тепер, використовуючи зображення, згенероване на останньому кроці, ми можемо використовувати параметри Grid X і Grid Y, щоб розділити зображення на кілька сіток.

На основі зображення вище ми можемо витягти гістограму кожного регіону таким чином:

– Оскільки у нас є зображення у відтінках сірого, кожна гістограма (з кожної сітки) міститиме лише 256 позицій (0~255), що представляють інтенсивність кожного пікселя;

– Потім нам потрібно об'єднати кожну гістограму, щоб створити нову і більшу гістограму. Припустимо, що у нас є сітки 8x8, ми матимемо 8x8x256=16,384 позиції на кінцевій гістограмі. Остаточна гістограма представляє характеристики вихідного зображення зображення.

Виконання розпізнавання обличчя: на цьому кроці алгоритм уже навчений. Кожна створена гістограма використовується для представлення кожного зображення з навчального набору даних. Отже, надавши вхідне зображення, ми знову виконуємо кроки для цього нового зображення та створюємо гістограму, яка представляє зображення.

Висновки:

- LBPH є одним із найпростіших алгоритмів розпізнавання обличчя;
- Він може представляти місцеві особливості на зображеннях;
- Можна отримати чудові результати;
- Він стійкий до монотонних перетворень відтінків сірого;
- Він надається бібліотекою OpenCV (Open Source Computer Vision Library);

## 2.2 Моделювання поведінки системи

Розглянемо поведінку системи розпізнавання осіб як набір функцій та акторів, що з ними взаємодіють. Модель поведінки системи розпізнавання осіб подано в нотації UML, діаграмою прецедентів, що описує доступні дії та випадки рис. 2.8.

Для предметної області виділено наступних акторів, що зображено в табл. 2.1:

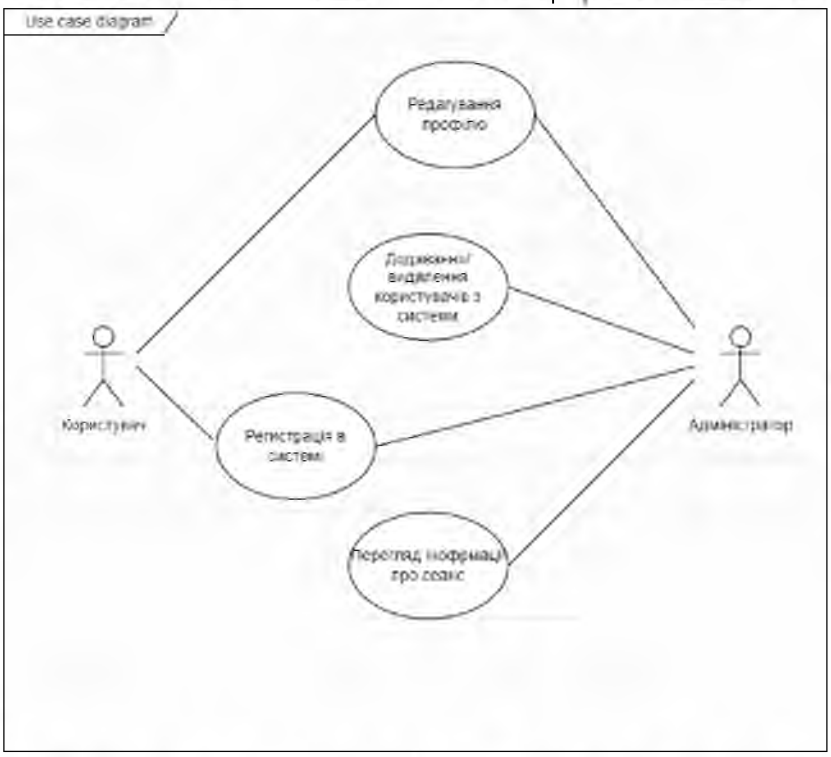



Рис. 2.8. Діаграма прецедентів системи розпізнавання осіб

Таблиця 2.1

Визначені актори

| Актор | Короткий опис |
|---|---|
| Адміністратор | Співробітник, який має можливість керувати усіма основними процесами системи |
| Користувач | Співробітник, який має можливість редагувати свої дані |

Розглянемо тепер, які можливості має надавати наша система:

– актор Адміністратор використовує систему для редагування даних користувачів, створення та видалення користувачів, перегляду перегляду зареєстрованих користувачів;

– актор Користувач використовує систему для редагування своїх даних та реєстрації.

На підставі вишевикладеного можна виділити наступні факти, що зображені в таблиці 2.2:

Таблиця 2.2

Опис прецедентів

| Прецедент | Короткий опис |
|---|---|
| Редагування профілю | Запускається Адміном, Користувачем. Дозволяє редагувати адміну редагувати наявні профілі в системі. Користувач має можливість редагувати тільки свій профіль. |
| Додавання/видалення користувачів з системи | Запускається Адміном. Дозволяє додати або видалити користувача з системи. |
| Регистрація в системі | Запускається Адміном та Користувачем. Дозволяю адміну реєструвати нових користувачів. Дозволяє користувачу зареєструвати новий профіль для себе |
| Перегляд інформації про сеанс | Запускається Адміном. Дозволяє адміну переглядати зареєстрованих користувачів в системі. |

Алгоритм для розпізнавання обличчя та збереження зображень. Спочатку намагається знайти обличчя на відео, наступним кроком перевіряє чи воно є в кадрі, якщо немає то закінчую роботу. Далі виконуються трансформації зображення обличчя необхідні для тренування алгоритму розпізнавання. В кінці результат зберігається в сховище. Алгоритм на рис. 2.9.


Рис. 2.9. Алгоритм створення набору зображень

Алгоритм створення патернів для методу розпізнавання обличь. Алгоритм перебирає збережені зображення, перетворює їх в необхідний формат та створює файл патерн для розпізнавання рис. 2.10.


Рис. 2.10. Алгоритм створення патернів

Алгоритм розпізнавання облич. Розпізнавання виконується в режимі реального часу, програма приймає потік відео з камери. Обробляє кожний кадр вхідного зображення. Потім порівнює ці кадри з патернами які були створенні з певних облич. В кінці якщо обличчя ідентифіковано, тобто програма розпізнає його, то вона виводить на данні цієї особи. Якщо особа яка розпізнається не може знайтись, то виводиться що особу не опізнано рис. 2.11.

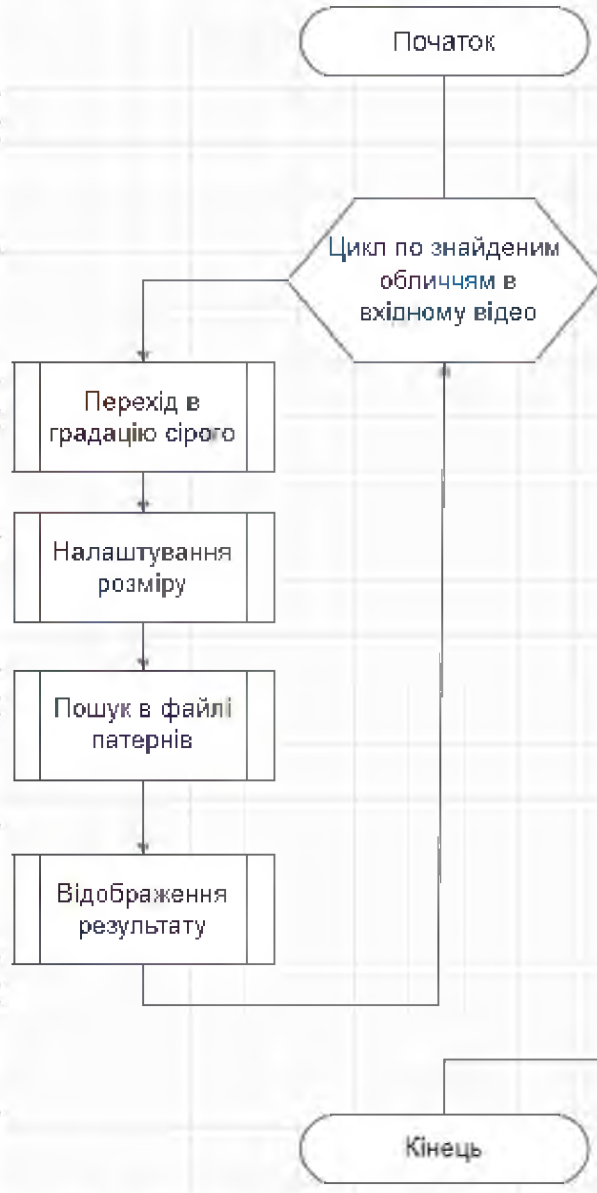



Рис. 2.11. Алгоритм розпізнавання облич

# 3 Реалізація та розробка компонентів системи

## 3.1 Інструментарій розробки

Розробка системи втілена на мові програмування Python в середовищі розробки JetBrains PyCharm.

Python — це інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Його високорівневі вбудовані структури даних у поєднанні з динамічним типізацією та динамічним зв'язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання в якості мови сценаріїв або склеювання для з'єднання існуючих компонентів. Простий, легкий у засвоєнні синтаксис Python підкреслює читабельність і, отже, знижує витрати на обслуговування програми. Python підтримує модулі та пакунки, що сприяє модульності програм і повторному використанню коду. Інтерпретатор Python і велика стандартна бібліотека доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Для прискорення процесу та ефективності реалізації розробки використано бібліотеку OpenCV

OpenCV (Open Source Computer Vision Library) — бібліотека комп'ютерного бачення та машинного навчання з відкритим кодом. OpenCV створено для забезпечення загальної інфраструктури для додатків комп'ютерного зору та для прискорення використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом BSD, OpenCV дозволяє компаніям легко використовувати та змінювати код.

Бібліотека має понд 2500 оптимізованих алгоритмів, які включають повний набір як класичних, так і найсучасніших алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людини на відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення тривимірних хмар точок із стереокамер, з'єднання зображень

разом для отримання високої роздільної здатності. зображення всієї сцени, знаходити подібні зображення з бази даних зображень, видаляти червоні очі із зображень, зроблених за допомогою спалаху, слідкувати за рухами очей, розпізнавати декорації та встановлювати маркери для накладання на них доповненої реальності тощо. OpenCV має понад 47 тисяч користувачів спільноти та оціночна кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується компаніями, науковими групами та державними органами.

Він має інтерфейси C++, Python, Java і MATLAB і підтримує Windows, Linux, Android і Mac OS. OpenCV схиляється здебільшого до програм бачення в реальному часі та використовує переваги інструкцій MMX та SSE, якщо вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які створюють або підтримують ці алгоритми. OpenCV написаний на C++ і має шаблонний інтерфейс, який безперебійно працює з контейнерами STL.

Для створення веб додатку використано веб-фреймворк Django.

Django — це високорівневий веб-фреймворк Python, який заохочує швидкий розвиток і чистий, прагматичний дизайн. Створений досвідченими розробниками, він вирішує велику частину клопоту веб-розробки, тому ви можете зосередитися на написанні свого додатка без необхідності винаходити велосипед. Django безкоштовний та з відкритим вихідним кодом.

### 3.2 Створення налаштування проекту

Створюємо новий проект, вибравши Файл / Новий проект з головного меню

Потім вибираємо потрібний тип проекту, це Django. Вказуємо назву та місце розташування проекту.

Найкращою практикою Python є створення virtualenv для кожного проекту. У більшості випадків PyCharm автоматично створює нове віртуальне середовище, і вам не потрібно нічого налаштовувати. Тим не менш, можливо попередньо переглянути та змінити параметри venv. Розгортаємо вузол Python Interpreter: New

Virtualenv Environment і вибираємо інструмент, який використовується для створення нового віртуального середовища. Вибираємо інструмент Virtualenv і вказуємо розташування середовища та базового інтерпретатора Python, що використовується для нового віртуального середовища.

Далі розгораємо вузол Додаткові параметри та вказуємо параметри, пов'язані з Django. У полі Назва програми вказуємо назву програми рис. 3.1.

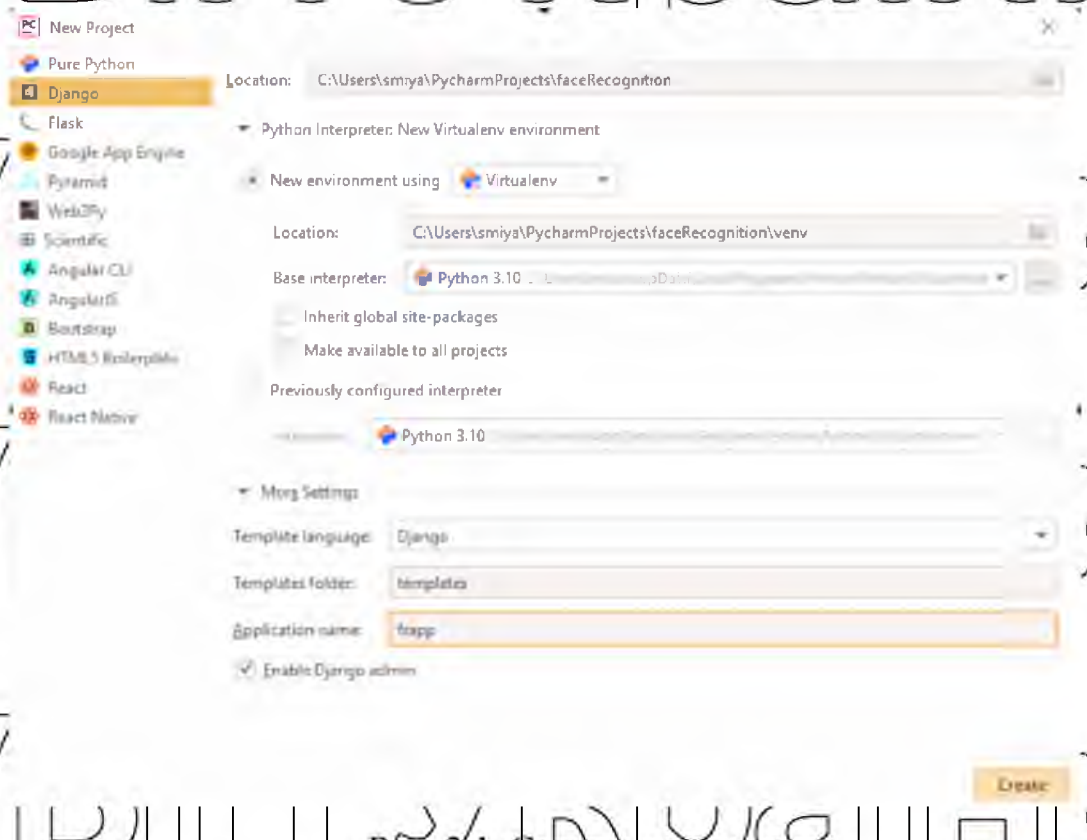



Рис. 3.1. Створення проекту

Створений проект містить файли та каталоги, що стосуються платформи.

Цей вид відображається за замовчуванням. Він показує структуру проекту, специфічну для Django: faceRecognition та frapp каталоги; також ви бачите файли manage.py та settings.py.

Що подано в структурі проекту?

– Каталог faceRecognition є контейнером для вашого проекту. У структурі проекту він позначений жирним шрифтом;

– manage.py: це утиліта командного рядка, яка дозволяє вам взаємодіяти з проектом Django. Додаткову інформацію можна знайти в документації Django;

– Вкладений каталог faceRecognition – це фактичний пакет Python для вашого проекту;

– faceRecognition /__init__.py: Цей порожній файл повідомляє Python, що цей каталог слід вважати пакетом Python;

– faceRecognition /settings.py: цей файл містить конфігурацію для проекту Django;

– faceRecognition /urls.py: цей файл містить оголошення URL для проекту Django;

– faceRecognition /wsgi.py: цей файл визначає точку входу для WSGI-сумісних веб-серверів для обслуговування проекту;

– Вкладений каталог frapp містить усі файли, необхідні для розробки програми Django (на даний момент ці файли порожні) ;

– Знову ж таки, frapp / _init_.py повідомляє Python, що цей каталог слід вважати пакетом Python;

– frapp/models.py: У цьому файлі ми створимо моделі для нашої програми;

– frapp /views.py: у цьому файлі ми створимо подання;

– каталог templates на даний момент порожній. Він повинен містити файли шаблонів;

Вкладений каталог migrations наразі містить лише файл пакету _init_.py, але він буде використовуватися в майбутньому для поширення змін, які ви внесете у свої моделі (додавання поля, видалення моделі тощо) у вашу схему бази даних.

Також зверніть увагу на налаштування INSTALLED_APPS у верхній частині файлу. Це містить імена всіх програм Django, які активовані в цьому екземплярі Django. Програми можна використовувати в кількох проектах, і ви можете упаковувати та розповсюджувати їх для використання іншими у своїх проектах.

Файл faceRecognition/settings.py/INSTALLED_APPS містить такі програми, усі з які постачаються разом із Django. Також він містить нову програму яку ми створили, поки вона нічого не робить, рис. 3.2.

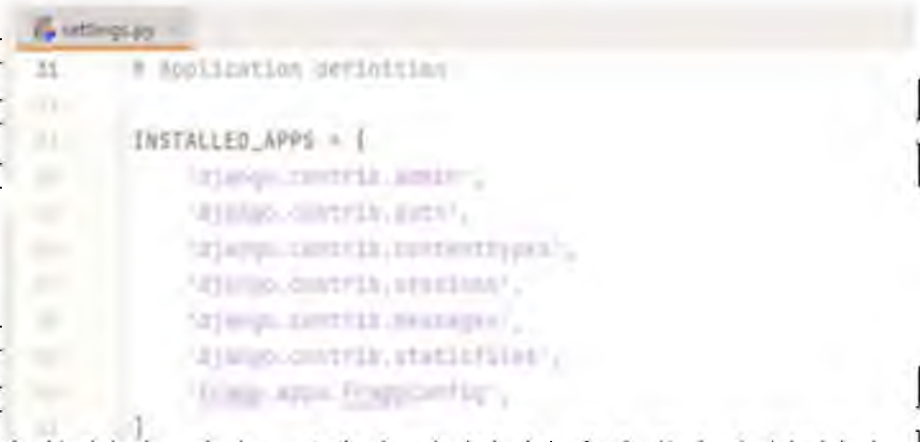



Рис. 3.2. Програми які додані до проекту

Деякі з цих програм використовують принаймні одну таблицю бази даних, тому нам потрібно створити таблиці в базі даних, перш ніж ми зможемо їх використовувати. Для цього треба виконати таку команду як на рис. 3.3


Рис. 3.3. Міграції

Результат виконання команди рис. 3.4.

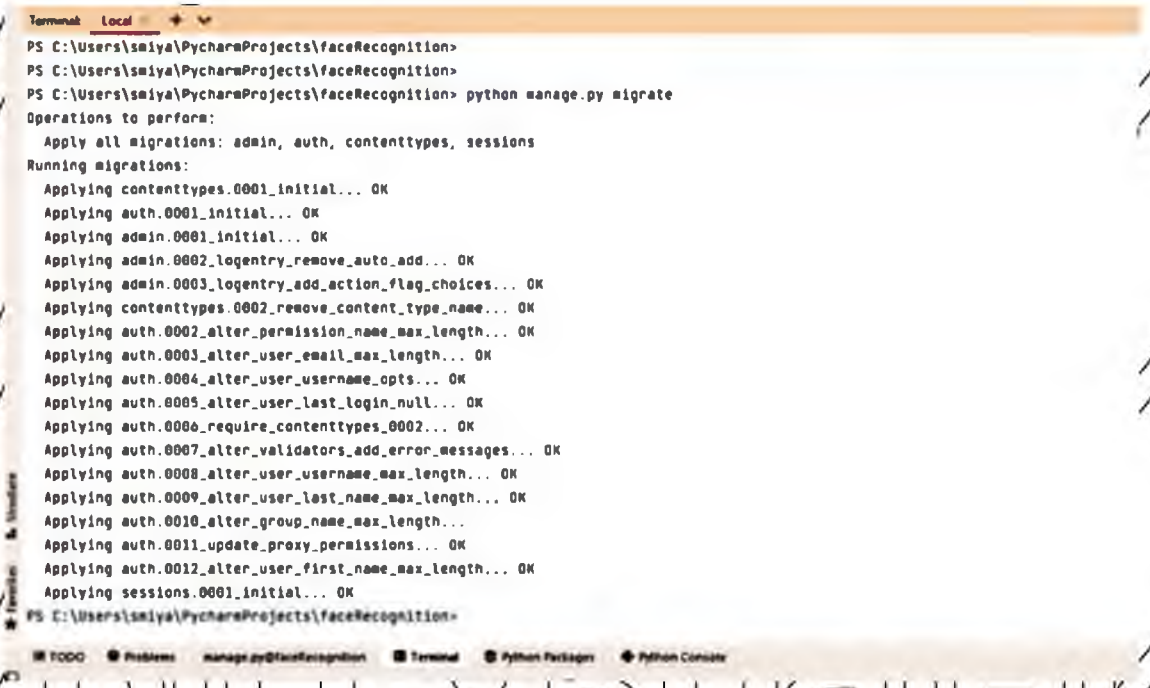



Рис. 3.4. Результат виконання команди рисунок

Тепер в нас є можливість запустити сервер та перевірити чи він працює. Це дасть нам змогу переконатися що всі налаштування виконанні правильно.

Для цього використаємо команду вказану на рис. 3.5

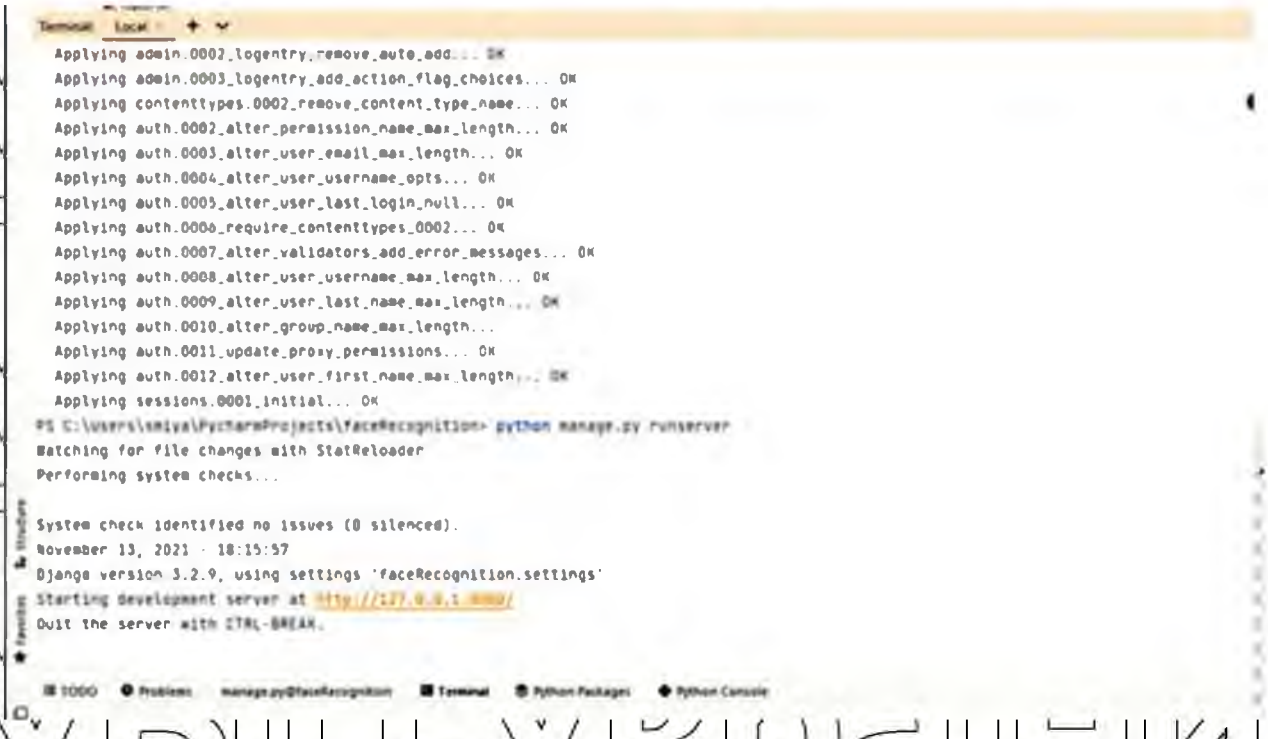



Рис. 3.5. Успішний запуск серверу

Наступним кроком необхідно вказати шлях до файлу з посиланнями нашого додатку. Для цього необхідно відкрити файл urls.py у директорії faceRecognition та додати до нього шлях. Спочатку вказується URLpatern, шлях який буде відображатись URL, потім вказати файл де знаходяться вказівники на ресурси додатка, приклад на рис. 3.6.

```
16  from django.contrib import admin
17  from django.urls import path, include
18
19  urlpatterns = [
20      path('app/', include('frapp.urls')),
21      path('admin/', admin.site.urls),
22  ]
23
```

Рис. 3.6. Конфігурація URL

## 3.3 Розробка програми

Першим кроком потрібно створити три директорії: cascade, dataset, trainer. Вони належать до директорії самого додатку (рис. 3.7)

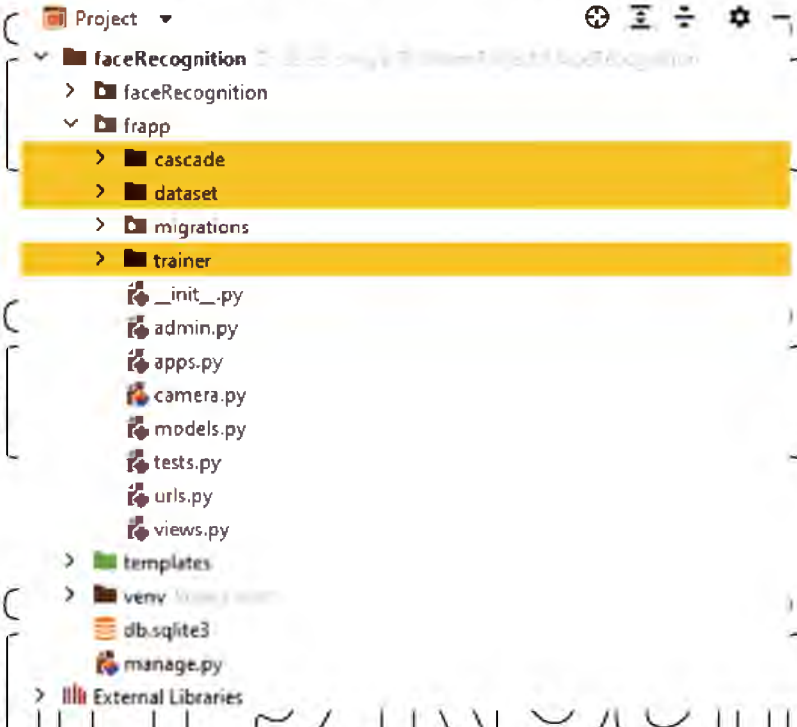



Рис. 3.7. Нові директорії

Директорія cascade зберігатиме Хаарові каскади які надає бібліотека OpenCV

Директорія dataset міститеме зображення для навчання методу розпізнавання. Trainer зберігатиме файл який містить дані про вже навчені обличя.

Необхідно завантажити бібліотеку opencv для нашого проекту, це робить за допомогою pip. Pip — це менеджер пакетів для Python. Це означає, що це інструмент, який дозволяє встановлювати та керувати додатковими бібліотеками та залежностями, які не поширюються як частина стандартної бібліотеки.

Управління пакетами настільки важливо, що pip було включено в інсталятор Python з версій 3.4 для Python 3 і 2.7.9 для Python 2, і він використовується багатьма проектами Python, що робить його важливим інструментом для кожного Pythonista.

Концепція менеджера пакетів може бути знайома вам, якщо ви прийшли з інших мов. JavaScript використовує npm для керування пакетами, Ruby використовує gem, а .NET використовує NuGet. У Python pip став стандартним менеджером пакетів.

Програма встановлення Python встановлює pip, тому він повинен бути готовий до використання, якщо ви не встановили стару версію. Завантаження opencv виконуються нсатупною командою

Лістинг 3.1 – Завантаження opencv

```
pip install opencv-python
```

Створюємо файл camera.py, це файл Python який буде містити основні функції нашою програми рис. 3.8.

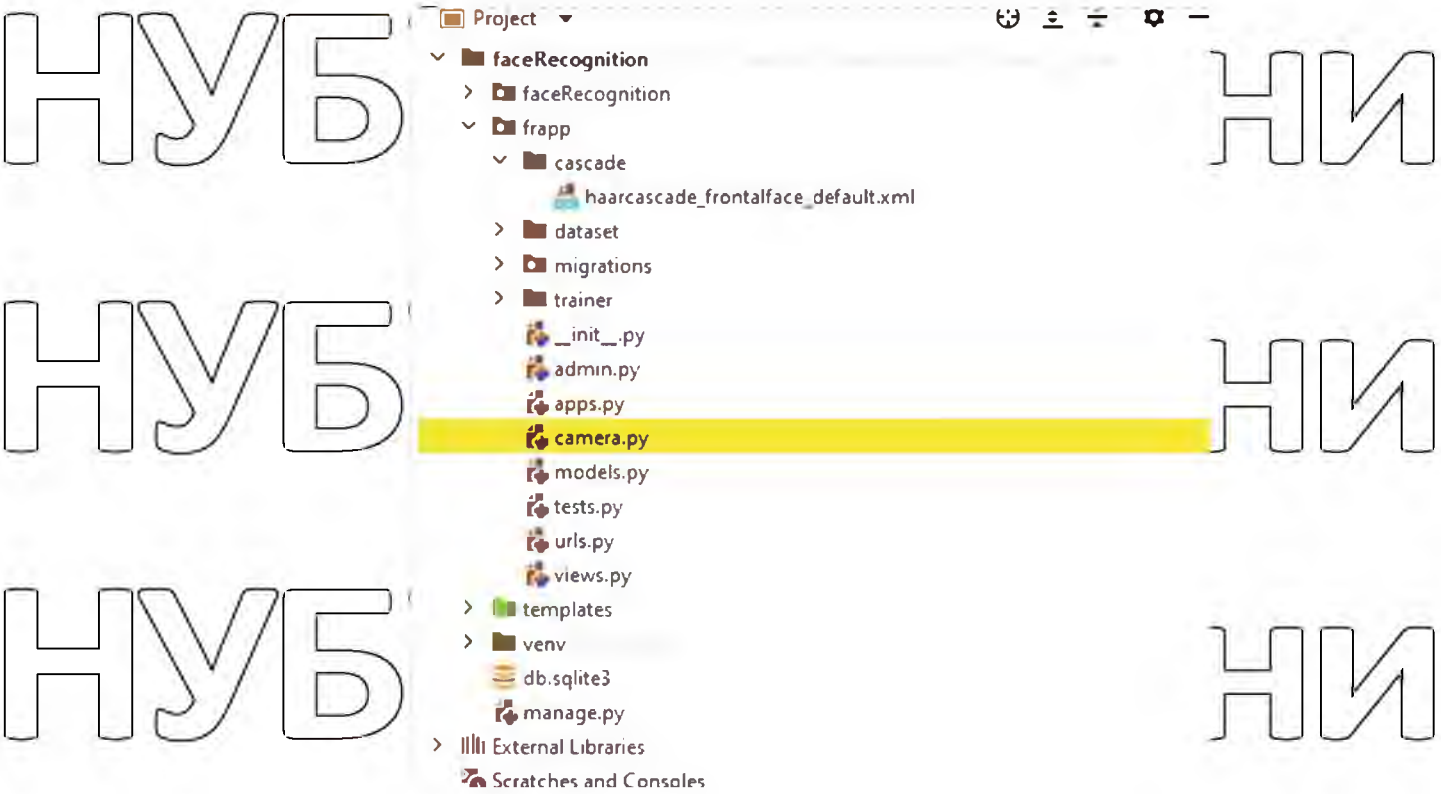



Рис. 3.8. Положення файку camera

Для підключення камери в классі VideoCamera є конструктор. При виклику цього об'єкта він ініціює підключення до ip-камери. В бібліотеці OpenCV є методот який може прийняти відеопотік з різних пристроїв.

Наприклад, якщо потрібно підключити ip-камеру, необхідно передати цьому методу посилання на неї, де вказано логін користувача, пароль, ip адреса та порт.

Якщо потрібно підключити веб камеру, то потрібно просто передати 0 в метод.

Лістинг 3.2 – Функція підключення камери

```
def __init__(self):
    self.video = cv2.VideoCapture('rtsp://username:password@192.168.1.64/1')
```

Наступна функція знаходить обличчя в відеопотоці. Вона бере кожний кадр з відео яке надходить, перетворю зміню гамму кольорів на чорно-білу. Потоім на основі Хаарових каскадів вона викликає метод з бібліотеки detectMultiScale. Він приймає наступні параметри:

– scaleFactor : Параметр, який визначає, наскільки зменшується розмір зображення в кожному масштабі зображення;

– minNeighbors : Параметр, який визначає, скільки сусідів має мати кожен прямокутник-кандидат, щоб його зберегти. Цей параметр вплине на якість виявлених облич: більше значення призводить до меншої кількості виявлення, але з вищою якістю;

– minSize : мінімально можливий розмір об'єкта. Об’єкти, менші за цей, ігноруються;

Потім ми виділяємо знайдене обличчя прямокутником для того щоб побачити результат

Лістинг 3.3 – Функція виведення потоку з камери

```
        def get_frame(self):
            face_cascade                                    =
cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml')
            success, image = self.video.read()
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces       =       face_cascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
            for (x, y, w, h) in faces:
                cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255,
0), 2)
            ret, jpeg = cv2.imencode('.jpg', image)
            return jpeg.tobytes()
```

Функція train_model спочатку приймає значення id та ім’я людини для якої буде створено модель обличчя для розпізнавання.

Потім вона знаходить обличчя в відео потоці та зберігає його в необхідному форматі в директорії dataset. Приклад зібраних зображень чотирьох облич рис. 3.9.

Лістинг 3.3 – Функція яка створює набір зображень

```
        def train_model(self, userid, name):
            face_cascade                                    =
cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml')
```

```
        face_id = userid
        count = 0
        try:
            f = open("frapp/trainer/ref_name.pkl", "rb")
            names = pickle.load(f)
            f.close()
        except:
            names = {}

        names[int(userid)] = name

        while (True):

            success, image = self.video.read()
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.3, 5)
            for (x, y, w, h) in faces:
                cv2.rectangle(image, (x, y), (x + w, y + h), (255,
0, 0), 2)
                count += 1
                cv2.imwrite("frapp/dataset/User." + str(face_id)
+ '.' + str(count) + ".jpg", gray[y:y + h, x:x + w])

            ret, jpeg = cv2.imencode('.jpg', image)
            k = cv2.waitKey(100) & 0xff
            print(count)
            if count >= 3:
                break
        VideoCamera.train(self)
        f = open("frapp/trainer/ref_name.pkl", "wb")
        pickle.dump(names, f)
        f.close()
        print(names)
        return jpeg.tobytes()
```


Рис. 3.9. Зібранні зображення облич

Наступна функція використовує LBPH метод який надаєтьсясамою бібліотекою OpenCv. Вона спочатку перебирає в циклі кожну фотографію та треную LBPH розпізнавач. В кінці своєї роботи створює файл описанням кожнох особи якої розпізнала обличчя.

Лістинг 3.4 – Тренування з допомогою LBPH

```
def train(self):

        print("+++++++++++++++++++++++++++++++++++")
        path = 'frapp/dataset'
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        detector =
cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml');

        def getImagesAndLabels(path):
```

```
        imagepaths = [os.path.join(path, f) for f in
os.listdir(path)]
        facesamples = []
        ids = []

        for imagePath in imagepaths:

            PIL_img = Image.open(imagePath).convert('L')
            img_numpy = np.array(PIL_img, 'uint8')

            id = int(os.path.split(imagePath)[-
1].split(".")[1])
            faces = detector.detectMultiScale(img_numpy)

            for (x, y, w, h) in faces:
                facesamples.append(img_numpy[y:y + h, x:x +
w])

                ids.append(id)

        return facesamples, ids

    print("\n [INFO] Training faces. It will take a few
seconds. Wait ...")
    faces, ids = getImagesAndLabels(path)
    recognizer.train(faces, np.array(ids))

    recognizer.write('frapp\trainer/trainer.yml')

    print("\n [INFO] {0} faces trained. Exiting
Program".format(len(np.unique(ids))))
```

Останній метод в класі який використовується безпосередньо для розпізнавання обличчя в реальному часі. Він працює наступним чином.

Спочатку він знаходить обличчя на відео, потім бере кожний та перетворює його в чорно-білий кадр. Далі наданим методом бібліотекою який використовую LBPH підхід для розпізнавання, починається перевірка та розпізнавання обличчя в кадрі. Мето вже навчений набором зображень, тож він id обличчя яка розпізнає і тільки якщо розпізнає, та відсоток впевненості. Як зрозуміло з назви, це відсоток наскільки алгоритм впевнений що в кадрі якась певна особа.

Лістинг 3.6 – Розпізнавання обличчя

```
def recognize_face(self):
```

```
face_cascade =
cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('frapp/trainer/trainer.yml')
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
try:
    f = open("frapp/trainer/ref_name.pkl", "rb")
    names = pickle.load(f)
    f.close()
except:
    names = {}

success, image = self.video.read()
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.2, minNeighbors=5,
                                      minSize=(30, 30),
)
for (x, y, w, h) in faces:

    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0),2)
id, confidence = recognizer.predict(gray[y:y + h, x:x + w])

    if (confidence < 100):
        id = names[id]
        confidence = "  {0}%".format(round(100 -
confidence))
    else:
        id = "unknown"
        confidence = "  {0}%".format(round(100 -
confidence))

    cv2.putText(image, str(id), (x + 5, y - 5), font, 1,
(255, 255, 255), 2)
    cv2.putText(image, str(confidence), (x + 5, y + h -
5), font, 1, (255, 255, 0), 1)
    ret, jpeg = cv2.imencode('.jpg', image)  return jpeg.tobytes()
```

### 3.4 Розробка сайту та тестування

Для розробки веб-сторінок було використано інструменти які надає Django.

Django — це широко використовуваний безкоштовний фреймворк веб-розробки з відкритим вихідним кодом високого рівня. Він надає розробникам безліч функцій «з коробки», тому розробка може бути швидкою. Веб-сайти, створені на його основі, захищені, масштабовані та підтримуються одночасно.

Найпоширеніший підхід для створення динамічного контенту на сторінці грунтується на шаблонах. Шаблон містить статичні частини бажаного виведення HTML, а також деякий спеціальний синтаксис, який описує, як буде вставлено динамічний контент.

Фреймворк має вбудований шаблонізатор.

Мова шаблонів Django (DTL) – це власна система шаблонів Django. До Django 1.8 це була єдина альтернатива. Це хороша система шаблонів, але із своїми особливостями. Якщо у вас немає причин використовувати іншу систему шаблон, використовуйте вбудовану, особливо, якщо розробляєте розповсюджувану програму, що містить шаблони.

Шаблон Django - це текстовий документ або рядок Python, розмічений за допомогою мови шаблонів Django. Деякі конструкції розпізнаються та інтерпретуються механізмом шаблонів. Основні з них - змінні та теги.

Шаблон відображається у контексті. Візуалізація замінює змінні значеннями, які переглядаються в контексті, і виконує теги. Решта виводиться як є.

Приклад сторінки для реєстрації подано в лістингу 3.7.

Лістинг 3.7 – Сторінка реєстрації

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Video Live Stream</title>
</head>
<body>
<h1> Video Live Stream </h1>
<img src="{% url 'video_feed' %}">
<br>
<h1>Registration</h1>
<form action="{% url 'train' %}" method='GET'>
    <input id="userid" type="text" name="userid">
    <label for="userid" >Input id</label>

    <input id="name" type="text" name="name">
    <label for="name" >name id</label>
    <button type='submit'>Submit</button>
</form>
<form action="{% url 'recognition' %}" method='GET'>
    <button type='submit'>Recognize</button>
</form>
</body>
```

```
</html>
```

На сторінці виводиться відео з камери і на ньому обличчя знаходиться та виділяється квадратом. Є форма для заповнення даних ідентифікатор та ім'я. Кнопка підтвердження реєстрації та кнопка для відображення сторінки з розпізнаванням облич осіб які вже зареєстровані рис. 3.10.

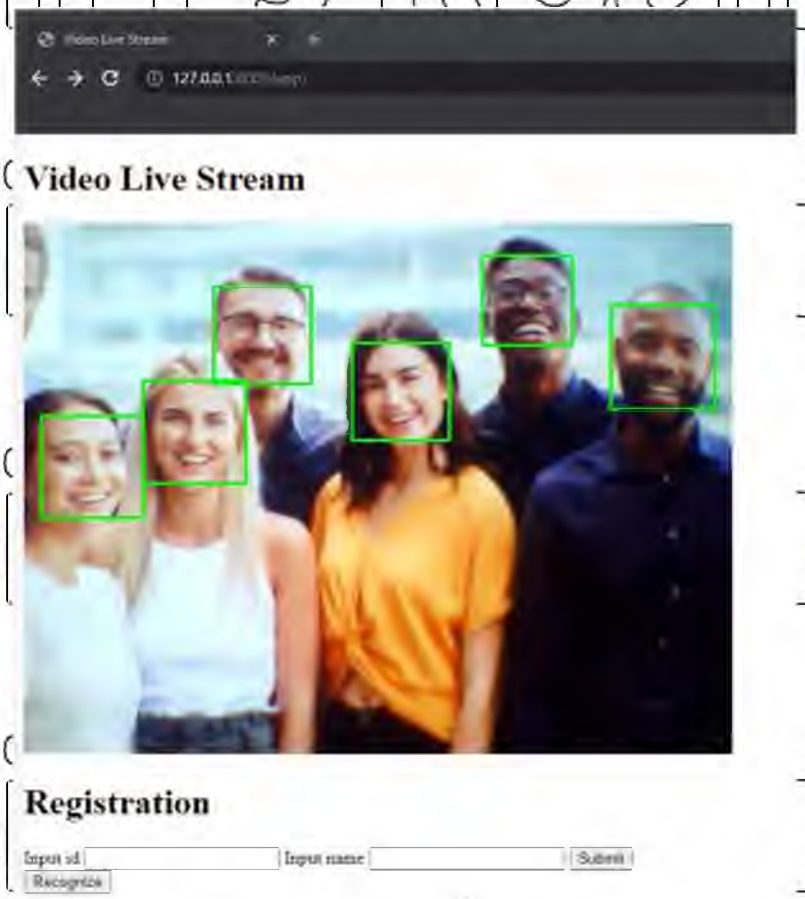



Рис. 3.10. Сторінка реєстрації

Наступний листинг відображає код сторінки про успішне реєстрування особи в системі та відображає знайдене обличчя та ім'я особи, листинг 3.8.

Лістинг 3.8 – Сторінка тренування системи

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Training</title>
</head>
```

```
<body>
<h1> Training </h1>
<img src = "{% url 'take_image' %}">
<br>
{{ name }}

<form action="{% url 'recognition' %}" method='GET'>
    <button type='submit'>Recognize</button>
</form>
</body>
</html>
```

На рис. 3.11 відображено що система успішно зареєструвала нове обличчя особи. Відображено фото та ім'я. Також сторінка має кнопку яка адресує на сторінку для відображення відео з розпізнаванням облич осіб які вже зареєстровані.


Рис. 3.11 Тренування системи

Лістинг 3.9 мітить код для сторінки з розпізнаванням облич.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Recognition</title>
</head>
<body>
```

```
<h1> Recognition </h1>
<img src = "{% url 'recognize' %}">
<br>
<a href="{% url 'index' %}">Home</a>
</body>
</html>
```

На рис. 3.12 показано як система розпізнає осіб шо зареєстровані.

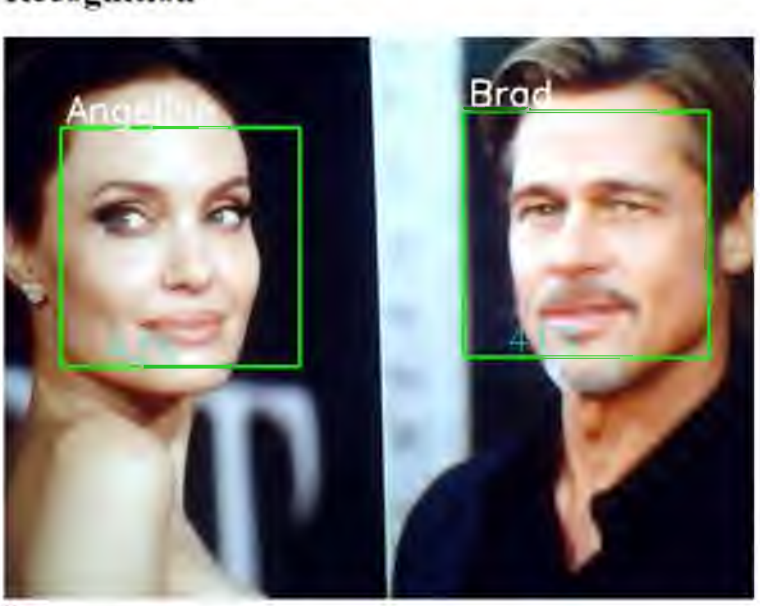



Рис. 3.12. Розпізнавання обличч

Якщо в кадр потрапляє обличчя якого не немає в системі, то воно підписується як «unknown - невідомий»

Шаблони та основний код програми з'єднується з допомогою views або представлення.

Функція представлення, або коротко представлення – це функція Python, яка приймає Web-запит і повертає Web-відповідь. Відповіддю може бути HTML-вміст сторінки, або перенаправлення, або 404 помилка, або XML-документ, або

зображення, що завгодно. Представлення містить усю необхідну логіку для створення відповіді. Цей код може знаходитися будь-де, головне, щоб він знаходився в PYTHON_PATH. Жодних інших вимог немає – жодної “магії”. Незважаючи на можливість розмістити код представлень будь-де, прийнято тримати його у файлі views.py, який знаходиться в каталозі проекту або додатку.

Поданий код в лістингу 3.10 відповідає за відображення сторінок в веб-додатку. Функції index та recognition тільки відображають сторінки, головну де відбувається реєстрація і сторінку з розпізнаванням облич. Функція train відображає сторінку з успішним навчанням системи новому обличчю та приймає данні особи яка реєструється, дані це ідентифікатор та ім’я.

Лістинг 3.10 – Функції для відтворення веб-сторінок

```
def index(request):
    return render(request, 'home.html')

def train(request):
    context = {}
    context['userid'] = request.GET['userid']
    context['name'] = request.GET['name']
    global idface
    global name
    idface = request.GET['userid']
    name = request.GET['name']
    return render(request, 'train.html', context)

def recognition(request):
    return render(request, 'recognition.html')
```

Методи в лістингу 3.11 відповідають за відображення відео кмери з знаходженням обличчя на сторінці реєстрації.

Лістинг 3.11 – Методи для сторінка реєстрації

```
def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n\r\n')

def video_feed(request):
    return StreamingHttpResponse(gen(VideoCamera()),
                                 content_type='multipart/x-
mixed-replace; boundary=frame')
```

Методи в лістингу 3.12 відповідають за відображення результату про успішну реєстрацію.

Лістинг 3.12 – Методи для сторінка підтвердження реєстрації

```
        def genmodel(camera):
            frame = camera.train_model(idface, name)
            yield (b'--frame\r\n'
                   b'Content-Type:  image/jpeg\r\n\r\n'  +  frame  +
b'\r\n\r\n')

        def take_image(request):
            return StreamingHttpResponse(genmodel(VideoCamera()),
                                         content_type='multipart/x-
mixed-replace; boundary=frame')
```

Методи в лістингу 3.13 відповідають за відображення відео камери на якому відбувається розпізнавання облич.

Лістинг 3.13 – Методи для сторінки з розпізнаванням облич

```
        def genrecognize(camera):
            while True:
                frame = camera.recognize_face()
                yield (b'--frame\r\n'
                       b'Content-Type:  image/jpeg\r\n\r\n'  +  frame  +
b'\r\n\r\n')

        def recognize(request):
            return StreamingHttpResponse(genrecognize(VideoCamera()),
                                         content_type='multipart/x-
mixed-replace; boundary=frame')
```

# ВИСНОВКИ

У результаті виконання магістерської роботи було спроектовано та розроблено систему розпізнавання осіб.

У проекті вирішено наступні задачі:

1. Проведено аналіз та опис предметної області програмного продукту. Проаналізовано усі існуючі подібні системи та виявлено сильні та слабкі сторони. Описано основні функції програмного продукту.

2. Виділено основні бізнес вимоги, функціональні вимоги та нефункціональні вимоги до системи. Підібрано мову програмування, яка забезпечить високу якість системи.

3. Спроектовано систему, розроблено ієрархію проекту та проведено опис діаграм та основних алгоритмів роботи.

4. Описано кроки реалізації, наведено код системи з коментарями до нього, описано основні архітектурні рішення при використанні фреймворку Django.

5. Проведено тестування системи. Тестування системи показує, що вона відповідає вимогам, які були створені для неї.

Аналіз системи показав, що створений проект є конкурентоспроможним.

У ході роботи досліджено можливості використання сучасних технологій для розробки програми для розпізнавання осіб з використанням комп'ютерного зору. Було проведено аналіз роботи програми та розроблено програмний продукт що дозволяє розпізнавати обличчя в реальному часі.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. OpenCV Library. URL: https://opencv.org/ (дата звернення: 20.05.2021).
2. OpenCV Library документація. https://docs.opencv.org/ (дата звернення: 20.05.2021).
3. Python документація. URL: https://docs.python.org/3/ (дата звернення: 20.05.2021).
4. Django документація. URL: https://docs.djangoproject.com/en/4.0/ (дата звернення: 20.05.2021).
5. Аллен Б. Дауни, Think Python: How to Think Like a Computer Scientist, 2nd edition
6. Створення шаблонів Django. URL: https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-django-project.html#create-templates (дата звернення: 20.05.2021).
7. Історія комп'ютерного зору URL: https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3 (дата звернення: 20.05.2021).
8. Принцип роботи комп'ютерного зору URL: https://indatalabs.com/blog/how-does-computer-vision-work (дата звернення: 20.05.2021).
9. IP camera URL: https://en.wikipedia.org/wiki/IP_camera (дата звернення: 20.05.2021).
10. Хаарові каскади URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (дата звернення: 20.05.2021).

**Додаток А.1**

Компонент «camera.py»

```
import cv2.cv2 as cv2
import cv2
import os
from PIL import Image
import numpy as np
import pickle

class VideoCamera(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)

    def __del__(self):
        self.video.release()

    def stopf(self):
        return self.video.release()

    def get_frame(self):
        face_cascade =
cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml')
        success, image = self.video.read()
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))
        for (x, y, w, h) in faces:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255,
0), 2)
        ret, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tobytes()
    def train_model(self, userid, name):
        face_cascade =
```

```
        cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
.xml')
        face_id = userid
        count = 0
        try:
            f = open("frapp/trainer/ref_name.pkl", "rb")
            names = pickle.load(f)
            f.close()
        except:
            names = {}

        names[int(userid)] = name

        while (True):
            success, image = self.video.read()
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.3, 5)
            for (x, y, w, h) in faces:
                cv2.rectangle(image, (x, y), (x + w, y + h), (255,
0, 0), 2)
                count += 1
                cv2.imwrite("frapp/dataset/User." + str(face_id) +
'.' + str(count) + ".jpg", gray[y:y + h, x:x + w])
                ret, jpeg = cv2.imencode('.jpg', image)
                k = cv2.waitKey(100) & 0xff
                print(count)
                if count >= 3:
                    break
        VideoCamera.train(self)
        f = open("frapp/trainer/ref_name.pkl", "wb")
        pickle.dump(names, f)
        f.close()
        print(names)
        return jpeg.tobytes()
    def train(self):
```

```
            print("++++++++++++++++++++++++++++++++")
            path = 'frapp/dataset'
            recognizer = cv2.face.LBPHFaceRecognizer_create()
            detector =
        cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
        .xml');

            def getImagesAndLabels(path):
                imagepaths = [os.path.join(path, f) for f in
        os.listdir(path)]
                facesamples = []
                ids = []
                for imagePath in imagepaths:
                    PIL_img = Image.open(imagePath).convert('L')
                    img_numpy = np.array(PIL_img, 'uint8')
                    id = int(os.path.split(imagePath)[-1].split(".")[1])
                    faces = detector.detectMultiScale(img_numpy)
                    for (x, y, w, h) in faces:
                        facesamples.append(img_numpy[y:y + h, x:x + w])
                        ids.append(id)
                return facesamples, ids
            print("\n [INFO] Training faces. It will take a few seconds.
        Wait ...")
            faces, ids = getImagesAndLabels(path)
            recognizer.train(faces, np.array(ids))
            recognizer.write('frapp/trainer/trainer.yml')
            print("\n [INFO] {0} faces trained. Exiting
        Program".format(len(np.unique(ids))))
        def recognize_face(self):
            face_cascade =
        cv2.CascadeClassifier('frapp/cascade/haarcascade_frontalface_default
        .xml')
            recognizer = cv2.face.LBPHFaceRecognizer_create()
            recognizer.read('frapp/trainer/trainer.yml')
            font = cv2.FONT_HERSHEY_SIMPLEX
```

```
        id = 0
        try:
            f = open("frapp/trainer/ref_name.pkl", "rb")
            names = pickle.load(f)
            f.close()
        except:
            names = {}
        success, image = self.video.read()
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2,
minNeighbors=5,
                                              minSize=(30, 30), )
        for (x, y, w, h) in faces:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255,
0), 2)
            id, confidence = recognizer.predict(gray[y:y + h, x:x +
w])
            print(confidence)
            if (confidence < 60):
                name = names[id]
                confidence = "  {0}%".format(round(100 -
confidence))
            else:
                name = "unknown"
                confidence = "  {0}%".format(round(100 -
confidence))
            cv2.putText(image, str(name), (x + 5, y - 5), font, 1,
(255, 255, 255), 2)
            cv2.putText(image, str(confidence), (x + 5, y + h - 5),
font, 1, (255, 255, 0), 1)
            ret, jpeg = cv2.imencode('.jpg', image)
            return jpeg.tobytes()
```

**Додаток А.2**

Компонент «views.py»

```
def index(request):
    return render(request, 'home.html')
#idface = 0
def train(request):
    context = {}
    context['userid'] = request.GET['userid']
    context['name'] = request.GET['name']
    global idface
    global name
    idface = request.GET['userid']
    name = request.GET['name']
    return render(request, 'train.html', context)
def recognition(request):
    return render(request, 'recognition.html')
def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n\r\n')
def video_feed(request):
    return StreamingHttpResponse(gen(VideoCamera()),
                                 content_type='multipart/x-mixed-
replace; boundary=frame')
def genmodel(camera):
    frame = camera.train_model(idface, name)
    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n\r\n')
def take_image(request):
    return StreamingHttpResponse(genmodel(VideoCamera()),
                                 content_type='multipart/x-mixed-
replace; boundary=frame')
def genrecognize(camera):
    while True:
        frame = camera.recognize_face()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame +
b'\r\n\r\n')
def recognize(request):
    return StreamingHttpResponse(genrecognize(VideoCamera()),
                                 content_type='multipart/x-mixed-
replace; boundary=frame')
```