МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп’ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

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

_____І. М. Журавська

« _ » _____ 2024 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп’ютеру Raspberry Pi

Спеціальність 123 «Комп'ютерна інженерія»

123 – КМР.ПЗ.00 – 605.21810206

Студент _____С. Ю. Воздіцький

підпис

«__» _____ 202__ р.

Керівник к. т. н., доцент _____В. Ю. Савінов

підпис

«__» _____ 202__ р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної магістерської роботи

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

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

« __ » _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної магістерської роботи

Видано студенту групи _605_ факультету комп'ютерних наук

_____Воздіцькому Сергію Юрійовичу_____
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютеру Raspberry Pi_____
_____

Затверджена наказом по ЧНУ від «08» __листопада__ 2024 р. № __226__

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

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
реалізація програмно-апаратного комплексу детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютеру Raspberry Pi_____
_____

4. Перелік питань, що підлягають розробці _аналіз актуальності обраної теми, розгляд та вивчення загальної теорії, обґрунтування засобів програмної реалізації, програмна реалізація детектування рухів та розпізнавання обличчя, тестування функціональності програмно-апаратного комплексу на базі одноплатного комп'ютеру Raspberry Pi та отримання результатів роботи_____

5. Перелік графічних матеріалів

формули, рисунки, презентація

6. Завдання до спеціальної частини

Забезпечення вимог охорони праці у приміщенні комп'ютерної лабораторії вищого навчального закладу

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Григор’єва Л. І.	Кафедра екології	Спеціальна частина з охорони праці та безпеки у надзвичайних ситуаціях
Савінов В. Ю.	Кафедра комп’ютерної інженерії	Методична частина

Керівник роботи __к. т. н., доцент, Савінов Володимир Юрійович

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

Воздіцький Сергій Юрійович

(прізвище, ім’я, по батькові студента)

_____

(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема: Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютеру Raspberry Pi

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	03.10.2023	10.10.2023	Виконано
2.	Огляд літератури за темою роботи	12.10.2023	19.10.2023	Виконано
3.	Складання календарного плану КМР	20.10.2023	21.10.2023	Виконано
4.	Аналіз предметної області	22.10.2023	29.10.2023	Виконано
5.	Розробка проєктних рішень	30.10.2023	04.11.2023	Виконано
6.	Моделювання та конструювання АПЗ	05.11.2023	19.11.2023	Виконано
7.	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування, розробка керівництва користувача	20.11.2023	27.11.2023	Виконано
8.	Відгук керівника КМР	29.11.2023	30.11.2023	Виконано
9.	Оформлення КМР та презентації	01.12.2023	08.12.2023	Виконано
10.	Попередній захист	31.01.2024	31.01.2024	Виконано
11.	Рецензування	14.02.2024	14.02.2024	Виконано
12.	Завершення оформлення КМР та презентації	16.02.2024	20.02.2024	Виконано
13.	Захист кваліфікаційної роботи	28.02.2024	28.02.2024	Виконано

Розробив студент Воздіцький Сергій Юрійович _____
(*прізвище, ім'я, по батькові)* *(підпис)*
«__» _____ 20__ р.

Керівник роботи к. т. н., доцент Савінов Володимир Юрійович _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи
«Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп’ютеру Raspberry Pi»
Студент 605 гр.: Воздіцький Сергій Юрійович
Керівник: к. т. н, доцент. Савінов В. Ю.

Кваліфікаційна магістерська робота на тему "Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп’ютеру Raspberry Pi" є дуже **актуальною** в контексті зростаючої потреби в системах безпеки, відеоспостереження та автоматизації. Використання одноплатних комп'ютерів, таких як Raspberry Pi, робить ці технології більш доступними та вартісно-ефективними для розробників і дослідників. Розробка програмно-апаратного комплексу для детектування рухів та розпізнавання обличчя відкриває широкі можливості застосування в різних сферах, від безпеки до автоматизованих систем управління. Такі системи стають важливою складовою в розвитку інтернету речей (IoT) та сприяють у вирішенні багатьох завдань зі збереження безпеки та оптимізації процесів у різних областях життя.

Об'єктом дослідження є програмно-апаратний комплекс, спрямований на детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютера Raspberry Pi.

Предмет дослідження охоплює глибоке вивчення, розробку та оптимізацію алгоритмів для точного детектування рухів та розпізнавання обличчя на обмежених ресурсах одноплатного комп'ютера Raspberry Pi. Важливою частиною дослідження є створення оптимальних методів обробки відеоданих, взаємодії з сенсорами та апаратною частиною пристрою, а також розробка зручного інтерфейсу для управління системою.

Мета цієї роботи спрямована на створення програмно-апаратного комплексу з метою виявлення рухів та ідентифікації облич на одноплатному комп'ютері Raspberry Pi.

Це дослідження має **практичне значення** для розвитку вбудованих систем, що використовують відеоаналітику, оскільки його результати можуть значно

покращити ефективність різних систем нагляду, безпеки та автоматизації. Розроблений програмно-апаратний комплекс може знайти широке застосування у сферах відеоспостереження, контролю доступу, відтворення емоцій, та навіть у робототехніці для створення інтерактивних систем. Такі інновації сприяють покращенню якості життя та безпеки в різних сферах людської діяльності, роблячи їх більш ефективними та зручними для користувачів.

Наукова новизна даної роботи полягає в розробці програмно-апаратного комплексу з урахуванням обмежених ресурсів пристрою. Дослідження включає в себе оптимізацію алгоритмів обробки відеоданих, використання сенсорів та апаратної частини пристрою для підвищення ефективності роботи системи. Такий підхід відкриває нові можливості в області вбудованих систем та відеоаналітики, а також віддзеркалює потенціал для подальших досліджень у сфері розпізнавання образів та автоматизації процесів з розумінням оточуючого середовища.

Дана робота складається з 4 розділів. Кожний розділ присвячений: аналізу предметної області, математичним моделям і методам, використаним у магістерській роботі, розробці і візуалізації системи, аналізу отриманих результатів, охороні праці та методичній частині магістерської роботи. Загальний обсяг роботи – 80 сторінок (без додатків). Кваліфікаційна магістерська робота містить 3 додатки, 3 формули, 37 рисунків, посилання на 33 літературних джерела.

Ключові слова: програмно-апаратний комплекс, детектування рухів, розпізнавання обличчя, одноплатний комп'ютер Raspberry Pi, оптимізація алгоритмів, апаратна частина, сенсори, інтерфейс користувача, вбудовані системи, відеоаналітика, точність розпізнавання, ефективність роботи системи.

ABSTRACT

to the qualifying master's thesis
"Software and hardware complex of motion detection and face recognition
based on a Raspberry Pi single-board computer"
Student 605 gr.: Serhii Vozditskyi
Supervisor: candidate technical sciences, associate professor.
V. Savinov

The qualifying master's thesis on the topic "Software and hardware complex of motion detection and face recognition based on a Raspberry Pi single-board computer" is very relevant in the context of the growing need for security, video surveillance and automation systems. The use of single-board computers such as the Raspberry Pi makes these technologies more accessible and cost-effective for developers and researchers. The development of a software-hardware complex for motion detection and face recognition opens up wide application possibilities in various fields, from security to automated control systems. Such systems are becoming an important component in the development of the Internet of Things (IoT) and contribute to the solution of many tasks of maintaining security and optimizing processes in various areas of life.

The object of the research is a software-hardware complex aimed at detecting movements and recognizing faces in real time based on a Raspberry Pi single-board computer.

The research subject covers the in-depth study, development and optimization of algorithms for accurate motion detection and face recognition on the limited resources of a Raspberry Pi single-board computer. An important part of the research is the creation of optimal methods of processing video data, interaction with sensors and the hardware part of the device, as well as the development of a convenient interface for managing the system.

The purpose of this work is aimed at creating a software-hardware complex for the purpose of motion detection and face identification on a Raspberry Pi single-board computer.

This research has practical implications for the development of embedded systems using video analytics, as its results can significantly improve the performance of various

surveillance, security and automation systems. The developed software and hardware complex can be widely used in the fields of video surveillance, access control, reproduction of emotions, and even in robotics for creating interactive systems. Such innovations contribute to improving the quality of life and safety in various areas of human activity, making them more efficient and convenient for users.

The scientific novelty of this work consists in the development of a hardware and software complex taking into account the limited resources of the device. The research includes the optimization of video data processing algorithms, the use of sensors and the hardware part of the device to increase the efficiency of the system. This approach opens up new opportunities in the field of embedded systems and video analytics, and also reflects the potential for further research in the field of pattern recognition and automation of processes with an understanding of the surrounding environment.

This work consists of 4 chapters. Each section is devoted to: analysis of the subject area, mathematical models and methods used in the master's work, development and visualization of the system, analysis of the obtained results, labor protection and methodical part of the master's work. The total volume of work is 80 pages (without appendix). The qualifying master's thesis contains 3 appendix, 3 formulas, 37 figures, references to 33 literary sources.

Keywords: hardware and software complex, motion detection, face recognition, Raspberry Pi single-board computer, algorithm optimization, hardware, sensors, user interface, embedded systems, video analytics, recognition accuracy, system efficiency.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ШІ	–	Штучний інтелект
ЕОМ	–	Електронна обчислювальна машина
БД	–	База даних
CNN	–	Convolutional Neural Network
PCA	–	Principal Component Analysis
LBP	–	Local Binary Patterns
LSTM	–	Long Short-Term Memory
R-CNN	–	Region-Based Convolutional Neural Network
R-FCN	–	Region-Based Fully Convolutional Network
SSD	–	Single Shot Detector
SPP-net	–	Spatial Pyramid Pooling network
YOLO	–	You Only Look Once
mAP	–	Mean Average Precision
FCN	–	Fully Convolutional Network
ROI	–	Region of Interest

ВСТУП

В сучасному інформаційному суспільстві швидкість розвитку технологій зростає експоненційно, вносячи вагомий внесок у велику кількість галузей, включаючи обробку зображень та розпізнавання обличчя. Завдяки зростанню обчислювальних можливостей та доступності недорогих, але потужних пристроїв, таких як одноплатні комп'ютери, відкриваються нові перспективи для створення програмно-апаратних комплексів для обробки відеоданих.

Однією з ключових задач у цьому контексті є детектування рухів та розпізнавання обличчя, що відкриває широкі можливості в сферах безпеки, відеоспостереження, робототехніки та багатьох інших. Однак ці завдання вимагають ефективної обробки великих обсягів даних, що ставить виклик перед інженерами та дослідниками.

Ця дипломна робота присвячена розробці та вивченню програмно-апаратного комплексу для детектування рухів та розпізнавання обличчя на базі Raspberry Pi. Мета роботи – розробити ефективне та ресурсозберігаюче рішення для обробки відеоданих, використовуючи потужності одноплатної обчислювальної машини.

Об'єкт дипломної роботи становить собою програмно-апаратний комплекс, спрямований на вирішення завдань детектування рухів та розпізнавання обличчя. Даний комплекс призначений для інтеграції та ефективного функціонування на базі одноплатного комп'ютера. Об'єкт включає в себе не лише апаратну частину, таку як сам одноплатний комп'ютер та його складові, але й програмне забезпечення, що включає в себе розроблені алгоритми, моделі та інтерфейси для взаємодії з користувачем.

Предмет роботи полягає у глибокому вивченні, розробці та оптимізації алгоритмів, які забезпечують високу точність детектування рухів та розпізнавання обличчя, враховуючи при цьому обмежені ресурси одноплатного комп'ютера Raspberry Pi. Важливою складовою предмету дослідження є розробка оптимальних та ефективних методів обробки відеоданих, взаємодії з сенсорами та апаратною

частиною пристрою, а також створення інтерфейсу для зручного управління та взаємодії з системою.

Завдяки швидкому розвитку технологій та їх впровадженню в різноманітні галузі, виникає потреба в поєднанні високоефективного програмного із доступним апаратним забезпеченням.

Використання одноплатних комп'ютерів, таких як Raspberry Pi, стає все більш поширеним у розробці вбудованих систем, забезпечуючи ефективне використання ресурсів та доступ до різноманітних можливостей для інженерів та розробників. Розробка програмно-апаратного комплексу на базі Raspberry Pi стає актуальною задачею, оскільки це відкриває перспективи для створення доступних та ефективних рішень у вищезазначених областях. Для досягнення цієї мети передбачені такі завдання:

а) аналіз та вибір методів детектування рухів та розпізнавання обличчя. Розгляд різних підходів та методів для вибору оптимального способу обробки відеоданих;

б) розробка програмного забезпечення для взаємодії з камерою та обробки відеоданих. Створення ефективного алгоритму для розпізнавання рухів та облич;

в) інтеграція програмного забезпечення з апаратним забезпеченням Raspberry Pi. Забезпечення оптимальної взаємодії між програмним та апаратним середовищем;

г) експериментальне дослідження та апробація розробленого комплексу. Перевірка ефективності та точності розробленого рішення на реальних відеоданих.

Це дослідження має велике значення для розвитку вбудованих систем, що використовують відеоаналітику, і може знайти практичне застосування в різних сферах людської діяльності.

Дана робота проходила апробацію 02.02.2024 на конференції кафедри комп’ютерної інженерії Чорноморського національного університету імені Петра Могили.

1 АНАЛІТИЧНА ЧАСТИНА

Сучасні технології обробки відеоданих та розпізнавання обличчя відкривають широкі можливості для розробки програмно-апаратних комплексів з високою ефективністю та точністю. Зростання інтересу до цієї області обумовлене не лише швидким технологічним прогресом, але і розширеним застосуванням в різних сферах життя, включаючи безпеку, медицину, робототехніку та різноманітні сфери віртуальної та доповненої реальності. Одним із ключових аспектів розвитку цієї галузі є аналіз поточного стану предметної області. Нові методи та алгоритми обробки відеоданих надають змогу значно підвищити швидкість та точність розпізнавання обличчя. Використання нейронних мереж та глибокого навчання відкриває можливості для автоматизованого виявлення складних шаблонів та контекстів, що робить системи розпізнавання більш адаптивними до різних умов та сценаріїв.

Огляд існуючих рішень в цій області включає в себе як аналіз апаратних рішень, так і програмних підходів. Різноманітні сенсори, камери та системи введення дозволяють покращити якість збору відеоданих, що в свою чергу впливає на якість розпізнавання. Одночасно, розробники активно вдосконалюють алгоритми обробки, розширюючи їхню здатність працювати та забезпечуючи надійність в умовах різноманітних обставин. Формулювання завдань для розробників включає в себе вдосконалення алгоритмів розпізнавання, оптимізацію роботи на апаратному рівні та забезпечення високої захищеності та конфіденційності даних. Забезпечення взаємодії з іншими системами та інтеграція з різними платформами також є важливими аспектами розвитку цієї галузі.

У цілому, сучасні технології обробки відеоданих та розпізнавання обличчя відкривають перспективи для створення ефективних та універсальних систем, що знаходять застосування у різних сферах сучасного життя.

1.1 Розкриття об’єкту і предмету дослідження

Дипломна робота присвячена розробці програмно-апаратного комплексу для детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютера Raspberry Pi.

Об'єктом дослідження є системи комп'ютерного зору та їхнє використання у завданнях детектування рухів та розпізнавання обличчя. Системи комп'ютерного зору є комплексними технічними рішеннями, які використовують апаратне та програмне забезпечення для аналізу та обробки зображень. Ці системи здатні виявляти об'єкти, рухи та особливості зображення, роблячи їх доступними для подальшого використання в різних областях, включаючи безпеку, медицину та розваги.

Предметом дослідження є програмно-апаратний комплекс для детектування рухів та розпізнавання обличчя, розроблений на базі Raspberry Pi. Raspberry Pi є популярним пристроєм у сфері вбудованих систем та має невеликі розміри та вартість, що робить його привабливим для використання у великому спектрі застосувань, включаючи системи комп'ютерного зору. Цей програмно-апаратний комплекс буде призначений для виявлення та аналізу рухів в зображеннях, а також для ідентифікації обличчя на зображеннях.

Одноплатний комп'ютер Raspberry Pi обраний як основа для програмно-апаратного комплексу з численних причин. Важливі характеристики цього пристрою включають низьку вартість, компактність та відмінну спільноту розробників, що робить його привабливим для широкого кола завдань. Розгляд апаратних можливостей Raspberry Pi, таких як процесор, об'єм оперативної пам'яті, наявність відео- та фотокамер, дасть можливість максимально використовувати його потенціал у вирішенні завдань детектування рухів та розпізнавання обличчя.

Програмно-апаратний комплекс спрямований на вирішення завдань комп'ютерного зору, зокрема детектування рухів та розпізнавання обличчя. Він може бути застосований в різноманітних галузях, таких як системи безпеки,

відеоспостереження, автоматизація та інші області, де важливо відслідковувати рухи та ідентифікувати особи.

Розглядаючи апаратні та програмні особливості системи, акцент робиться на можливостях вбудованих модулів, таких як камери для захоплення зображень, графічний процесор для обробки графіки та потужний процесор для роботи з алгоритмами розпізнавання. Детальний аналіз апаратно-програмної архітектури дозволяє максимально використовувати можливості Raspberry Pi для ефективного вирішення поставлених завдань.

1.2 Опис і аналіз структурних і функціональних особливостей об’єкта дослідження

Структурні та функціональні особливості об'єкта дослідження будуть розглянуті з точки зору апаратної та програмної реалізації. Основна увага приділяється технічним характеристикам одноплатного комп'ютера Raspberry Pi, його можливостям у реалізації завдань з детектування рухів та розпізнавання обличчя.

Важливим аспектом є аналіз структурних елементів об'єкта дослідження – одноплатного комп'ютера Raspberry Pi. Розглядаються ключові компоненти, такі як процесор, оперативна пам'ять, порти введення-виведення, модуль бездротового зв'язку тощо. Особлива увага приділяється здатності пристрою взаємодіяти з камерами для захоплення відео та фото, що є ключовим елементом для дослідження.

Для початку розглянемо що таке системи комп’ютерного зору взагалі. Системи комп'ютерного зору представляють собою складні технічні рішення, які використовують апаратне та програмне забезпечення для аналізу та обробки зображень. Їхні структурні особливості можна розглядати в контексті їхніх апаратних та програмних компонентів, а також методів обробки та аналізу зображень. Нижче подано детальний огляд структурних особливостей систем комп'ютерного зору.

Системи комп’ютерного зору складаються з таких компонентів:

а) камери та сенсори: системи комп'ютерного зору використовують різні типи камер та сенсорів для отримання зображень. Вони можуть включати в себе відеокамери, теплові камери, 3D-сенсори та інші;

б) апаратні модулі для обробки зображень: в системах комп'ютерного зору графічні процесори використовуються для швидкого виконання операцій над зображеннями, що дозволяє прискорити обробку та аналіз;

в) оперативна пам'ять та зберігання: для тимчасового зберігання та обробки даних, отриманих від камер та сенсорів використовується RAM – random access memory (оперативна пам’ять). А для довгострокового зберігання можуть включати в себе SSD-диски – solid-state drive (твердотілий накопичувач) або інші носії для зберігання великої кількості зображень та інших вхідних даних;

г) процесор та алгоритми обробки: Центральний процесор - відповідає за загальне управління та координацію роботи системи, а детектори та класифікатори використовуються для виявлення об'єктів та класифікації їхніх ознак на зображеннях;

д) подача та виведення даних: монітори чи інші вихідні пристрої виводять результати аналізу та обробки для відображення користувачеві або іншим системам;

е) системи штучного інтелекту та навчання з учителем: нейронні мережі використовуються для складних завдань виявлення та розпізнавання об'єктів, обличчя, жестів тощо. Алгоритми машинного навчання використовують історичні дані для навчання системи на основі певних завдань;

ж) інтерфейси та зовнішні пристрої: порти введення-виведення дозволяють підключати зовнішні пристрої, такі як монітори, клавіатури, миші, або інші сенсори;

з) мережеве підключення: Ethernet/Wi-Fi/Bluetooth дозволяють обмін даними між системами комп'ютерного зору та іншими пристроями через мережу.

Ці структурні особливості дозволяють системам комп'ютерного зору взаємодіяти з навколишнім середовищем, оброблювати та аналізувати зображення для різноманітних застосувань, включаючи розпізнавання облич, виявлення рухів, системи безпеки, медицину, та багато іншого.

1.3 Огляд і аналіз сучасного стану інформаційних технологій у предметній області

Сучасний стан інформаційних технологій в області систем комп'ютерного зору є вражаючим та динамічно розвивається. За останні роки відбулися значні зміни та інновації, які впливають на способи застосування цих технологій в різних галузях.

Однією з ключових тенденцій є значний розвиток апаратних засобів для систем комп'ютерного зору. Процесори стали більш потужними, операційна пам'ять – більш об'ємною, а графічні процесори стали більш ефективними в обробці графіки та виконанні складних алгоритмів. Моделі одноплатних комп'ютерів, таких як Raspberry Pi, надають дослідникам та розробникам доступ до потужних інструментів за доступною ціною.

Крім того, зростання важливості штучного інтелекту і глибокого навчання призвело до розширення можливостей систем комп'ютерного зору. Використання нейронних мереж, зокрема глибоких нейронних мереж, сприяє покращенню точності розпізнавання облич, об'єктів та інших образів. Архітектури, такі як конволюційні нейронні мережі та рекурентні нейронні мережі, виявляються особливо ефективними у завданнях обробки відеоданих.

Застосування систем комп'ютерного зору і стає все більш актуальним, особливо в галузі безпеки та моніторингу. Розвиток алгоритмів виявлення та відстеження об'єктів робить можливим використання цих технологій для швидкої реакції на події та автоматизації процесів.

1.3.1 Використання штучного інтелекту та нейронних мереж

Штучний інтелект та нейронні мережі визначають новий рівень розвитку сучасних систем комп'ютерного зору. Машинне навчання, що є ключовою складовою цих технологій, дозволяє системам автоматично навчатися та здатністю покращувати свою продуктивність з часом. Використання глибокого навчання та нейронних мереж відкриває широкі можливості для ефективного вирішення завдань розпізнавання облич, об'єктів та виявлення рухів.

Глибоке навчання, засноване на нейронних мережах, дозволяє системам визначати та аналізувати складні патерни в великих обсягах даних. Це забезпечує високу точність розпізнавання облич та об'єктів навіть у складних умовах, роблячи системи комп'ютерного зору більш адаптивними до різноманітних сценаріїв використання.

Машинне навчання в контексті систем комп'ютерного зору дозволяє автоматизовано адаптувати моделі до змін у вхідних даних, що робить їхню продуктивність більш гнучкою та ефективною. Це особливо важливо в задачах розпізнавання облич та об'єктів, де можуть виникати зміни в освітленні, позахисні та інші фактори.

Отже, використання штучного інтелекту, машинного навчання та глибокого навчання з нейронними мережами визначає сучасний прогрес у розвитку систем комп'ютерного зору, роблячи їхні можливості широкими та динамічними у різних областях застосування.

1.3.2 Розширення застосувань у різних галузях

Системи комп'ютерного зору на сьогоднішній день активно застосовуються в різних галузях, відзначаючись високою ефективністю та широким спектром застосувань. У сфері безпеки ці системи забезпечують відеоспостереження та розпізнавання облич, що підвищує рівень безпеки та допомагає в ідентифікації осіб або подій. В медицині системи комп'ютерного зору використовуються для діагностики та аналізу зображень, таких як рентгенівські знімки. Завдяки їм

можливе більш точне визначення патологій та швидше прийняття рішень у лікуванні пацієнтів. У виробництві системи комп'ютерного зору застосовуються для контролю якості виробів та автоматизації виробничих процесів. Вони допомагають виявляти дефекти та забезпечують високу точність при виконанні завдань у виробничому середовищі.

В сучасних системах комп'ютерного зору впроваджуються технології відстеження рухів, які знаходять застосування в ігровій індустрії, фітнесі та вирізанні відео. Це дозволяє створювати інтерактивні та адаптивні досвіди для користувачів у різних галузях. Розробка систем доповненої та віртуальної реальності також залежить від систем комп'ютерного зору, які відіграють ключову роль у створенні іммерсивного середовища. Ці технології відкривають нові можливості для взаємодії з довкіллям та реалістичного відтворення віртуальних об'єктів.

1.3.3 Виклики та проблеми

Незважаючи на успіхи, існують виклики та проблеми у сфері систем комп'ютерного зору. До них відносяться питання конфіденційності даних, етичні аспекти використання технологій в масовому спостереженні, а також потреба вдосконалення алгоритмів для роботи з різноманітними умовами освітлення та зображення.

Однією з основних проблем є зростаюча обуреність стосовно етичних аспектів використання систем комп'ютерного зору в суспільстві. Використання цих систем для масового спостереження може порушувати приватність та конфіденційність особистих даних. Наприклад, системи відеоспостереження у громадських місцях можуть стати предметом обговорення з точки зору етики та прав особистості.

Розпізнавання облич та об'єктів може бути важким завданням у складних умовах освітлення або на низькоякісних зображеннях. Наприклад, системи вночі чи в умовах слабкої освітленості можуть зазнавати викликів у роботі. Розробка

алгоритмів, які ефективно функціонують у різноманітних умовах освітлення, залишається актуальною задачею.

Незважаючи на значний прогрес у глибокому навчанні та нейронних мережах, викликами залишається точність розпізнавання та здатність систем адаптуватися до нових або неочікуваних ситуацій. Розробники систем комп'ютерного зору постійно стикаються з завданням створення алгоритмів, які будуть ефективними та стійкими до різноманітних викликів. За всім ростом технологічних можливостей систем комп'ютерного зору стоїть питання їхньої безпеки. Потенційні загрози включають можливість використання цих систем для створення фальшивих зображень, зловживанням збереженими даними або використання в атаках на приватні мережі.

Ще однією проблемою є вартість та доступність технологій для широкого кола користувачів. Новітні системи можуть бути високою за ціною, що обмежує їхнє поширення в певних галузях або серед конкретних користувачів. Зростання використання систем комп'ютерного зору також викликає потребу у сучасних правових та регуляторних рамках. Це включає в себе створення правил щодо використання таких систем у громадських місцях, визначення відповідальності за можливі помилки, а також встановлення стандартів конфіденційності та безпеки.

Забезпечення точності та ефективності систем комп'ютерного зору вимагає тривалого процесу навчання та підготовки моделей. Важливо враховувати широкий спектр сценаріїв та контекстів для забезпечення адекватності та гнучкості в різних умовах використання. Ці виклики та проблеми визначають напрями подальших досліджень та розробок у сфері систем комп'ютерного зору та вимагають комплексного підходу для їхнього вирішення.

1.4 Огляд і аналіз існуючих методів і засобів вирішення завдань

Методи та засоби, які використовуються в сучасних системах для детектування рухів та розпізнавання облич, розглядаються з точки зору їхньої ефективності та застосування у реальних умовах. Оцінюється придатність цих

методів для використання в обраному програмно-апаратному комплексі. Розділ спрямований на детальний огляд та аналіз наявних методів і засобів, які використовуються для вирішення завдань у галузі комп'ютерного зору та розпізнавання обличчя. Цей огляд є ключовим етапом у визначенні того, які підходи та технології будуть використані для реалізації програмно-апаратного комплексу на базі одноплатного комп'ютера Raspberry Pi.

1.4.1 Аналогічні методи вирішення завдання кваліфікаційної магістерської роботи

а) традиційні методи розпізнавання обличчя:
   1) метод головних компонент: застосовується для зменшення розмірності даних та ефективного виділення головних компонент зображення. Використовується для визначення головних характеристик обличчя та реалізації завдань розпізнавання;
   2) метод локальних бінарних шаблонів: орієнтований на аналіз текстури зображення за допомогою бінарних шаблонів. Ефективно визначає локальні особливості обличчя, допомагаючи в розпізнаванні;

б) методи глибокого навчання:
   1) згорткові нейронні мережі: застосовуються для розпізнавання облич та об'єктів, здатні автоматично визначати та вивчати корисні ознаки. Високий рівень точності розпізнавання, використовується у великій кількості завдань КМР;
   2) мережі довготривалої пам'яті: використовуються для аналізу послідовностей та роботи з даними в часі. Допомагають у відстеженні рухів та аналізі динамічних змін на зображеннях;

в) Засоби для відстеження рухів:
   1) оптичний потік: визначає швидкість та напрямок руху об'єктів на зображенні. Важливий для відстеження рухів, особливо у визначенні напрямку обличчя;

2) методи на основі фільтрів Калмана: використовують фільтри для прогнозування та корекції руху об'єктів. Забезпечують точність та стабільність у відстеженні рухів;

г) Сучасні підходи до розпізнавання обличчя:

1) використання 3D-моделей обличчя: використовують тривимірні моделі для поліпшення точності розпізнавання та адаптації до змін у положенні обличчя. Важливий для задач, де необхідно точне визначення форми та глибини обличчя;
2) OpenCV (Open Source Computer Vision Library): відкрита бібліотека, що містить реалізації багатьох алгоритмів комп'ютерного зору та обробки зображень. Використовується для широкого спектру завдань у комп'ютерному зорі;
3) Dlib: Засіб для розпізнавання обличчя та обробки зображень, що включає в себе ефективні інструменти для відстеження рухів. Забезпечує надійні засоби для виконання різноманітних завдань у галузі КМР.

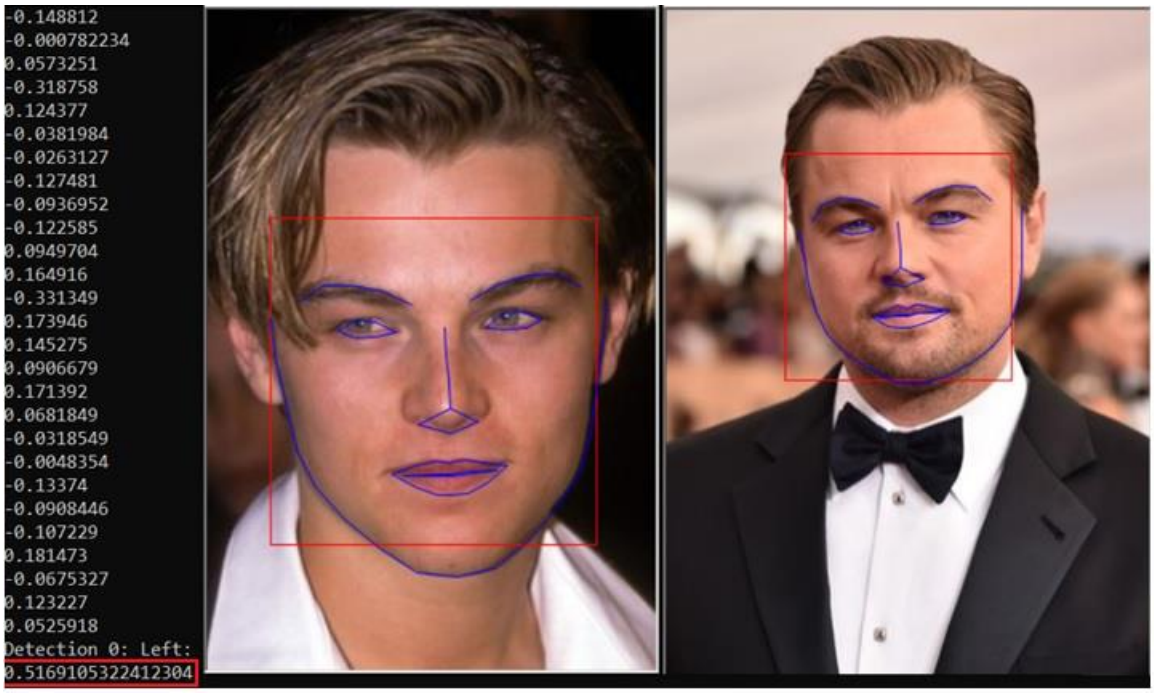


Рисунок 1.1 – Як dlib розпізнає обличчя

1.5 Обґрунтування та вибір підходів щодо виконання завдань

На підставі докладного аналізу різноманітних методів та існуючих технічних рішень виокремлюється та обґрунтовується стратегічний вибір підходів, які визначать розробку високоефективного програмно-апаратного комплексу. Основна акція робиться на технологічних рішеннях, спрямованих на детекцію рухів та розпізнавання обличчя, і спеціально адаптованих для впровадження на платформі Raspberry Pi.

Вибір конкретних підходів визначається їхньою придатністю до оптимального використання обмежених ресурсів Raspberry Pi, включаючи його обчислювальну потужність та обсяг оперативної пам'яті. Ретельний аналіз дозволяє відібрати технології, які оптимально впишуться у контекст обраної платформи, забезпечуючи високий рівень ефективності та точності в завданнях детекції рухів та розпізнавання обличчя.

Зазначений вибір підходів є стратегічно важливим у контексті реалізації програмно-апаратного комплексу, оскільки впливає на його загальну продуктивність та здатність адаптуватися до особливостей Raspberry Pi. Це також покликано забезпечити оптимальну сумісність з вимогами завдань детектування рухів та розпізнавання обличчя, що можуть варіюватися в залежності від конкретних сценаріїв та використання програмно-апаратного комплексу.

Висновки до розділу 1

Огляд сучасного стану інформаційних технологій у предметній області систем комп'ютерного зору свідчить про їхню велику потенційну користь та широкі можливості застосування в різних сферах життя. З розвитком апаратних засобів, алгоритмів машинного навчання та розширення застосувань в індустрії, системи комп'ютерного зору стають все більш необхідними і перспективними технологіями для майбутнього.

У розділі розглянуто широкий спектр існуючих методів та засобів, які можна використовувати для розробки програмно-апаратного комплексу на базі Raspberry Pi в галузі комп'ютерного зору та розпізнавання обличчя. Вибір конкретних підходів буде залежати від конкретних вимог, обмежень та завдань системи, а також від технічних характеристик самого пристрою.

Вибір підходів до виконання завдань кваліфікаційної магістерської роботи на базі Raspberry Pi вимагає уважного обґрунтування та аналізу. Прийняття рішень повинно бути підпорядковане специфікаціям системи, обмеженням платформи та вимогам до продуктивності, щоб забезпечити ефективну та стабільну роботу програмно-апаратного комплексу.

2 МАТЕМАТИЧНІ МЕТОДИ ТА МОДЕЛЮВАННЯ СИСТЕМИ

Вперше задача розпізнавання обличчя виникла в криміналістиці. Це призвело до розробки математичних методів та моделей як самого обличчя, так і процедур розпізнавання особи, що можна назвати завданням верифікації. Існує розмаїття факторів, таких як зміна освітлення, положення, міміка, які призводять до внутрішньокласової мінливості об'єкта розпізнавання, ускладнюючи роботу детектора обличчя. Незважаючи на значну кількість досліджень, які відбуваються у лабораторіях усього світу протягом декількох десятиліть, вимоги до реально працюючих систем комп'ютерного зору, здатних виявляти та розпізнавати людину в будь-яких умовах та позиціях, ще не були визначені. Сьогодні, потреба в розпізнаванні загострюється у п'яти напрямках: системи безпеки, банківська сфера, підтвердження особи, соціальні мережі та як особистий пароль[10].

При розпізнаванні особи можуть мати місце різні ситуації. Перша ситуація - це бажання об'єкта бути розпізнаним, наприклад, на терміналі прикордонної служби або при визначенні, чи має людина, яка хоче увійти в приміщення, дозвіл. Інша назва такої ситуації - система з кооперацією, коли людина сама допомагає при такій процедурі. Другий варіант ситуації - відсутність кооперації суб'єкта з системою візуального моніторингу, наприклад, вулична або прихована камера в банку. Цей варіант стикається з проблемою схожості людей один на одного. У найбільш складному випадку, коли система виявлення та ідентифікації людини використовується в умовах, коли швидко змінюється сцена подій або загальний фон, вимагається використання максимально доступної додаткової інформації для досягнення задовільних результатів роботи алгоритму.

Другим компонентом виступає можливість системи фіксувати рухомі об’єкти та реагувати на них. Основною ціллю виявлення руху є визначення зміни положення об'єкту між двома послідовними зображеннями. Крім того, виявлення руху об'єктів може сприяти їх розпізнаванню. Основна мета дослідження полягає в розпізнаванні пікселів, що належать одному об’єкту. Аналіз руху людського тіла

зацікавив дослідників через його різноманітність, яку впливають такі фактори, як фізична активність, медична діагностика, оцінка та віртуальна реальність. На сьогоднішній день для виявлення рухомих об'єктів використовуються в основному методи кадрового віднімання, віднімання фону та оптичного потоку.

2.1 Загальний підхід до моделювання системи розпізнавання обличчя

Загальний алгоритм для виявлення та ідентифікації людини за її обличчям складається з наступних етапів: спочатку виявляється присутність людини на зображенні, потім виокремлюється її фігура, виділяється голова, визначається кут спостереження голови (анфас або профіль), відбувається виділення особливостей обличчя та порівняння з еталонами для кінцевої ідентифікації. Деякі алгоритми дозволяють одразу перейти до визначення області, на якій розташоване обличчя людини. До таких алгоритмів відносяться методи, що базуються на часткових ознаках: аналіз контурів, класифікація за кольором шкіри, а також пошук/розпізнавання анатомічних ознак. Проте слід зауважити, що кращі результати можливі за допомогою методів, що використовують узагальнені ознаки та мета-алгоритми, такі як AdaBoost, RealBoost, та "слабкі" класифікатори для побудови "сильних" класифікаторів. До них належать такі методи, як HAAR-подібні ознаки, LBP, і поля Гауса[11].

Загальний порядок завдань при створенні системи розпізнавання представлений на рисунку 2.1. При наявності зображення або відеопотоку, розбитого на кадри, перше завдання - визначити місцезнаходження осіб на зображенні. Визначення ключових точок та нормалізація обличчя є двома ключовими пунктами, які мають найбільший вплив на якість та результати розпізнавання.

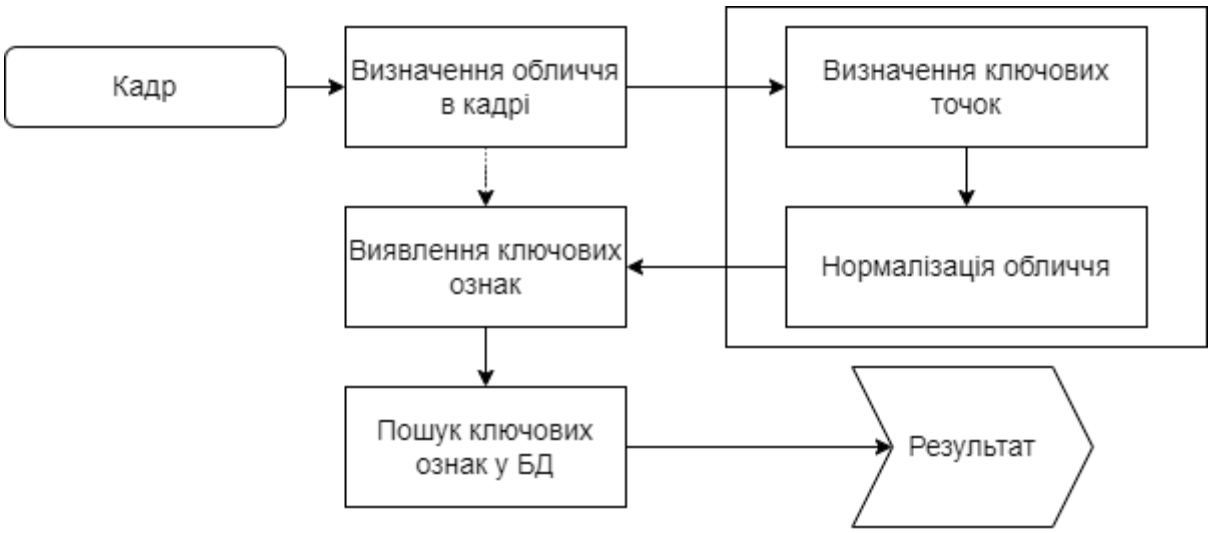


Рисунок 2.1 - Порядок завдань при розпізнаванні обличчя

2.1.1 Розгляд технології комп’ютерного зору

Комп'ютерний зір (Computer Vision, CV) - це галузь технологій, що спрямована на розробку комп'ютерних систем, які здатні виявляти, відстежувати та класифікувати об'єкти. Застосування технологій штучного інтелекту та машинного навчання перетворили аналіз, прогнозування та розпізнавання на новий рівень. Останнім часом комп'ютерний зір виявляється дуже перспективною галуззю. Однією з найважливіших проблем, яка активно досліджується в області комп'ютерного зору, є розпізнавання облич. Технологія Face ID , представлена на заході Apple Special Event у вересні 2017 року, стала чітким прикладом цього напрямку[12]. Компанія Facebook, що є лідером у цій сфері, розробила алгоритм, який демонструє ефективність розпізнавання на рівні 93%. Популярність Face ID призвела до створення багатьох бібліотек та інтерфейсів програмування додатків для розпізнавання облич. Ці рішення можуть бути спеціалізованими для конкретних галузей або універсальними, написаними на різних мовах програмування. Часто вибір серед цього розмаїття інструментів важко зрозуміти, який саме найкраще підходить для конкретної задачі.

2.1.2 Розгляд технології розпізнавання обличчя

Давайте розглянемо деякі відомі рішення для задачі розпізнавання облич на основі машинного навчання, такі як Facebook DeepFace, Apple Face ID, Microsoft Face API, Aware Nexa|Face та Samsung Face Recognition.

Facebook DeepFace є сервісом, спрямованим на полегшення користувачам соціальної мережі Facebook. Він використовує фотографії, які вже були позначені користувачами, шукає на них обличчя та призначає їм відповідні імена. Його ефективність розпізнавання становить 97,25%, що є вражаючим результатом.

Microsoft Face API є інструментом для створення систем розпізнавання облич. Він містить функції для пошуку, ідентифікації, групування та інші можливості, що дозволяють розробникам створювати власні додатки на основі цього API.

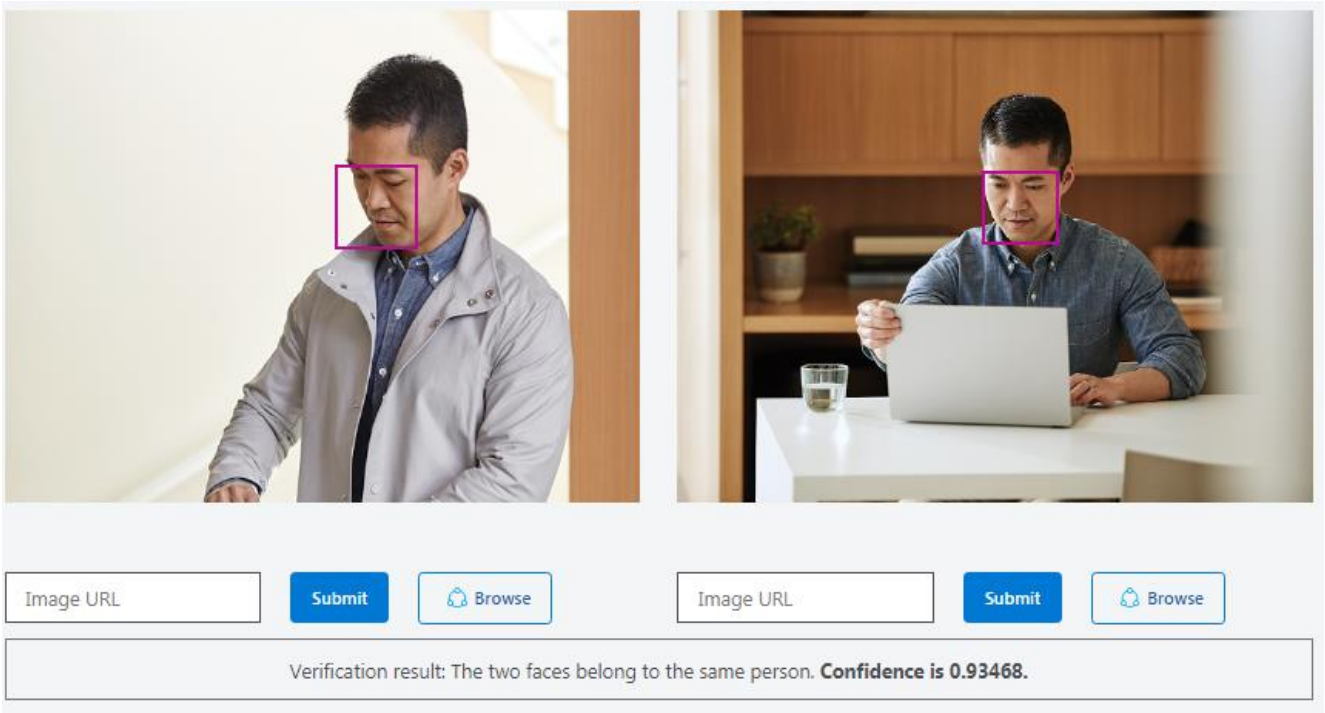


Рисунок 2.2 – Результат від Microsoft Face API

Система Apple Face ID призначена для авторизації власника телефону. Вона використовує спеціальну апаратну складову для розпізнавання облич у 2D та за поганих умов освітлення.

Система Aware Nexa|Face є складовою біометричної системи для ідентифікації особи, яка також включає в себе розпізнавання відбитків пальців, сітківки ока та інші. Ця система відрізняється високою ефективністю та захищеністю, але також має високу вартість.

Система Samsung Face Recognition розроблена для флагманських моделей телефонів Samsung і використовується для підтвердження особи. Хоча вона має високу ефективність, вона також має певні обмеження, такі як проблеми з розпізнаванням при поганому освітленні.

Рисунок 2.3 – Результат від Samsung Face Recognition

Кожна з цих систем має свої переваги та недоліки, і вибір між ними залежить від конкретних потреб та обмежень користувача.

2.2 Розробка системи розпізнавання обличчя

Процес розпізнавання облич [13] передбачає виконання ряду кроків, кожен з яких відіграє важливу роль у точності та ефективності системи.

а) пошук обличчя, що є необхідним навіть у випадку розпізнавання людей на фотографіях, відеоряді або в будь-якому іншому форматі;

б) позиціонування обличчя, оскільки більшість фотографій мають обличчя, які знаходяться під кутом до камери;

в) фіксування унікальних особливостей обличчя, що є ключовим кроком у процесі розпізнавання.

г) на останньому етапі [14] проводиться ідентифікація особи, де порівнюються отримані дані з наявними в базі. Якщо знаходяться відповідності, то виводиться ім'я особи, в іншому випадку розпізнається як невідома особа.

У цій роботі ми реалізуємо всі ці кроки побудови системи розпізнавання облич з використанням різних бібліотек. Ми порівнюємо результати розпізнавання, отримані з використанням цих бібліотек, та аналізуємо швидкість роботи кожного етапу у різних бібліотеках комп'ютерного зору.

Важливо відзначити, що результати досліджень можуть варіюватися залежно від конфігурації комп'ютера, на якому виконується розпізнавання. Враховуючи, що ця система буде функціонувати на одноплатному комп’ютері Raspberry Pi, усі використані бібліотеки налаштовані для роботи виключно на центральних процесорах, без використання графічних процесорів.

2.2.1 Вибір математичного методу

Розглянемо різноманітні методи, що використовуються для розпізнавання облич. Низка застосувань продемонструвала ефективність методів, що базуються на лінійній проекції, таких як аналіз основних компонент (PCA) [15] , незалежний аналіз компонент (ICA) [16], лінійний дискримінаційний аналіз (LDA) [17, 18], 2DPCA [19] та лінійний класифікатор регресії (LRC) [20]. Однак, у зв'язку з

різноманітністю умов освітлення, виразності обличчя та інших факторів, ці методи можуть недостатньо адекватно відображати обличчя. Це пояснюється тим, що лінії обличчя лежать на складному нелінійному та неопуклому колекторі у просторовому просторі.

Для вирішення таких випадків були розроблені нелінійні розширення, такі як ядерний аналіз основних компонент (KPCA) [21], ядерний лінійний дискримінаційний аналіз (KLDA) або локальне лінійне вбудовування (LLE) [22]. Найбільш нелінійні методи, що використовують методи ядра, спрямовані на відображення зображень облич у простір вищої розмірності, де колектор граней лінійний та спрощений. Таким чином, можна використовувати традиційні лінійні методи.

Хоча PCA, LDA та LRC вважаються алгоритмами лінійного підпростору, методи PCA та LDA зосереджені на глобальній структурі евклідового простору, тоді як LRC орієнтований на локальну структуру колектора. Ці методи спрямовані на лінійний підпростір, що охоплюється зображеннями власних поверхонь. Відстань від простору обличчя – це ортогональна відстань до площини [23], тоді як відстань у просторі – це відстань уздовж площини від середнього зображення. Ці обидві відстані можна перетворити на відстані Махаланобіса та дати ймовірнісні інтерпретації.

Крім того, були розроблені KPCA [24], ядерний CA [25] та узагальнений лінійний дискримінантний аналіз [26]. Незважаючи на сильну теоретичну основу методів на основі ядра, їх практичне застосування у проблемах розпізнавання не дало значного поліпшення порівняно з лінійними методами. Тому було розроблено ще одне сімейство нелінійних методів проекції, які успадкували простоту від лінійних методів та здатність обробляти складні дані з нелінійних. Серед них можна відзначити LLE [27] та збереження проекції місцевості (LPP) [28]. Вони розробляють схему прогнозування лише для навчальних даних, але їх спроможність проектувати нові елементи даних доволі низька.

У другій категорії методів знаходяться репрезентативні методи, які шукають особливості кожного зображення та мають переваги перед цілісними ознаками. Ці методи є стійкішими до локальних змін, таких як експресія, оклюзія та нерівність. Один з загальних репрезентативних методів – це локальні бінарні шаблони (LBP) [29,30], які описують суміжні зміни навколо центрального пікселя простим, але ефективним способом.

2.3 Математична модель розпізнавання обличчя

Один з відомих алгоритмів для виявлення облич на зображенні – це алгоритм HOG (від англ. - Histogram of Oriented Gradients, гістограми напрямлених градієнтів), розроблений у 2005 році Навітаном Далалом і Білом Трігсом. Це один з дескрипторів ознак, які використовуються в комп'ютерному зорі та обробці зображень для виявлення об'єктів. Ця методика вимірює напрямки градієнта в окремих областях зображення. Вона подібна до методу гістограм спрямування контурів, описувачів масштабоінваріантного перетворення ознак та значень форми, але відрізняється тим, що обчислення проводиться на щільній сітці рівномірно розташованих комірок, і вона використовує локальне нормування з перекриттям для підвищення точності. Робота цього алгоритму включає наступні кроки:

а) розрахунок значення градієнта: це часто виконується за допомогою одновимірних центрованих масок і фільтрування кольору або насиченості зображення за допомогою фільтруючих ядер, наприклад, [−1, 0, 1] та [−1, 0, 1];

б) побудова гістограмних комірок: комірки можуть мати прямокутну або круглу форму, і канали гістограми можуть розподілятися рівномірно від 0 до 180 або 0 до 360 градусів, залежно від того, чи використовується беззнаковий чи знаковий градієнт. Для розпізнавання облич найчастіше використовується беззнаковий градієнт, який комбінується з 9 каналами гістограми;

в) локальна нормалізація: для визначення змін освітлення та контрастності сила градієнта повинна бути локально нормалізованою. Це

досягається шляхом об'єднання комірок в більші блоки. Зазвичай ці блоки перекриваються, що означає, що кожна комірка враховується декілька разів у фінальному дескрипторі. Існують дві основні геометрії блоків: прямокутні (R-HOG) і круглі (C-HOG). Експериментально виявлено, що оптимальним для багатьох завдань є об'єднання чотирьох комірок по 8x8 пікселів у один блок розміром 16x16 пікселів.

г) після об'єднання блоків важливо нормалізувати їхні значення, щоб уникнути впливу неякісних зображень. Існують чотири методи нормалізації, із яких найефективнішим довів себе метод, згаданий в джерелі [30].

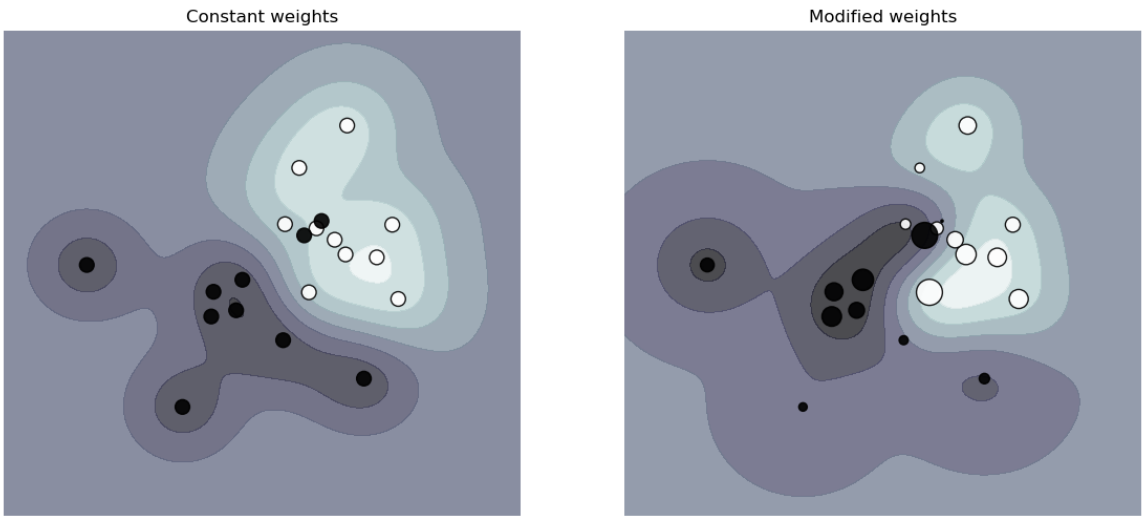


Рисунок 2.4 – нормалізація вагів

У цьому методі можна винести формулу:

$$\text{L2} - \text{norm: } f = \frac{\text{V}}{\|\text{V}\|_2^2 + e^2},$$

де:

а) V – не нормалізований вектор блоків;

б) || V ||k – k-норма вектора;

в) k – достатня константа.

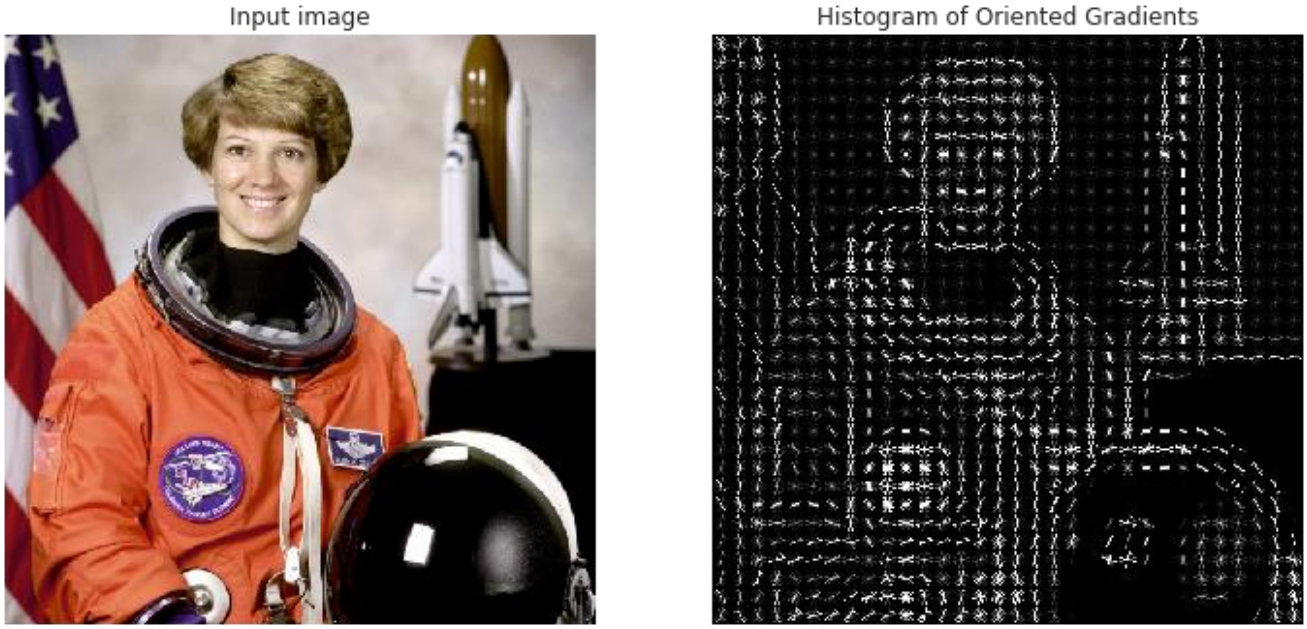


Рисунок 2.5 – спрощена візуалізація HOG методу

HOG дескриптори ефективно використовуються для виявлення та класифікації об'єктів у зображеннях, особливо коли мова йде про людські силуети. Наприклад, у комбінації з методом підтримуваних векторів для класифікації, HOG є міцним фундаментом для алгоритмів виявлення пішоходів. Однак метод HOG має певні обмеження. Він може бути чутливим до дуже сильних змін в позиції об'єкта, його розміру або орієнтації та до деформацій.

Після виявлення обличчя на зображенні виникає необхідність коректного його позиціонування, оскільки більшість облич не знаходяться у центрі або можуть бути повернуті під кутом, що може негативно вплинути на подальшу якість розпізнавання. Для вирішення цієї проблеми застосовується алгоритм оцінки орієнтирів обличчя (face landmark estimation). Основна концепція полягає в пошуку 68 орієнтирів (англ. landmarks), які характерні для кожного обличчя, такі як верхня частина підборіддя, внутрішній край брів, зовнішній край очей, нижня точка носа, верхня і нижня точки губ та інші. Після виявлення цих орієнтирів за допомогою простих маніпуляцій (повороти, збільшення або зменшення), обличчя центрується для подальшої обробки.

Рисунок 2.6 - 68 точок-орієнтирів, які розпізнаватиме метод

Для вирішення завдання позиціонування облич у застосунку ми використовуємо бібліотеку OpenFace. Переходимо до етапу, який непосредньо пов'язаний з розпізнаванням облич. Щоб ідентифікувати обличчя, потрібно виявити їхні особливості. Перше питання, яке виникає, це організація процесу розпізнавання. Найпростіший спосіб полягає у порівнянні невідомого обличчя, отриманого на попередньому кроці, з усіма записаними в базі даних обличчями. Однак при великих обсягах даних цей підхід призводить до затримок у часі вибірки та може вимагати розгляду альтернатив, таких як зберігання даних у хмарі. Для більшості проектів з розпізнаванням, як вказано в, швидкість розпізнавання є вирішальним чинником.

Через неефективність та високу обчислювальну складність попереднього підходу було запропоновано виділяти кілька основних ознак обличчя. Проте дослідження показали, що розмір, колір очей, довжина носа, величина губ та інші характеристики не мають значення для комп'ютерного розпізнавання обличчя, оскільки комп'ютер оцінює зображення піксель за пікселем, а не обличчя загалом.

Для вирішення цієї проблеми було запропоновано використовувати глибоку згорткову нейронну мережу (deep convolutional neural network, DCNN), яка навчається визначати 128 унікальних числових характеристик обличчя. Процес навчання такої нейронної мережі працює за таким принципом:

а) завантажити зображення обличчя відомої особи;

б) завантажити інше зображення обличчя цієї ж людини;

в) завантажити зображення обличчя іншої людини;

г) нейронна мережа виправляє отримані значення таким чином, щоб 128 характеристик зображень, які були завантажені на перших двох кроках, максимально наближалися одна до одної, тоді як зображення, що було завантажене на третьому кроці, максимально відрізнялося.

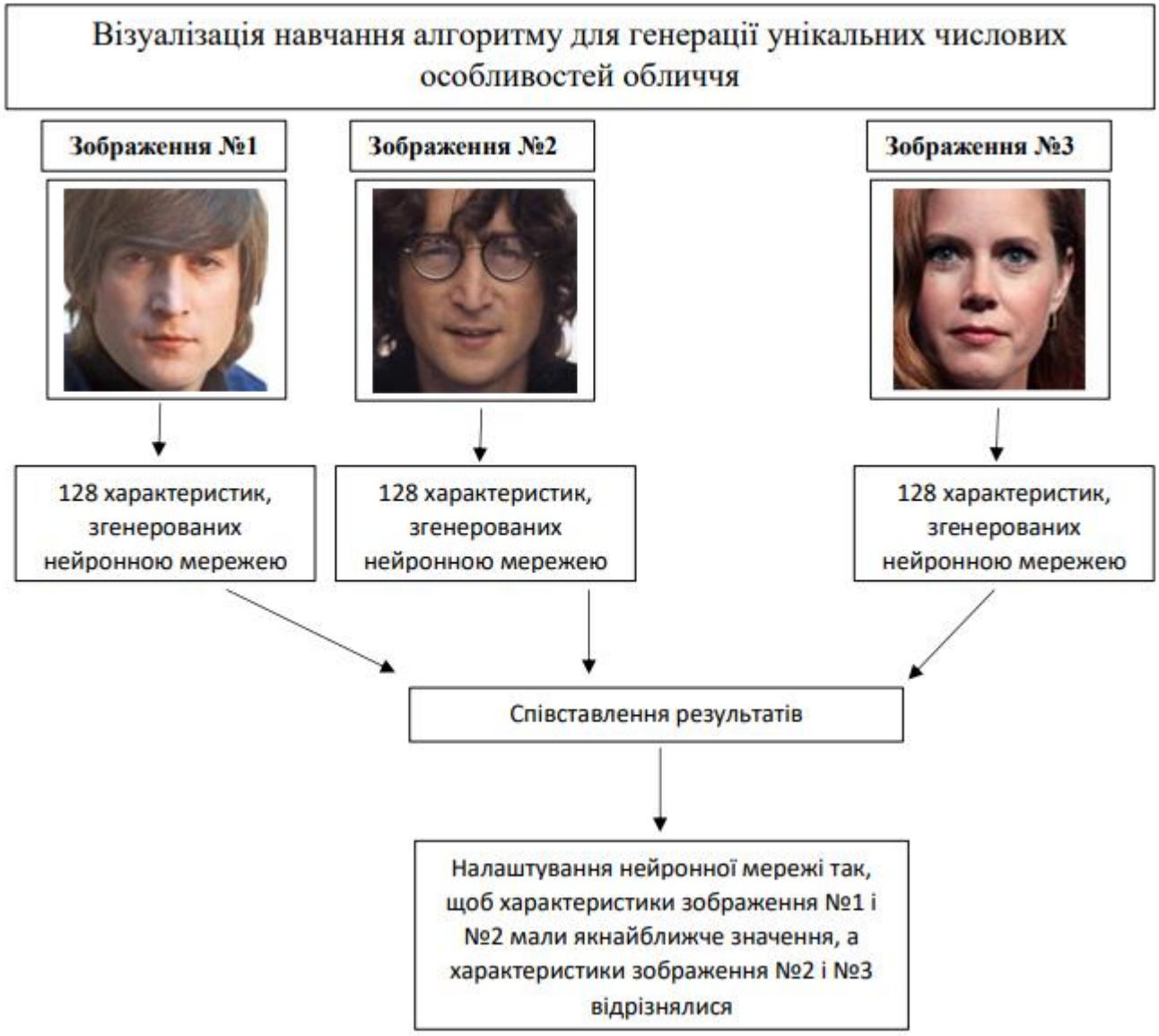


Рисунок 2.7 - Візуалізація процесу навчання алгоритму

Ідея зведення складних необроблених даних до списку комп'ютерно генерованих номерів отримала найбільший розвиток у сфері машинного навчання і спочатку була використана в галузі перекладу. Перший такий підхід до обробки обличь був представлений інженерами компанії Google у 2015 році. Процес навчання глибокої згорткової нейронної мережі для створення унікальних числових характеристик обличчя є складним, оскільки вимагає великої бази даних облич та значної обчислювальної потужності комп'ютера.

Навіть з використанням графічних карт NVIDIA Tesla, які підтримують CUDA, для прискорення моделі, процес навчання може займати до 24 годин. Однак, як тільки нейронна мережа натренується, вона зможе генерувати унікальні характеристики для будь-якого обличчя, навіть якщо модель раніше не бачила його. Цей етап надзвичайно важливий, оскільки нейронна мережа тренується лише один раз, і від неї залежить ефективність розпізнавання облич. У випадку, якщо немає можливості натренувати власну нейронну мережу для створення характеристик, можна використати вже навчену мережу.

Останній етап алгоритму полягає у порівнянні наявних даних, зокрема 128 характеристик певного обличчя, які були отримані на попередньому етапі, з наявними даними про людей. Цей етап може бути реалізований двома способами:

- база даних: Цей спосіб організації даних може бути використаний для невеликих об'ємів даних, коли необхідно ідентифікувати невелику групу людей та немає достатньої кількості фотографій для якісного навчання методу опорних векторів;
- метод опорних векторів (SVM): Це лінійний класифікатор, який використовується для навчання набору даних з n точок у виді:

$$(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)$$

де y є 1 або -1, і кожен з них вказує клас, до якого належить точка $\vec{x}i$. Кожний $\vec{x}i$ є p-вимірним дійсним вектором. Тоді потрібно знайти «максимально

розділову гіперплощину», яка відділяє групу точок $\vec{xi}$, для яких $yi = 1$, від групи точок, для яких $yi = -1$, і визначається так, що відстань між цією гіперплощиною та найближчою точкою $\vec{xi}$ з кожної з груп є максимальною. Даний графік зображений на рис. 2.8.

Будь-яку гіперплощину можна описати як набір точок $\vec{x}$, які відповідають наступному критерію:

$$\vec{w} * \vec{x} - b = 0$$

де $\vec{w}$ є вектором, що не обов'язково є нормалізованим, до цієї гіперплощини. Параметр $\frac{b}{\|\vec{w}\|}$ визначає зсув гіперплощини відносно початку координат вздовж вектора нормалі $\vec{w}$. Усе, що потрібно зробити, це навчити метод опорних векторів знаходити найвдаліший збіг оброблюваного зображення із зображеннями з бази даних відомих обличь. Час роботи цього класифікатора займає мілісекунди, що ідеально підходить для швидкої ідентифікації обличь.

Враховуючи всі переваги та недоліки, ми обираємо спосіб ідентифікації за допомогою методу опорних векторів. На цьому етапі немає сенсу порівнювати швидкість, оскільки набір даних для навчання однаковий, відповідно, класифікатор створюємо один для обох бібліотек, тому на цьому етапі швидкість роботи буде однаковою для обох бібліотек.

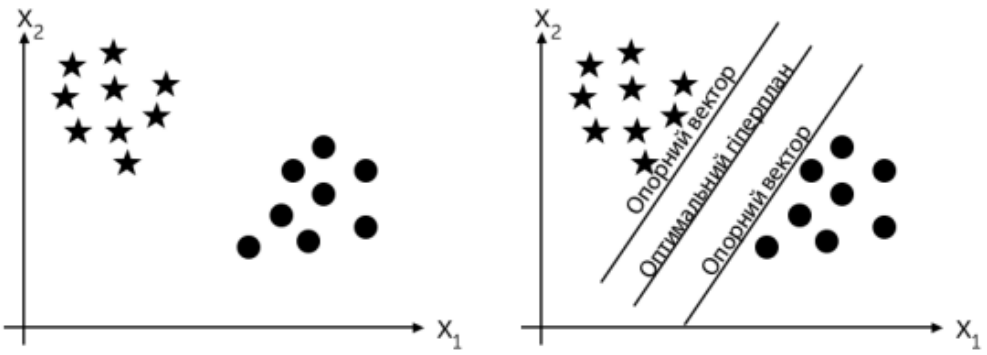


Рисунок 2.8 – Принцип дії SVM

Зокрема, існують специфічні аспекти, які можуть сприяти підвищенню ефективності процесу навчання:

– наявність чітких і різких меж між даними (у нашому випадку - 128 характеристик обличчя) є ключовою, оскільки це дозволить розбити їх на окремі групи. Ефективність цього процесу залежить від якості алгоритму визначення унікальних особливостей обличчя;

– збільшення обсягу навчальної вибірки не завжди призводить до очікуваного збільшення швидкодії. У багатьох випадках воно лише збільшує час навчання алгоритму і може призвести до перенавчання (overfitting). Для підвищення ефективності, важливо використовувати дані, очищені від шумів. Також важливо, щоб дані з одного класу не перетиналися з даними іншого класу.

Разом з методом опорних векторів, пропонується використовувати Local Binary Pattern (LBP). Центрально-симетрична модифікація цього алгоритму дозволяє ефективно зменшити обсяг пам'яті, а також складність обчислень під час класифікації. Замість використання значень яскравості центрального пікселя околиці, як граничного значення для кожного пікселя околиці, береться враховане значення яскравості протилежного пікселя відносно центра околиці. Це дозволяє скоротити кількість розрядів значень перетворень пікселів до чотирьох. В результаті розмірність гістограми ознак зменшується до $2^4 = 16$. Така модифікація алгоритму дозволяє застосовувати його для розпізнавання зображень у режимі реального часу.

2.4 Загальний підхід до моделювання детекції рухів

Методи виділення об'єктів і класифікація зображень є критично важливими в галузі комп'ютерного зору. Ці методи дозволяють комп'ютерам розуміти та ідентифікувати об'єкти та за допомогою цифрових зображень як вхідних даних. Протягом багатьох років методи комп’ютерного зору використовувалися в кількох сферах, включаючи охорону здоров’я, виробництво, роздрібну торгівлю, та багато

інших. Однак існує плутанина між класифікацією зображень та виявленням об'єктів, і їх різницю необхідно чітко зрозуміти.

Класифікація зображень - це техніка, що використовується для класифікації або прогнозування класу конкретного об'єкта на зображенні. Основна мета цієї техніки - точно визначити особливості зображення. Зазвичай методи класифікації зображень можна розділити на параметричні та непараметричні, контрольовані та неконтрольовані, а також жорсткі та м'які класифікатори. У випадку контрольованої класифікації результати базуються на створеній межі рішення, яка ґрунтується на вхідних та вихідних даних, наданих під час навчання моделі. У випадку неконтрольованої класифікації результати базуються на аналізі вхідного набору даних, і функції не надаються безпосередньо в моделі. Основні кроки, пов'язані з методами класифікації зображень, включають визначення системи класифікації, виділення ознак, вибір навчальних зразків, попередню обробку зображення, вибір методу класифікації, обробку після класифікації та оцінку загальної точності. Вхідні дані зазвичай є зображенням певного об'єкта, а вихідні дані - передбаченими класами, які визначають і відповідають вхідним об'єктам.

Виявлення об'єктів, з іншого боку, визначає розташування об'єктів на зображенні та їх класифікацію. Ця методика дозволяє шукати певні класи об'єктів, такі як автомобілі, люди, тварини тощо, і використовується в різних проектах, включаючи виявлення облич, пішоходів, транспортних засобів, дорожніх знаків та відеоспостереження. Конвеєр традиційних моделей виявлення об'єктів складається з трьох етапів: вибір інформативної області, виділення ознак і класифікація. Однак з появою глибокого навчання стали доступні більш ефективні моделі, що забезпечують високу точність у виявленні об'єктів.

Метод R-CNN (Region-based Convolutional Neural Network) представляє собою поєднання пропозицій регіонів зі згортковими нейронними мережами. Він дозволяє локалізувати об'єкти за допомогою глибокої мережі та навчати моделі з високою точністю, навіть при невеликій кількості анотованих даних. R-CNN

забезпечує високу точність виявлення об'єктів, використовуючи ConvNet для класифікації об'єктів. Цей метод може масштабуватися до тисяч класів об'єктів і уникати приблизних методів, таких як хешування.

Повністю згорткові мережі на основі регіонів або R-FCN (Region-based Fully Convolutional Network) є детектором об'єктів, що використовує повністю згорткову архітектуру. На відміну від інших регіональних детекторів, таких як Fast R-CNN або Faster R-CNN, R-FCN є повністю згортковим, що дозволяє використовувати майже всі обчислення для всього зображення. R-FCN використовує спільні, повністю згорткові архітектури, такі як у FCN, для класифікації ROI (Region of Interest) за категоріями об'єктів і фонами. Single Shot Detector (SSD) є методом виявлення об'єктів на зображеннях за допомогою однієї глибокої нейронної мережі. SSD дискретизує простір обмежувальних рамок і масштабує його відповідно до розташування об'єктів. Мережа поєднує прогнози з кількох карт об'єктів з різною роздільною здатністю, щоб ефективно обробляти об'єкти різного розміру.

Об'єднання просторових пірамід (SPP-net) - це мережева структура, яка генерує представлення фіксованої довжини незалежно від розміру зображення. SPP-net покращує всі методи класифікації зображень на основі CNN і дозволяє уникнути багаторазового обчислення згорткових ознак. You Only Look Once (YOLO) є одним з популярних алгоритмів виявлення об'єктів, який пропонує швидку обробку зображень з високою точністю. YOLO використовується дослідниками по всьому світу і надзвичайно швидкий у порівнянні з іншими методами виявлення об'єктів. Незважаючи на досягнення в контрольованому середовищі, проблема виявлення об'єктів залишається невирішеною в неконтрольованих місцях. Однак технології, що базуються на нейронних мережах, широко використовуються в різних галузях, завдяки швидкій автоматизації процесів та здешевленню апаратних засобів для реалізації програмно-апаратних комплексів. Продовжуються активні розробки в цьому напрямку.

2.5 Обґрунтування та вибір програмного забезпечення

Вибір програмного забезпечення для програмно-апаратного комплексу з розпізнавання облич та детекції рухів є ключовим етапом, оскільки від цього залежить ефективність та надійність системи. Програмне забезпечення повинне мати високу точність в розпізнаванні облич та детекції рухів, а також здатність працювати з великим обсягом даних. Важливо провести тестування програмного забезпечення на різних умовах, щоб переконатися в його ефективності.

2.5.1 Використання Python

Перш за все, для виконання завдання магістерської роботи, треба роздивитись таку мову програмування як Python. Python - це потужна мова програмування, яка знаходить застосування у різних галузях. Однією з ключових особливостей програмування на Python є можливість інтерпретації коду, що означає, що оператори мови транслюються та виконуються динамічно під час виконання. Інтерпретатор та стандартна бібліотека Python доступні для вільного використання на веб-сайті Python і працюють на всіх сучасних операційних системах. Мова програмування Python також доступна для будь-якого бажаючого на тому ж веб-сайті.

Цікавою особливістю є можливість легко розширити інтерпретатор Python за допомогою додавання нових типів даних або функцій, навіть на інших сучасних мовах програмування. У Python для написання коду використовуються англійські ключові слова, а не розділові знаки, що призводить до меншої кількості конструкцій синтаксису порівняно з іншими мовами програмування.

Серед переваг Python можна відзначити:

а) легкість освоєння мови, простота написання та читання коду, що робить її ідеальним вибором для початківців. часто для виконання завдань потрібно менше рядків коду, ніж на багатьох інших мовах програмування;

б) інтерпретованість мови, що означає, що програміст не повинен компілювати програму перед її виконанням;

в) можливість інтерактивної взаємодії з інтерпретатором python для написання коду;

г) відкритий вихідний код;

д) велика стандартна бібліотека та багато існуючих бібліотек, які можна імпортувати за допомогою спеціального менеджера pip.

2.5.2 Використання бібліотеки OpenCV

Далі нам знадобиться бібліотека, яка вже включає у себе певнй код для досягання нашої мети. OpenCV є бібліотекою, яка писалася мовою програмування C++ і дозволяє створювати програми з використанням комп'ютерного зору в реальному часі. Вона виконує обробку зображень та аналіз відео, зокрема виявлення об'єктів, таких як обличчя людини.

Термін "комп'ютерний зір" описує дисципліну, яка допомагає комп'ютерам розуміти тривимірну сцену на основі її 2D зображень. Метою комп'ютерного зору є створення моделей, які наближені до людського зору, з використанням програмного та апаратного забезпечення. За допомогою OpenCV можна:

а) зчитувати та записувати зображення;

б) захоплювати та зберігати відео;

в) обробляти зображення, такі як фільтрування чи перетворення;

г) виявляти конкретні об'єкти, такі як обличчя, люди, очі, автомобілі на відео чи зображеннях за допомогою шаблонів;

д) аналізувати відеопотік.

Незважаючи на те, що OpenCV розроблено на C++, на сьогодні існує безліч прив'язок до різних сучасних мов програмування, таких як Python і Java. OpenCV може працювати на різних операційних системах, включаючи Windows, Linux і інші.

2.5.3 Використання Face Recognition Library

Ще одна відкрита бібліотека, що базується на C++, - це Face Recognition Library, яка відрізняється високою точністю та зручністю у використанні. Вона легко розгортається на сервері, навіть без потреби у дорогому GPU, і має досить прості вимоги. Що може зробити ця бібліотека?

Вона може здійснювати пошук облич на фотографіях. Подане зображення аналізується, і програма визначає обличчя, причому, на відміну від OpenCV, вона допускає поворот голови більше, ніж на 30 градусів. Також вона може розпізнавати риси обличчя на фотографіях, виділяючи контури і важливі точки для ідентифікації, такі як очі, ніс, рот та підборіддя. Бібліотека здатна будувати карти координат специфічних точок обличчя, таких як контур ока, який визначається за допомогою 6 точок, та загальна карта, що містить 68 точок, що дозволяє проводити точні порівняння та ідентифікацію облич. Вона може ідентифікувати осіб на фотографіях, визначаючи, хто зображений на кожному зображенні. Можна створювати базу "своїх" людей та визначати їх, а також "чужих", за допомогою координат точок облич.

Ця бібліотека може застосовуватися в реальних часових рішеннях, наприклад, в роботі з потоковим відео, і використовувати з іншими бібліотеками Python.

2.5.4 Використання MySQL

MySQL представляє собою систему керування реляційними базами даних, які можна уявити як структуровану колекцію інформації. Ця інформація може включати в себе як невеликі списки товарів або переліки книг в бібліотеці, так і великі обсяги даних щодо рахунків користувачів у банківській сфері. Сучасні комп'ютери здатні ефективно опрацьовувати великі обсяги інформації, тому система управління базами даних стала ключовою складовою у виконанні програм. Її можна впроваджувати за допомогою спеціальних утиліт, фреймворків або вбудованого коду у вже існуючі додатки.

Реляційна база даних - це набір інформації, який не зберігається в одному списку, а розділяється на таблиці для швидкого та гнучкого управління великими обсягами даних. Таблиці пов'язуються між собою за допомогою спеціальних зв'язків, що дозволяє об'єднувати дані з різних таблиць під час виконання запитів. SQL (Structured Query Language) - це мова запитів, яка є стандартом для доступу до баз даних.

Основними перевагами MySQL є швидкість, легкість використання та надійність. Вона була розроблена для управління великими обсягами даних з метою забезпечення високої швидкості обробки. MySQL має широкий функціонал для користувача і підходить для доступу до баз даних через Інтернет завдяки своїй доступності, високому рівню безпеки та швидкості. Програмне забезпечення MySQL поширюється з відкритим вихідним кодом, що дозволяє його використання та модифікацію будь-кому.

2.5.5 Використання Flask

Flask - це засіб для розробки веб-сайтів на мові Python, що представляє собою невеликий фреймворк із вбудованим веб-сервером. Під розробкою веб-додатку ми розуміємо створення спеціальних контролерів для обробки HTTP запитів на власному сервері з метою динамічного отримання даних для всього сервісу. Тому для роботи було вибрано Python 3 та Flask для створення запитів REST API. Завданням REST API є обслуговування простої структури даних, де будь-яке оновлення маркується новим часовим позначенням та відображається в консолі. Ці дані можуть бути збережені в базі даних, локальному файлі або доступні через мережевий протокол.

Однією з метою API є відокремлення даних від програми, яка їх використовує, приховуючи деталі реалізації. Використання REST API передбачає налаштування можливостей клієнта для відправлення запитів згідно з вказаними методами на кінцеві точки серверного додатку. Це дозволяє реалізувати

спілкування клієнта з сервером в додатку, спрямовуючи увагу на розробку конкретних частин додатку.

Початково створювався як проста оболонка для Werkzeug і Jinja, але став одним з найпопулярніших фреймворків для веб-додатків Python. Він дає пропозиції реалізації, але не вимагає попередніх залежностей або конкретного макету проекту. Розробник просто обирає інструменти та бібліотеки, які хоче використовувати.

Розроблений для швидкого і легкого початку розробки з можливістю розширення до складних додатків у майбутньому.

2.5.6 Використання Microsoft Visual Studio Code

Microsoft Visual Studio Code - це безкоштовний редактор, який призначений для розробників і дозволяє писати код на будь-якій мові програмування, уникнувши постійного переключення між редакторами. Однією з головних особливостей VS Code є його можливість кастомізації за допомогою різних розширень: ви можете змінювати теми, додавати нові мови програмування та інструменти для дебагу, а також підключатися до різних сервісів. Крім того, окрім підсвічування синтаксису, VS Code підтримує функцію IntelliSense, яка надає підказки для ключових слів мови програмування та інформацію про їх параметри. Це означає, що під час написання програми ви отримуєте спеціальне контекстне меню, яке допомагає економити час, потрібний для введення тексту. Узагальнюючи, цей редактор є дуже простим і зручним для роботи з кількома різними мовами програмування.

Висновки до розділу 2

Комп'ютерний зір, який представляє галузь технологій, спрямованих на розробку систем виявлення, відстеження та класифікації об'єктів, став необхідною складовою в сучасному світі. Використання штучного інтелекту та машинного навчання перетворило аналіз та розпізнавання на новий рівень ефективності. Особливо важливим напрямком є розпізнавання облич, що знайшло широке

застосування у таких системах, як Face ID від Apple та алгоритми від Facebook, що показують високу ефективність.

Сучасні технології розпізнавання облич засновані на машинному навчанні та використовуються у системах Facebook DeepFace та Microsoft Face API. Ці рішення відображають високу точність і дозволяють створювати ефективні системи розпізнавання для різних галузей. Хоча вибір серед інструментів для розпізнавання облич може бути складним, зважаючи на різноманіття доступних рішень, ключовою є здатність вибрати інструмент, який найкраще відповідає конкретним вимогам та завданням. Тому для досягання мети цієї кваліфікаційнї магістерської роботи було обрано апаратне забезпечення: одноплатний мікрокомп’ютер Raspberry Pi та модуль ширококутної камери 12МП IMX708. Серед програмного забезеченнння: Python, OpenCV, MySQL та середа розробки Microsoft Visual Studio Code.

3 АПАРАТНО-ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

При проектуванні апаратно-програмного забезпечення важливо враховувати, що одноплатний комп'ютер Raspberry Pi має свої унікальні особливості як платформа. Цей пристрій відомий своїми обмеженими ресурсами, такими як обчислювальна потужність, обсяг оперативної пам'яті та можливості графічного прискорення. Отже, вибір методів та алгоритмів повинен бути ретельно продуманий та оптимізований для ефективної роботи на даному пристрої. Розробка комплексу на Raspberry Pi вимагає особливої уваги до ефективного використання обмежених ресурсів. Зокрема, вибрані методи мають бути адаптовані до конкретних характеристик пристрою, таких як його обмежена обчислювальна мережа та обмежена оперативна пам'ять.

Крім того, при розробці програмно-апаратного комплексу важливо враховувати його призначення та конкретні потреби користувачів. Наприклад, якщо система призначена для вбудованих застосувань або IoT проектів, то вона повинна бути енергоефективною та стабільною у роботі на низьких рівнях енергоспоживання. У такому випадку, оптимізація алгоритмів та ресурсів Raspberry Pi стає ще важливішою, оскільки доводиться працювати з обмеженими можливостями, щоб забезпечити найкращу продуктивність та функціональність пристрою. Доцільно також розглянути можливість використання додаткових модулів та периферійних пристроїв, таких як камери, сенсори руху, чи модулі зв'язку, для розширення можливостей та функціональності системи. Врахування цих аспектів сприятиме створенню більш ефективних та користувацьких програмно-апаратних рішень на базі Raspberry Pi.

Щодо використання глибокого навчання, це потужний інструмент у галузі комп'ютерного зору. Використання згорткових нейронних мереж може допомогти високоточному розпізнаванні обличчя, а мережі довготривалої пам'яті можуть покращити відстеження рухів та аналіз динамічних змін. Оскільки система передбачає використання для відстеження та розпізнавання, вибрані алгоритми мають бути оптимізовані для мінімізації часу обробки. Використання методів

оптичного потоку та оптимізація коду для використання апаратних можливостей Raspberry Pi можуть значно покращити швидкість системи. Також, важливо враховувати можливості розширення платформи Raspberry Pi за допомогою додаткових модулів та інтерфейсів, що дозволяє розробляти більш складні програмно-апаратні рішення.

3.1 Врахування особливостей Raspberry Pi

Перш за все, важливо враховувати особливості апаратно-програмної платформи Raspberry Pi. Цей одноплатний комп'ютер має обмежені ресурси, такі як потужність обчислень, обсяг оперативної пам'яті та можливості графічного прискорення. Тому вибір методів та алгоритмів повинен бути оптимізований для працездатності на цьому пристрої. Розробка програмно-апаратного комплексу на Raspberry Pi вимагає особливої уваги до ефективного використання обмежених ресурсів. Проаналізовані методи повинні бути адаптовані до специфічних характеристик пристрою, зокрема, його відносно скромної обчислювальної мережі та доступної оперативної пам'яті.

3.1.1 Застосування глибокого навчання

Глибоке навчання є потужним інструментом у галузі комп'ютерного зору. Використання згорткових нейронних мереж може допомогти у високоточному розпізнаванні обличчя, а використання мереж довготривалої пам'яті може поліпшити відстеження рухів та аналіз динамічних змін. Застосування методів оптичного потоку та оптимізація коду для використання апаратних можливостей Raspberry Pi можуть покращити швидкість системи.

3.1.2 Забезпечення надійності та відмовостійкості

Важливою вимогою є забезпечення надійності та стійкості системи в різних умовах експлуатації. Обрані методи повинні виявлятися стійкими до змін освітлення, різних поз та масштабування обличчя, щоб забезпечити

консистентність та точність розпізнавання. Однак важливо також враховувати можливість працездатності системи в різних середовищах, включаючи змінні умови освітлення та різноманітні контексти використання.

При виборі інструментів для розробки слід акцентувати увагу на зручності використання, продуктивності та підтримці специфічних функцій. Використання бібліотек OpenCV та Dlib може надати необхідний набір інструментів для візуалізації та обробки зображень, враховуючи їхню широку функціональність та ефективність в обробці відомих задач комп'ютерного зору.

Обрані інструменти повинні відповідати вимогам системи щодо стійкості та продуктивності, а їхні можливості повинні дозволяти легко і ефективно враховувати різноманітні умови реального світу для досягнення оптимальної працездатності системи розпізнавання рухів та обличчя на платформі Raspberry Pi.

3.2 Опис одноплатного комп’ютеру Raspberry Pi

Raspberry Pi базується на системі-на-чипі (SoC) Broadcom BCM2835, що включає процесор ARM із тактовою частотою 700 МГц, графічний процесор VideoCore IV, і 512 чи 256 мегабайтів оперативної пам'яті. Відсутній вбудований жорсткий диск, замість нього використовується SD карта. Ця апаратна конфігурація дозволяє відтворювати відео у форматі H.264 у роздільній здатності 1080p та запускати комп'ютерні ігри, наприклад, Quake III Arena.

Ініціатором проекту Raspberry Pi є британський благодійний фонд Raspberry Pi Foundation. Комп'ютер був задуманий як засіб для навчання програмуванню дітей, але отримав популярність і в інших галузях, таких як створення домашніх медіацентрів. Найбільш доступна модель Raspberry Pi продається без корпусу і має форму фізичної плати, подібної до кредитної картки, з вагою 45 грамів.

Комп'ютер використовує процесор на архітектурі ARM з тактовою частотою 700 мегагерц, має роз'єм для навушників і слот для карти пам'яті. Молодша (A) і старша (B) моделі Raspberry Pi відрізняються об'ємом оперативної пам'яті (256 і 512 мегабайтів відповідно) і кількістю USB-портів (один проти двох). Крім того, старша модель має роз'єм Ethernet 10/100, тоді як молодша споживає на

третину менше енергії. Старша модель Raspberry Pi була представлена до продажу наприкінці лютого 2012 року за ціною 35 доларів США. У лютому 2013 року в Європі з'явилася молодша модель комп'ютера Raspberry Pi вартістю 25 доларів США. Комп'ютери Raspberry Pi доступні для придбання у дистриб'юторів RS Components та element14. З лютого 2012 року було продано понад мільйон пристроїв.

У даній кваліфікаційній магістерській роботі використовується модель одноплатного мікрокомп’ютеру Raspberry Pi версії 4 Model B. Він є новинкою у популярному асортименті мікрокомп'ютерів Raspberry Pi. Ця плата має значні покращення порівняно з попереднім поколінням Raspberry Pi 3 Model B+, включаючи підвищення швидкості процесора, поліпшену продуктивність мультимедійного модуля, збільшення обсягу та швидкості оперативної пам'яті і оновлені мережеві модулі, при цьому зберігаючи рівень енергоспоживання та сумісність з периферійними модулями.

Для кінцевого користувача Raspberry Pi 4 Model B забезпечує продуктивність стаціонарних рішень початкового рівня на архітектурі X86. Ключові характеристики цього продукту включають:

а) новий, високопродуктивний 64-розрядний чотирьохядерний процесор;

б) підтримка двох дисплеїв з роздільною здатністю до 4K через пару micro-HDMI портів;

в) апаратне декодування відео в 4Kp60;

г) від 1 до 4 ГБ оперативної пам'яті (в залежності від моделі);

д) двухдіапазонна бездротова мережа на 2,4 і 5,0 ГГц, Bluetooth 5.0;

е) Gigabit Ethernet;

ж) два порти USB 3.0;

з) підтримка PoE (Power over Ethernet) за допомогою окремого модуля PoE HAT.

Двоканальна бездротова LAN і Bluetooth мають модульну сертифікацію, що дозволяє використовувати плату у кінцевих продуктах зі значно спрощеним

тестуванням на відповідність стандартам, що поліпшує як вартість, так і час виходу на ринок.

Нижче приведена характеристика одноплатного мікрокомп’ютеру, який використовується у даній кваліфікаційній роботі:

а) процесор: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz;

б) оперативна пам’ять 4GB LPDDR4-2400 SDRAM;

в) підтримка 2.4 GHz та 5.0 GHz IEEE 802.11ac бездротової мережі, Bluetooth 5.0, BLE;

г) Gigabit Ethernet;

д) 2 x USB 3.0 порти;

е) 2 x USB 2.0 порти;

ж) роз'єм GPIO: стандартний 40-пиновий роз'єм GPIO, повністю сумісний з попередніми версіями плат;

з) підтримка мультимедіа:

1) підтримка H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode);

2) Графіка OpenGL ES 3.0;

и) підтримка SD карт: micro-SD слот для завантаження операційної системи та зберігання даних;

к) живлення комп’ютеру:

1) 5V DC через USB-C коннектор (мінімум 3A);

2) 5V DC через роз'єм GPIO (мінімум 3A);

3) Підтримка Power over Ethernet (PoE) (потрібен окремий PoE HAT);

л) відео та звук:

1) підтримка до 4kp60 через 2 × micro-HDMI порти;

2) 2-канальний MIPI DSI вихід для дисплею;

3) 2-канальний MIPI CSI порт для камери;

4) 4-полюсний стерео аудіо та композитний відео порт;

3.3 Опис камери для Raspberry Pi

У якості веб-камери для дослідження було обрано спеціальний модуль ширококутної камери 12МП IMX708 для Raspberry Pi з кутом огляду 120° FOV. Оновлений модуль із високим розширенням 12 МП, автофокусом та сенсором IMX708, з кутом огляду 120°. Цей модуль підключається безпосередньо до роз'єму камери на всіх моделях Raspberry Pi. Основні особливості:

а) використання сенсора IMX708, що забезпечує високодеталізоване та реалістичне зображення;

б) просте підключення до різних версій Raspberry Pi, включаючи 4B/3B+/3A+/3B/2B/B+/A+.

Технічні характеристики:

а) сенсор: IMX708;

б) кількість пікселів: 11,9 МП (4608 × 2592);

в) фокусування: контролер I2C з підтримкою автофокуса;

г) розмір матриці: КМОП 7,4 мм;

д) діафрагма: F2.2;

е) фокусна відстань: 2,75 мм;

ж) поле зору: 120°.

Вигляд камери можна побачити на рисунку 2.9.

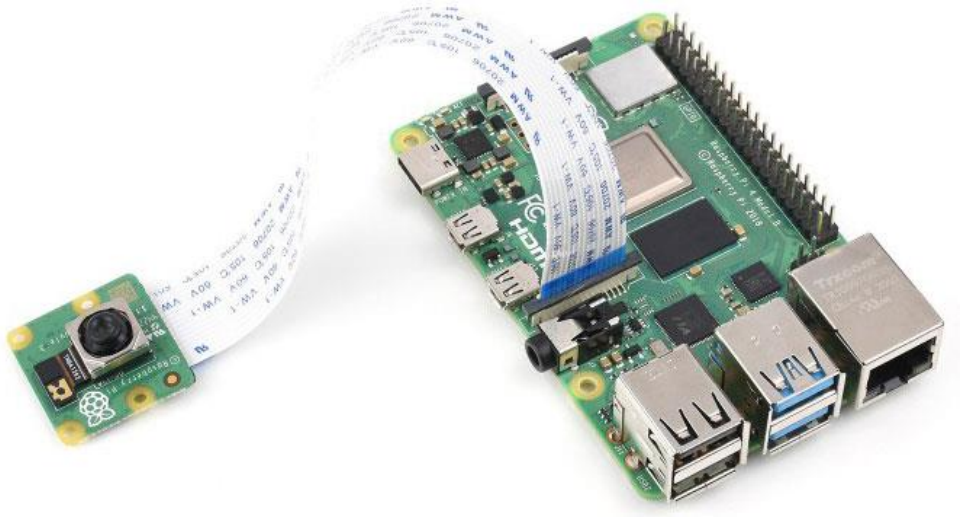


Рис. 2.9 – Модуль ширококутної камери 12МП IMX708 разом з Raspberry Pi

3.4 Схема розроблюваної системи

Основна мета схеми системи полягає в уявленні елементів системи або компонентів, які діють на етапі виконання. Дану схему буде подано у вигляді діаграми розготання. Основними символами на цій діаграмі є вузли, компоненти і зв'язки між ними, які позначені мітками, що вказують на фізичні з'єднання. Вузол на діаграмі представляє будь-який фізичний елемент системи, такий як камера або пристрій Raspberry Pi. На рисунку 3.1 показано діаграму розгортання розробленої системи.

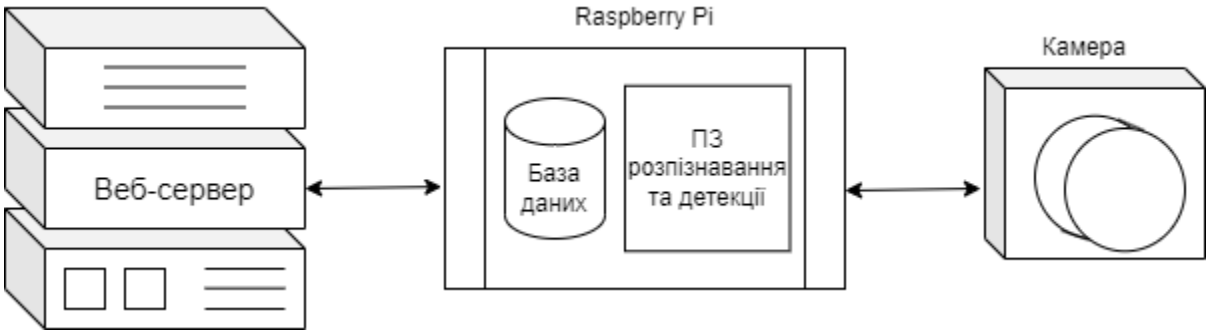


Рис. 3.1 – схема апаратно-програмного комплексу

Інформаційний набір, який створено в результаті вирішення поставленої задачі повної автономності і незалежності від мережі Інтернет, зберігається локально. Модель реляційної бази даних MySQL складається з двох пов’язаних зв’язком один до багатьох таблиць. На пристрої Raspberry Pi зберігаються набори фотографій для кожної зареєстрованої адміністратором особи.

Важливим аспектом є те, що відповідно до спроектованої системи, назви папок та поле "name" в таблиці "person" є унікальним ідентифікатором, який складається з прізвища та імені особи. Інформація про нового користувача додається одночасно до локального диску та бази даних. Така мінімальна інформаційна структура була створена, оскільки система не передбачає збору та збереження великої кількості особистих даних. Повна інформаційна структура розробленої системи, описана вище, наведена на рисунку 3.2.

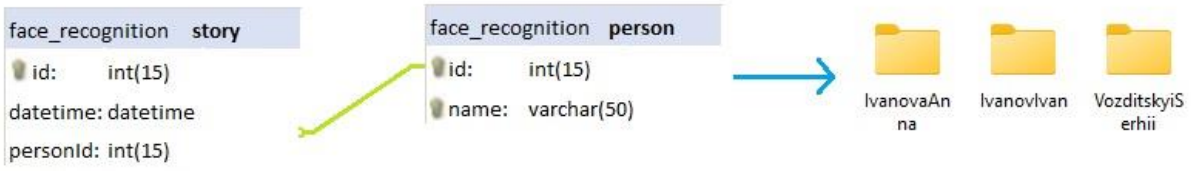


Рис. 3.2 – структура програмного забезпечення

На рисунку зображено зв'язок між іменами зареєстрованих осіб, які зберігаються локально на диску, та іменами в таблиці бази даних MySQL. Цей зв'язок підтримується на рівні програмної реалізації, тобто кожен раз, коли необхідно видалити особу, це робиться через спеціальну команду, про яку буде описано далі. Після введення даної команди всі дані про особу видаляються як з диску, так і з бази даних. Також у такому випадку відбувається і повне видалення історії для цієї особи, що здійснюється через існування спеціального типу зв'язку між таблицями.

Модуль програми `face_taker.py` відповідає за процес створення бази осіб. Під час виконання програми, користувачу потрібно ввести ідентифікатор, який буде використовуватися для збереження 30 фотографій. Ці фотографії використовуватимуться для подальшого розпізнавання конкретної особи. Після визначення особи, алгоритм автоматично зберігає зображення за форматом "Users.id.count.jpg", де "id" - це ідентифікатор, введений користувачем, а "count" - номер зображення. Робота алгоритму завершується, якщо оброблено 30 зображень.

REST сервер, побудований за допомогою фреймворку Flask мови програмування Python, є центральним компонентом системи, який взаємодіє з усіма іншими її складовими. Саме на сервері відбувається передача даних до бази даних MySQL та відправлення запитів на отримання даних з бази даних, яка, подібно до сервера, знаходиться на пристрої Raspberry Pi. Головною причиною розташування цих компонентів на одному пристрої є поставлена задача розробити автономну систему, яка не залежить від підключення до мережі Інтернет.

Центральною частиною сервера є REST controller, де визначені всі можливі URL-шляхи для подання запитів. Завдяки такій структурі ми можемо взаємодіяти з базою даних за допомогою методів GET, POST, DELETE тощо. Відповіді відправляються у форматі JSON.

Варто зауважити, що сервер розроблений таким чином, що при успішному виконанні запиту відправляється відповідь компоненту про правильну роботу, із кодом 200. У випадку, якщо запит не відбувся через якісь причини, буде повернутий код помилки, наприклад, код 403 (сервер недоступний), або 501 (метод не розпізнаний або не вдалося його виконати).

Саме на сервері відбувається з'єднання з базою даних MySQL. Для цього в коді введені конфігураційні дані, які зображені на рисунку 3.3, такі як хост, порт, пароль, назва бази даних тощо.

```
3   config = {
4       'host': '127.0.0.1',
5       'port': 3306,
6       'user': 'root',
7       'password': 'root',
8       'database': 'face_recognition'
9   }
```

Рис. 3.3 – конфігурація підключення

3.5 Опис датасету для навчання

Розробка програмного забезпечення відбувалась на мові програмування Python, що є інтерпретованою мовою, з використанням бібліотеки OpenCV. Основні завдання, пов'язані з розпізнаванням облич, включали створення бази даних осіб, навчання класифікатора та досягнення високої точності ідентифікації. Оскільки алгоритм вимагає навчання класифікаторів, потрібно було підготувати відповідні набори позитивних та негативних зображень.

Важливо відзначити, що OpenCV постачається з тренувальними даними та детектором, що спрощує процес розробки. Ця бібліотека дозволяє створювати

власні класифікатори для навчання будь-яких об'єктів, таких як автомобілі або літаки. Крім того, OpenCV вже містить багато попередньо навчених класифікаторів для розпізнавання облич, очей, посмішок та інших атрибутів.

Спочатку необхідно підготувати датасет - колекцію зображень, з якою ми будемо працювати [31, 32]. Для кожної особи в цьому датасеті створюється окрема тека, в якій зберігаються фотографії. Після того, як фотографії готові, ми запускаємо скрипт, який просканує всі зображення і створює базу розпізнаних осіб. У результаті ми отримуємо карту координат точок обличчя для кожного зображення (для кожної особи може бути кілька карт, по одній для кожної фотографії). Чим більше фотографій однієї і тієї ж особи зроблено з різних ракурсів, тим краще система навчиться розпізнавати її під різними кутами, наприклад, на відео.

Що відбувається далі? З веб-камери приходить зображення на мікрокомп’ютер. Спочатку OpenCV визначає, чи є на зображенні обличчя. Якщо обличчя виявлені, ми отримуємо їхні координати і вирізаємо лише цю частину зі загального кадру, передаючи її на розпізнавання та порівняння з базою даних - ми шукаємо відповідність. В кінцевому результаті на кадрі з'являється прямокутник з ім'ям особи або написом "невідома особа" та іншим. Для детекції рухів в OpenCV є спеціальні методи та функції, які дозволяють виявляти зміни в кадрах відеопотоку. Один з поширених підходів - це використання алгоритму оптичного потоку, такого як алгоритм Lucas-Kanade або алгоритм Farneback. Ці алгоритми дозволяють виявляти рух об'єктів у кадрі шляхом аналізу змін в інтенсивності пікселів між послідовними кадрами. У такій конфігурації швидкість обробки відео складає приблизно 2 кадри в секунду. Чим більше облич буде виявлено на зображенні, тим більше обчислень буде зроблено для кожного кадру (карта обличчя містить 68 точок), що призведе до зниження продуктивності системи.

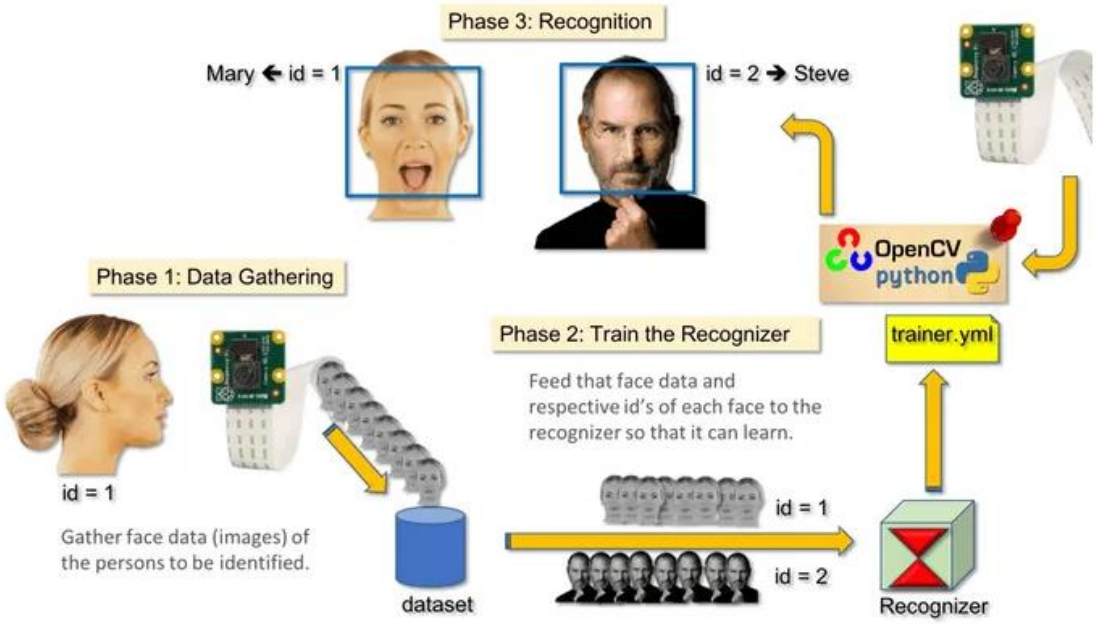


Рис. 3.4 – загальна схема комплексу

Крім того, важливо відзначити, що для ефективної роботи системи необхідно враховувати різноманітні умови освітлення та ракурси фотографій, які можуть змінюватися в залежності від умов зйомки. Чим більше різноманітних умов враховано під час підготовки датасету, тим більш адаптована система буде до реальних умов використання. Ретельна підготовка датасету забезпечить стабільну роботу системи та збільшить точність розпізнавання обличчя в різних умовах.

3.5.1 Навчання класифікатору

У бібліотеці OpenCV використовуються три алгоритми розпізнавання облич: Eigenfaces, Fisherfaces та LBPH. Кожен із цих алгоритмів підтримує навчання на основі заданого масиву зображень, передбачення знайденої особи, а також можливість завантаження та збереження стану моделі у форматах XML або YAML. Алгоритм LBPH, який був обраний для даної роботи, дозволяє оновлення моделі, що означає, що він може оновлювати існуючу модель новими зображеннями. Це впливає на якість розпізнавання, оскільки модель може вдосконалюватись з кожним доданим зображенням.

Скрипт train_model.py призначений для аналізу та обробки зображень, які знаходяться у папці dataset. Після обробки створюється відповідність між іменами

та особами, яка зберігається у файлі encodings.pickle. У цьому файлі зберігаються представлення всіх облич у вигляді 128-вимірних векторів.

Оскільки скрипт train_model.py працює з фотографіями у папці dataset, то фотографії для зручності зберігаються в папках з унікальними іменами, які складаються із прізвища та імені особи. Результати тренування можна побачити на рисунку 3.4. Графіки точності та втрат, які відображаються під час тренування системи виявлення маски для обличчя, свідчать про високу точність і лише незначні ознаки перетренування моделі на наявних даних. Графіки відображають надійність та ефективність алгоритму навчання, а також показують, що модель має стабільні показники точності і мінімальні втрати під час тренування. Це свідчить про те, що модель адекватно вивчає особливості даних і може успішно розпізнавати обличчя з маскою без надмірного врахування шуму або несуттєвих деталей.

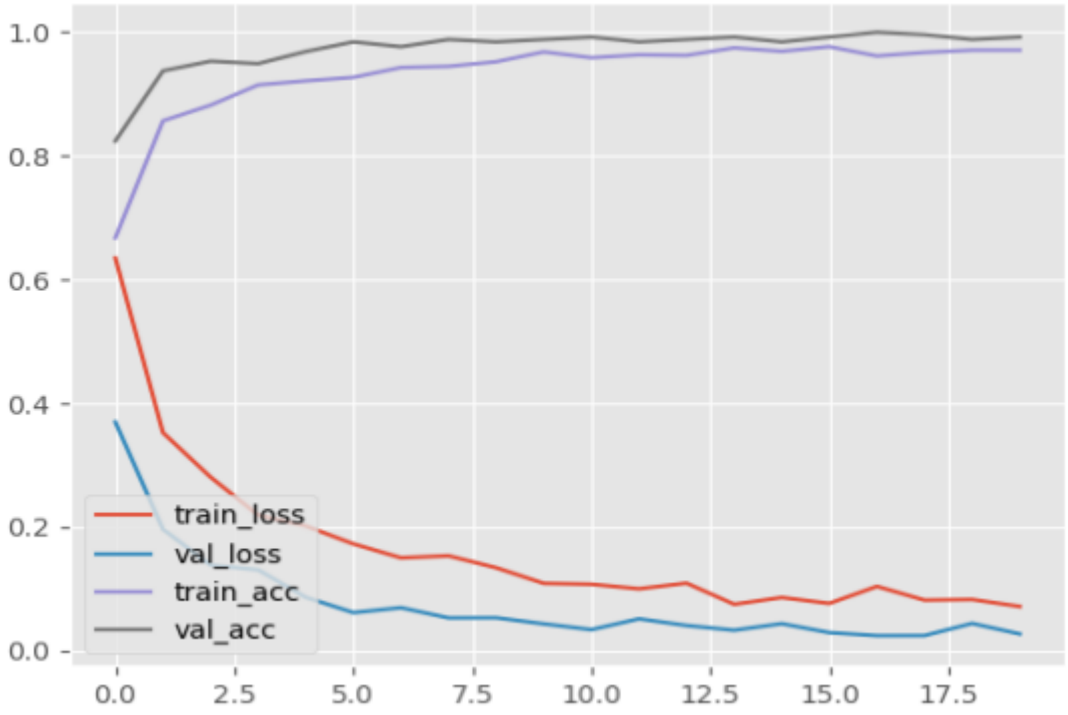


Рис. 3.5 – криві точності

Основним інструментом для розпізнавання облич у системі є скрипт facial_req.py. У цьому скрипті застосовується класифікатор разом з файлом

haarcascade_frontalface_default.xml для проведення розпізнавання облич на відеопотоці. Для проведення розпізнавання, спочатку завантажуються заздалегідь оброблені фотографії всіх зареєстрованих осіб. Потім ці фотографії порівнюються з обличчями, виявленими на відеопотоці з камери. Для реалізації розпізнавання використовується імпортований з бібліотеки OpenCV детектор CascadeClassifier. Далі проводиться його налаштування, включаючи встановлення параметрів, таких як мінімальні розміри облич на зображенні та інші необхідні дані. За допомогою коефіцієнта tolerance вказується точність розпізнавання, яка спрямована на зменшення можливості виявлення помилкових позитивних результатів.

Чим менший коефіцієнт tolerance, тим більш високі вимоги встановлені для порівняння системою облич з відеопотоку, який передається в реальному часі, та збережених фотографій облич зареєстрованих осіб. Коли програма виявляє точний збіг облич, відбувається автоматичний запит для внесення в базу даних інформації про час фіксації особи на пункті проходу. Крім того, зображення обличчя виділяється у вікні камери, що відкривається з запуском скрипта, рамкою та підписом імені.

3.6 Розробка компонентів апаратно-програмного комплексу

На операційній системі Windows 10 доступне вбудоване програмне забезпечення, яке дозволяє підключатися до пристроїв за їх IP-адресами, використовуючи логін та пароль, що були задані в налаштуваннях Raspberry Pi під час встановлення операційної системи. Це надає можливість керувати операційною системою, яка запущена у вигляді вікна програми. Підключення до пристрою здійснюється через захищений мережевий протокол SSH (Secure Shell) за допомогою Wi-Fi. Тому виникає необхідність підключення пристрою, яке забезпечує зв'язок між операційною системою та Raspberry Pi у межах однієї локальної мережі. Найпростіший спосіб для створення такого з'єднання - використання роутера.

3.6.1 Опис роботи розпізнавання облич

Для початку необхідно підключити камеру до плати. Після цього необхідно увімкнути можливість використання камери. Це можна зробити в налаштуваннях, перейшовши у вкладку "Interfaces" та увімкнувши опцію "Camera" (клацнувши на "Enabled").

Для базових маніпуляцій з камерою на Raspberry Pi існують три утиліти командного рядка, які встановлені в системі: raspivid, raspistill і raspiyuv для захоплення відео та фотографій. Створення файлу motion_detector та імпорт необхідних пакетів, включаючи imutils, є першими кроками в розробці системи виявлення руху та розпізнавання облич. Після цього розглянемо обробку аргументів командного рядка та налаштуємо програму таким чином, щоб вона працювала з веб-камерою за замовчуванням, якщо не вказано інше.

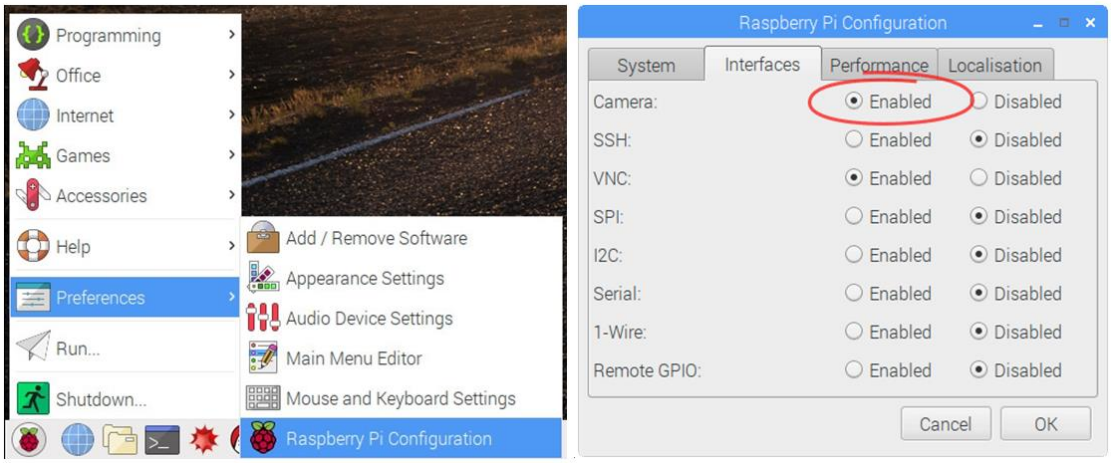


Рис. 3.6 – підключення до камери на мікрокомп’ютері

Під час аналізу та навчання на наборі даних, траплялися фотографії людей, які носять окуляри або медичні маски. І якщо в більшості випадків окуляри не заважають алгоритму розпізнавання облич, то маска тоді може значно змінити результат алгоритму. На прикладі нижче, наша модель також навчена розпізнавати чи носить людина маску, яка може заважати повноцінному розпізнаванню. Для відображення та обробки зображень, наступним кроком у розробці програми може

бути аналіз аргументів командного рядка. Цей процес дозволяє користувачам передавати параметри до програми при її запуску з командного рядка.

Аналіз аргументів командного рядка є важливим, оскільки він дозволяє програмі змінювати своє поведінку залежно від переданих параметрів. Наприклад, користувач може вказати шлях до папки з зображеннями для обробки або ввести додаткові параметри для налаштування роботи програми. Для аналізу аргументів командного рядка в Python існують різні бібліотеки, такі як argparse. Ці бібліотеки дозволяють зручно обробляти введені користувачем параметри та визначати їхні значення для подальшого використання у програмі.

Розглянемо кожен з аргументів командного рядка, які ми навели:

а) image: цей аргумент представляє шлях до вхідного зображення, на якому містяться грані, що використовуються в умові для води або будь-яких інших аналізів;

б) face: цей параметр вказує шлях до каталогу моделі системи виявлення облич. Його використання передбачає локалізацію облич перед класифікацією;

в) model: аргумент вказує шлях до моделі системи виявлення маски для облич, яку ви навчили раніше. Він використовується для подальшого аналізу або обробки зображень з обличчями;

г) впевненість (confidence): це необов'язковий параметр, який дозволяє встановити поріг ймовірності для фільтрації слабких виявлень обличчя. Якщо виявлені обличчя мають ймовірність нижче вказаного порогу, вони можуть бути відфільтровані.

Ці аргументи дозволяють користувачеві зручно налаштовувати роботу програми, визначаючи вхідні дані та параметри аналізу зображень з обличчями.

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
help="path to input image")
ap.add_argument("-f", "--face", type=str,
default="face_detector",
help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
default="mask_detector.model",
help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

Рис. 3.7 – аналіз аргументів командного рядка

Після цього ми будемо завантажувати наші моделі для системи виявлення облич та класифікатора масок.

```
# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
"res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
model = load_model(args["model"])
```

Рис. 3.8 – код модуля виявлення

Дякуючи нашим моделям глибокого навчання, які вже знаходяться в пам'яті, ми переходимо до завантаження та передобробки вхідного зображення.

```
# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]
# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
(104.0, 177.0, 123.0))
# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()
```

Рис. 3.9 – процес завантаження та попередньої обробки

Після завантаження зображення з диска, ми створюємо його копію та отримуємо розміри кадру для подальшого масштабування та відображення. Попередня обробка виконується за допомогою функції blobFromImage в OpenCV. В параметрах функції ми зменшуємо розмір зображення до 300 × 300 пікселів і виконуємо середнє віднімання. Далі ми виконуємо виявлення обличчя для локалізації місць на зображенні, де розташовані обличчя. Після цього ми перевіряємо, що впевненість (confidence) кожного виявленого обличчя відповідає встановленому пороговому значенню, перш ніж витягувати області інтересу (ROI).

Ми перебираємо всі виявлені обличчя і отримуємо їх впевненість, щоб встановити поріг - confidence. Потім обчислюємо значення обмежувального поля для певної грані, переконуючись, що це поле потрапляє в межі зображення. Далі ми застосовуємо нашу модель MaskNet для оцінки ефективності вкладень.

Наступним кроком ми витягаємо область інтересу обличчя за допомогою бібліотеки NumPy. Після цього ми застосовуємо до ROI ту ж попередню обробку, яку застосовували під час навчання моделі. Після цього ми використовуємо модель для виявлення маски, щоб передбачити, чи на зображенні присутня маска (with_mask) чи відсутня (without_mask).

```
face = image[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (224, 224))
face = img_to_array(face)
face = preprocess_input(face)
face = np.expand_dims(face, axis=0)
(mask, withoutMask) = model.predict(face)[0]
```

Рис. 3.10 – обрахуванння значення обмежувального поля для конкретної грані

Спочатку ми визначаємо мітку класу на основі ймовірностей, які повертає модель системи виявлення маски, і призначаємо пов'язаний колір для анотації. Колір буде «зеленим» для з маскою (with_mask) та «червоним» для без маски (without_mask). Потім ми малюємо текст мітки, а також обмежувальний прямокутник для обличчя, використовуючи функції малювання OpenCV. Після того, як всі виявлення будуть оброблені, відображається вихідне зображення. Як можна помітити, система виявлення маски для обличчя правильно позначила цезображення яке містить маску.

```
label = "Mask" if mask > withoutMask else "No Mask"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

cv2.putText(image, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
cv2.imshow("Output", image)
cv2.waitKey(0)
```

Рис. 3.11 – алгоритм детекції маски

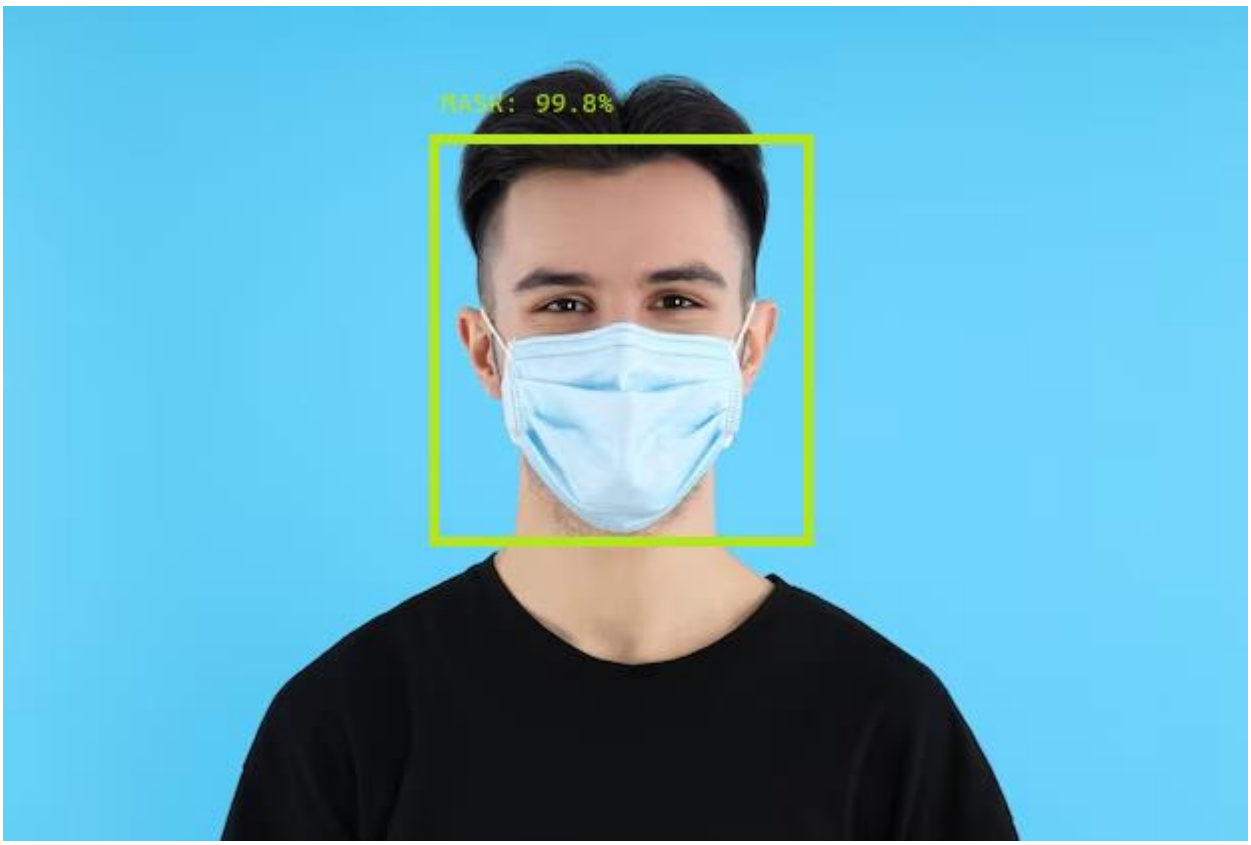


Рис. 3.12 – розпізнана маска на фото

Тепер, коли за наявність маски можна не хвилюватись, час перейти до звичайного розпізнавання облич. Для керування системою усі дії здійснюються через спеціальні скрипти та програми. Одним з основних скриптів є headshots.py, який дозволяє адміністратору додавати нові особи до бази даних системи. Після запуску цього скрипта користувачу надається можливість вибрати опцію реєстрації нового користувача або видалення існуючого. Під час реєстрації нового користувача система використовує камеру для отримання фотографій обличчя з різних кутів з метою покращення процесу розпізнавання.

Файл train_model.py використовується для аналізу фотографій у папці dataset. Він аналізує зображення користувачів, які були раніше оброблені, швидко, але обробка нових зображень може займати значний час. Основний процес розпізнавання відбувається у скрипті facial_req.py, де адміністратор може бачити зображення з камери в та обличчя зареєстрованих осіб, які автоматично розпізнаються системою.

Результати розпізнавання автоматично передаються до сервера. Однак, після реєстрації нового користувача необхідно перезапустити скрипт facial_req.py або перезавантажити пристрій для коректної роботи процесу розпізнавання. Автоматичний перезапуск пристрою один раз на добу може забезпечити безперервну роботу системи, щоб всі зареєстровані користувачі могли користуватися нею наступного дня.

Тепер залишається відправити дані на сервер та у базу даних. Робота адміністратора з сервером обмежена. Зокрема, він може переглядати історію запитів усіх інших компонентів до сервера. У консолі виводяться такі дані:

а) назва запиту;

б) час запиту;

в) IP-адреса компонента, з якого відбувся запит;

г) код відповіді сервера.

Передбачається, що сервер автоматично запускається кожного разу, коли вмикається пристрій Raspberry Pi разом з компонентом розпізнавання. Проте існує можливість вручну запустити його через консоль, введенням кількох команд.

3.6.2 Опис роботи детекції рухів

Як це вже було зроблено вище, ми також визначимо мінімальну область, яка вважатиметься мінімальним розміром (в пікселях) для зони зображення, щоб визначити фактичний рух. Часто ми спостерігаємо невеликі зміни на зображенні, які можуть бути спричинені шумом або змінами умов освітлення. Насправді, ці дрібні зміни зображення не вказують на рух, тому ми встановимо мінімальний розмір області для фільтрації шуму та виключення цих помилкових спрацьовувань. Далі ми створимо посилання на наш об'єкт. Якщо шлях до відеофайлу не вказаний, програма автоматично встановить шлях до веб-камери.

```
from imutils.video import VideoStream
import argparse
import datetime
import imutils
import time
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())
if args.get("video", None) is None:
    vs = VideoStream(src=0).start()
    time.sleep(2.0)
else:
    vs = cv2.VideoCapture(args["video"])
firstFrame = None
```

Рис. 3.13 – імпортування залежностей

У завершенні цього фрагменту коду ми визначимо змінну firstFrame, в якій буде зберігатися перший кадр потоку відеофайлу або веб-камери. Перший кадр нашого відеофайлу зазвичай не містить руху, лише фон, тому ми можемо використати його для моделювання фону нашого відеопотоку.

```
while True:
    frame = vs.read()
    frame = frame if args.get("video", None) is None else frame[1]
    text = "Unoccupied"
    if frame is None:
        break
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)
    if firstFrame is None:
        firstFrame = gray
        continue
```

Рис. 3.14 – ініціалізація першого кадру відеопотоку

Обробка зображення складається з кількох етапів, починаючи з перетворення трьохканального зображення в колірній моделі RGB до одноканального зображення у відтінках сірого. Цю операцію легко виконати за

допомогою функції cvtColor() бібліотеки OpenCV. Ця функція отримує вихідне та цільове зображення у вигляді масивів, тип перетворення та кількість каналів. Для конвертації RGB зображення в зображення у відтінках сірого, тип перетворення встановлюється як CV_BGR2GRAY. Можна побачити приклад такого перетворення. Якщо встановити 0 як значення параметра кількості каналів, кількість каналів автоматично визначається на основі даних про вихідне та цільове зображення.

Для детекції об'єктів на зображенні необхідно здійснити пошук контурів. Виділення меж здійснюється за допомогою застосування до вихідного зображення детектора кордонів Кенні. Однак недолік застосування детектора кордонів Кенні полягає в тому, що фільтр, який використовується цим алгоритмом, сприйнятливий до шуму.

Обробка зображення включає кілька етапів, починаючи з виділення меж за допомогою детектора кордонів Кенні. Хоча цей метод дозволяє виділити об'єкти на зображенні, він чутливий до шуму, що може призвести до неточностей. Для підвищення точності операцій використовується згладжування зображення за допомогою функції розмиття по Гаусу. Параметри цієї функції включають ядро згортки, стандартне відхилення в напрямку осі X та Y, а також метод екстраполяції пікселів. Цей процес необхідний для створення маски зображення. Окремо визначається мінімальна область, яка вважається фактичним рухом на зображенні. Це допомагає відфільтрувати невеликі зміни, що можуть бути зумовлені шумом або змінами умов освітлення.

Далі, при початку аналізу кожного кадру в рядку, отримуємо посилання на відеопотік. Визначається рядок, який вказує на нерухомий простір у відеопотоці. Якщо на відео спостерігається активність, цей рядок оновлюється. Якщо кадр не може бути прочитаний, цикл завершується. Для подальшої обробки кадру змінюється його розмір до 500 пікселів, щоб уникнути обробки великих зображень. Різниця між двома кадрами обчислюється як абсолютне значення їхніх різниць в інтенсивності пікселів.

Наступним кроком встановлюється поріг, щоб виявити області зі значними змінами інтенсивності пікселів. Пікселі з невеликими змінами відкидаються, а інші виділяються як передній план, показуючи де відбувається рух. Після виділення переднього плану і фону на зображенні, ми можемо продовжувати обробку для подальшого аналізу руху або об'єктів на ньому. Наприклад, можна використовувати алгоритми обробки зображень для визначення контуру об'єктів на передньому плані, використовуючи отриману маску. Це може включати застосування методів знаходження контурів та аналізу їх параметрів, таких як площа, периметр, або орієнтація, для визначення характеристик об'єктів та їхнього руху.

Далі можна реалізувати різноманітні алгоритми відстеження об'єктів, які включають в себе аналіз траєкторій руху, прогнозування майбутнього положення об'єктів, і виявлення зіткнень або взаємодій між ними. Крім того, можна реалізувати алгоритми виявлення аномальної поведінки або подій на відеозапису, використовуючи отриману інформацію про рух та об'єкти на зображенні. У цілому, подальший аналіз може включати в себе використання широкого спектру алгоритмів машинного навчання та комп'ютерного зору для розв'язання конкретних завдань в обробці зображень та відео.

У циклі по кожному контуру ми можемо застосовувати різноманітні фільтри, щоб відсіяти невеликі або нерелевантні контури, які не є об'єктами інтересу. Наприклад, ми можемо використовувати поріг щодо площі контуру, щоб відфільтрувати занадто малі області, які, ймовірно, є шумом чи випадковими дефектами на зображенні. Якщо площа контуру перевищує мінімально допустимий розмір, ми можемо малювати обмежуючий прямокутник або іншу форму, що відображає межі переднього плану або область руху на зображенні. Це дозволить нам візуалізувати області, які ми визначили як потенційно цікаві для подальшого аналізу або взаємодії. Після фільтрації контурів та визначення їх обмежень ми можемо переходити до подальших операцій, таких як аналіз динаміки руху,

виявлення об'єктів або подій, а також взаємодія з іншими системами або додатками, що використовують цю інформацію.

```
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
    (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
    cv2.imshow("Security Feed", frame)
    cv2.imshow("Thresh", thresh)
    cv2.imshow("Frame Delta", frameDelta)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
    break

vs.stop() if args.get("video", None) is None else vs.release()
cv2.destroyAllWindows()
```

Рис. 3.15 – формування рамки у рухомого об’єкта

Щодо подальшого використання отриманих даних, їх можна зберігати на карту MicroSD Raspberry Pi або транслювати за допомогою відповідних утиліт для подальшого аналізу, зберігання або трансляції відеопотоку на сервер або веб-браузер. Сповіщення оператора про порушників також може бути реалізовано за допомогою різних методів, таких як електронні листи, повідомлення на мобільний пристрій або спеціалізовані системи сповіщення.

Висновки до розділу 3

У цьому розділі ми розглянули процес розробки системи виявлення руху та розпізнавання облич з використанням Raspberry Pi та камери. Ми розпочали з встановлення необхідних пакетів і бібліотек, таких як OpenCV, для обробки зображень та виявлення руху. Після цього ми налаштували Raspberry Pi для роботи з камерою, сконфігурували отримання зображень та розробили алгоритм.

Ми розглянули важливі кроки такого процесу, як попередня обробка зображення, виявлення облич за допомогою моделей глибокого навчання, визначення областей інтересу, а також анотування та відображення результатів розпізнавання на зображенні.

Основні кроки виявлення руху включають перетворення зображень у відтінках сірого, застосування фільтрації для згладжування, визначення різниці між кадрами та встановлення порогу для виявлення руху. Ми також розглянули практичне застосування системи, таке як запис виявлених об'єктів, трансляція відеопотоку та сповіщення оператора про події.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Підсумок роботи програмно-апаратного комплексу

Виконавши усі необхідні дії, а саме проектування програмно-апаратного комплексу, вибір потрібних технологій, підбір апаратного забезпечення, яке зможе оброблювати весь процес, написання коду та проведення тестування, ми досягли повноцінного рішення завдання кваліфікаційної магістерської роботи, а саме розпізнавання облич та детекції рухів на базі одноплатного комп'ютеру Raspberry Pi версії 4B. Щодо частини роботи, яка стосується розпізнавання облич, то з аналізу описаного програмно-апаратного комплексу видно, що він розроблений для розпізнавання облич включаючи виявлення масок на них. Основні етапи роботи системи складають обробку зображення, локалізацію облич, виявлення масок та подальшу обробку результатів. Основні складові системи:

а) аналіз аргументів командного рядка, бо це важливий етап, який дозволяє користувачеві налаштовувати параметри аналізу зображень, використовуючи різні параметри, такі як шлях до зображення, шлях до моделі для виявлення облич, шлях до моделі для виявлення масок, а також поріг впевненості;

б) після аналізу аргументів система завантажує попередньо навчені моделі глибокого навчання для виконання виявлення облич;

в) зображення піддається попередній обробці, такі як зменшення розміру до 300x300 пікселів та середнє віднімання, так як це допомагає підготувати зображення для подальшого аналізу;

г) використовуючи навчені моделі, система виявляє облича на зображеннях, а також визначає, чи присутня маска на обличчі та потім розпізнає обличчя (рис. 4.1);

Рис. 4.1 – розпізнані обличчя

д) адміністратор має можливість працювати з системою через спеціальні скрипти та програми, такі як headshots.py для додавання нових користувачів та train_model.py для аналізу фотографій;

е) результати розпізнавання передаються на сервер і зберігаються у базі даних для подальшого аналізу та використання.

Процес обробки зображення складається з декількох етапів, які включають перетворення зображення в відтінки сірого, виділення контурів та згладжування зображення для підвищення точності аналізу. Для початку обробки зображення перетворюється з колірного (наприклад, RGB) відтінки сірого. Це зменшує обсяг даних і спрощує подальший аналіз. В бібліотеці OpenCV для цього використовується функція cvtColor() з параметром CV_BGR2GRAY. Після перетворення зображення в відтінки сірого як на рисунку 4.2, та застосовується детектор контурів, наприклад, детектор Кенні. Цей алгоритм дозволяє виявити межі об'єктів на зображенні, що буде використовуватися для подальшого аналізу.

Рис. 4.2 – перетворене зображення у відтінках сірого

Для підвищення точності та зменшення впливу шуму на результати аналізу застосовується фільтр Гаусса. Цей фільтр допомагає розмити зображення, зменшуючи вплив шуму і покращуючи якість результатів. Для виявлення руху на зображенні визначається мінімальна область, яка вважається фактичним рухом. Це допомагає відфільтрувати невеликі зміни, які можуть бути зумовлені шумом або змінами умов освітлення. Після виконання попередніх етапів можна проводити аналіз динаміки руху та об'єктів на зображенні. Це може включати в себе виявлення контурів, відстеження об'єктів та визначення їх параметрів, таких як швидкість руху, розмір або форма. На основі аналізу динаміки руху можна виявляти аномальну поведінку або події на зображенні. Це може включати в себе виявлення незвичайного руху, виникнення конфліктів або взаємодій між об'єктами.

Ми також встановимо мінімальну область, яка визначає найменший розмір у пікселях, який ми вважатимемо реальним рухом на зображенні. Часто ми спостерігаємо невеликі зміни в областях зображення, які можуть бути спричинені шумом або змінами в умовах освітлення. Однак ці дрібні області насправді не представляють собою руху, тому ми встановлюємо мінімальний розмір регіону і відсікаємо ці помилкові сигнали. Це дозволяє нам точніше визначити дійсний рух на зображенні та уникнути ложних виявлень.

Рис. 4.3 – маска фону

Ми обчислюємо різницю між двома кадрами, взявши абсолютне значення їхніх відповідних різниць у значеннях інтенсивності пікселів. Потім встановлюємо поріг “frameDelta” в рядку, щоб визначити області зображення, де відбулися значні зміни в інтенсивності пікселів. Якщо різниця менше 25, піксель відкидається і встановлюється чорним, наприклад фон. Якщо різниця перевищує 25, вона встановлюється в білий колір, тобто передній план. Це дає нам чорний фон зображення і білий передній план, що показує область, де відбувається рух. Потім ми переходимо до циклу по кожному контуру, де ми фільтруємо невеликі нерелевантні контури у відеопотоці. Якщо площа контуру перевищує мінімальну область, ми малюємо обмежуючий прямокутник або форму, яка оточує передній план і область руху.

4.2 Результат машинного навчання за дпомогою датасету

Набір даних містить понад 13 000 зображень облич, зібраних з Інтернету. Кожне обличчя має мітку з ім'ям особи на знімку. У наборі даних 1680 людей мають два або більше відмінних зображення. Єдиним обмеженням для цих облич є те, що вони були виявлені детектором облич Віоли-Джонса. Цей набір даних - це колекція зображень у форматі JPEG відомих осіб, зібраних з Інтернету. Кожне зображення сконцентроване на одному обличчі. Кожен піксель кожного каналу (колір у RGB) кодується у форматі float у діапазоні від 0,0 до 1,0. Задача називається розпізнаванням або ідентифікацією за допомогою зображення обличчя, треба

визначити ім'я особи за навчальним набором (галереєю). Вміст спеціального файлу, який описує кожне зображення для того, щоб програмно-апаратний комплекс міг навчатися на основі цих даних приведено нижче на рисунку 4.4.

```
<annotation>
    <folder>images</folder>
    <filename>maksssksksss100.png</filename>
    <size>
        <width>400</width>
        <height>226</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>189</xmin>
            <ymin>30</ymin>
            <xmax>245</xmax>
            <ymax>88</ymax>
        </bndbox>
    </object>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>387</xmin>
            <ymin>54</ymin>
            <xmax>400</xmax>
            <ymax>75</ymax>
        </bndbox>
    </object>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>118</xmin>
            <ymin>87</ymin>
            <xmax>163</xmax>
            <ymax>126</ymax>
        </bndbox>
    </object>
</annotation>
```

Рис. 4.4 – мета-дані кожного зображення

Оригінальні зображення мають розмір 250 x 250 пікселів, але за замовчуванням вони зменшуються до 62x47 пікселів за допомогою аргументів зрізу та зміни розміру. Приклади таких зображень приведені нижче. Можна також звернути увагу що є зображення на яких люди носять захисні маски.

Рис. 4.5 – зображення людей з масками

Рис. 4.6 – декілька зображень однієї людини без маски

Оптимізація моделі варіюється залежно від конкретного завдання, що виконується. Тут ми займаємались прогнозуванням безперервної змінної, яка порівнюється з класом. Цей аспект має значення, оскільки він змінює спосіб оптимізації моделі. Для проблеми з безперервною змінною головною задачею була мінімізація середньоквадратичної помилки. Ми тренували модель протягом 1500

епох і виводили втрати кожні 150 ітерацій. Згодом, ми змогли побудувати фактичні значення серії разом з передбаченими значеннями. Якщо наша модель виправлена, тоді передбачені значення слід розташувати поверх фактичних значень. Як можна побачити, модель має потенціал для вдосконалення. Зміна гіперпараметрів, таких як розмір вікна, розмір партії та кількість повторень нейронів, залежить від нас.

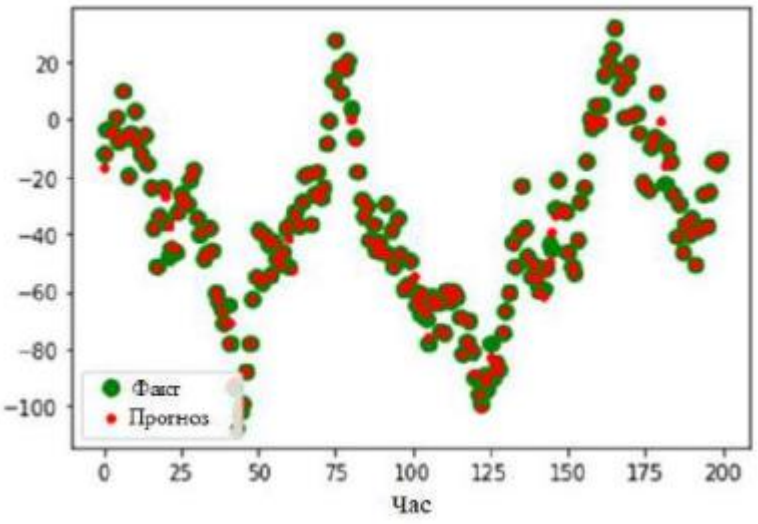


Рис. 4.7 – прогноз проти фактичних значень

4.3 Тестування програмно-апаратного комплексу

Наша система розпізнавання обличчя працює приблизно зі швидкістю 1-2 кадри на секунду. Більшість обчислень відбувається під час розпізнавання обличчя, а не під час його виявлення. Крім того, чим більше облич у наборі даних, тим більше порівнянь виконується для процесу голосування, що призводить до сповільнення розпізнавання облич.

Тому ми розглянули можливість обчислення повного розпізнавання облич (тобто витягнення 128-мерного відображення обличчя) один раз кожні N кадрів (де N - змінна, яку визначає користувач), а потім застосовували прості алгоритми відстеження, щоб відстежувати виявлені обличчя. Такий процес дозволив досягти швидкості 8-10 кадрів на секунду на Raspberry Pi для розпізнавання облич. Далі ми виконуємо скрипт, який аналізує всі зображення та формує базу розпізнаних осіб. Це призводить до створення карти координат точок обличчя для кожного

зображення. Для кожної особи може бути декілька карт, які відповідають різним фотографіям. Чим більше знімків однієї особи зроблено з різних ракурсів, тим краще система вивчає її розпізнавати, зокрема, на відео.

Нижче приведено розпізнані обличчя, які містились у датасеті у кожній папці з ім’ям зображеної людини. Для більш точного аналізу, тестувалося декілька зображень однієї й теж самої людини.

Рис. 4.8 – розпізнана особа під ім’ям Аарон Соркін

Рис. 4.9 – розпізнана особа під ім’ям Дональд Фер

Рис. 4.10 – розпізнана особа під ім’ям Джей Лено

Рис. 4.11 – розпізнана особа під ім’ям Кевін Спейсі

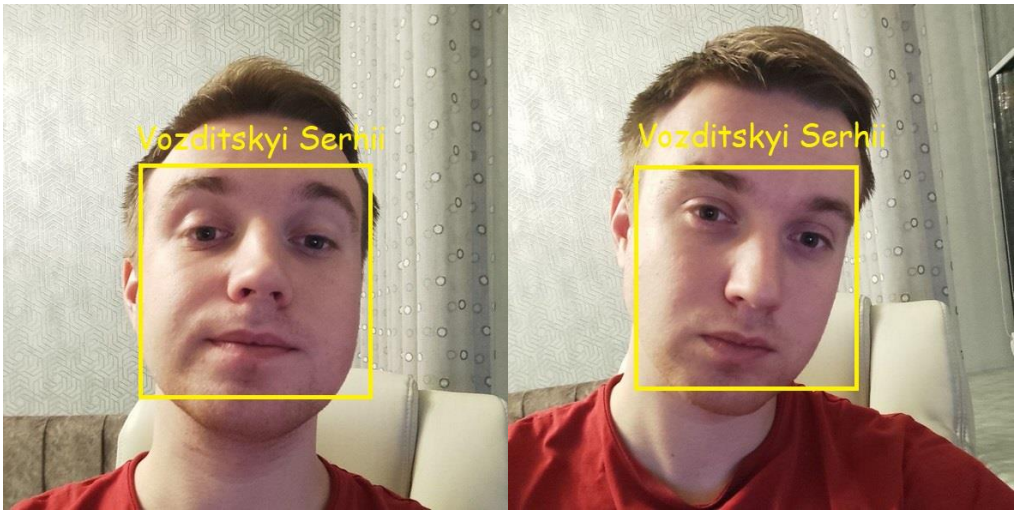


Рис. 4.12 – розпізнавання обличчя з камери Raspberry Pi

Висновки до розділу 4

Здійснюючи обробку зображень, ми використовуємо різноманітні техніки, починаючи з перетворення зображень у відтінки сірого для подальшого аналізу. Для виявлення руху на відео використовується порівняння двох кадрів та встановлення порогу для виділення областей зі значними змінами інтенсивності пікселів. Важливим етапом є визначення мінімального розміру області, що відображає фактичний рух, що дозволяє уникнути ложних спрацьовувань через шум чи малі зміни в освітленні.

Далі, з використанням алгоритмів обробки зображень, ми можемо аналізувати контури об'єктів та їх рух, що відкриває широкий спектр можливостей від відеоспостереження до відстеження об'єктів. Наприклад, аналіз руху на дорогах або в промислових умовах, виявлення аномальних подій та реагування на них. Застосування цих технік може покращити безпеку, ефективність та автоматизацію багатьох сфер, від безпеки до виробництва та медицини. Таким чином, обробка зображень та аналіз руху відіграють важливу роль у сучасних системах та додатках, що вимагають моніторингу та аналізу відеоданих.

ВИСНОВКИ

У сучасному інформаційному суспільстві швидкість розвитку технологій в області обробки зображень та розпізнавання обличчя зростає експоненційно, відкриваючи нові перспективи в багатьох галузях, включаючи безпеку, відеоспостереження та робототехніку. Поява потужних та доступних одноплатних комп'ютерів, таких як Raspberry Pi, створює можливості для створення програмно-апаратних комплексів для обробки відеоданих.

Огляд поточного стану інформаційних технологій у сфері комп'ютерного зору свідчить про їх великий потенціал та широкі можливості в різних сферах життя. Завдяки прогресу у розвитку апаратних засобів, алгоритмів машинного навчання та розширенню застосувань в промисловості, системи комп'ютерного зору стають надзвичайно важливими та перспективними технологіями для майбутнього.

Для досягнення цілей цієї магістерської роботи було вибрано таке обладнання: одноплатний комп'ютер Raspberry Pi разом з модулем ширококутної камери 12МП IMX708. Щодо програмного забезпечення, використовувалися мова програмування Python, бібліотека OpenCV, система управління базами даних MySQL, а також середовище розробки Microsoft Visual Studio Code.

У результаті виконання завдання кваліфікаційної магістерської роботи був розроблений програмно-апаратний комплекс з розпізнавання облич і детекції рухів на базі одноплатного мікрокомп’ютеру Raspberry Pi. Комплекс був протестований, що доказує, що він повністю виконує завдання які поставлені йому.

Розроблений програмно-апаратний комплекс має широкі можливості застосування у сферах відеоспостереження, контролю доступу, аналізу емоцій та робототехніки для створення інтерактивних систем. Ці новітні рішення сприяють покращенню якості життя та забезпеченню безпеки в різних сферах людської діяльності, роблячи їх більш ефективними та зручними для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Haidi Zhu, Xin Yan, Hongying Tang, Yuchao Chang. Moving object detection with Deep CNNs. 2015. URL: https://www.researchgate.net/publication/339174353_Moving_object_detection_with_Deep_CNNs (Last accessed: 01.12.2023).

2. Andrew Shepley. 2019. Deep Learning For Face Recognition: A Critical Analysis. URL: https://www.researchgate.net/publication/334783128_Deep_Learning_For_Face_Recognition_A_Critical_Analysis (Last accessed: 02.12.2023).

3. Yilun Wang, Xiangyu Zhang, Xiaogang Wang, Jian Sun. 2016. Face recognition with deep learning. URL: https://arxiv.org/ (Last accessed: 10.12.2023).

4. Caroline Dunn. 2022. URL: https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition (Last accessed: 11.12.2023).

5. Sravanti Chinta, Rajat Bothra Jain. 2022. Computer Vision Papers Literature Review. URL: https://hal.science/hal-03606640 (Last accessed: 11.12.2023).

6. Alain Granada. 2024. Scientific Landscape on the Computer Vision and Artificial Intelligence. URL: https://www.researchgate.net/publication/377221347_Scientific_Landscape_on_the_Computer_Vision_and_Artificial_Intelligence (Last accessed: 21.12.2023).

7. Posada J, Zorrilla M, Dominguez A et al. 2018. Graphics and Media Technologies for Operators in Industry 4.0[J] IEEE Computer Graphics and Applications. URL: https://ieeexplore.ieee.org/abstract/document/8474490/ (Last accessed: 10.12.2023).

8. Fengguang Jiang, Bolong Xiong, Chao Zhang. 2020. How to Achieve Strategic Breakthrough in Artificial Intelligence in China: A Comparison and Interpretation of Four Reports on Artificial Intelligence Development in China and the United States, pp. 3–4.

9. Brosnan Tadhg, Sun Da-Wen. 2004. Improving quality inspection of food products by computer vision. URL:

https://www.sciencedirect.com/science/article/pii/S0260877403001833 (Last accessed: 21.12.2023).

10. Poppe Ronald. 2006. Vision-based human motion analysis. URL: https://www.sciencedirect.com/science/article/pii/S1077314206002293 (Last accessed: 11.12.2023).

11. Ю. А. Тимошин, С. П. Орленко. 2018. Алгоритм Розпізнавання Обличчя Людей На Базі Згорткової Нейронної Мережі. URL: https://ela.kpi.ua/bitstream/123456789/34277/1/asau-2018-1_19.pdf (дата звернення 10.12.2023).

12. Face ID. URL: https://uk.wikipedia.org/wiki/Face_ID (Last accessed: 01.12.2023).

13. Raja R. 2017. Face Detection Using OpenCV and Python. URL: https://www.superdatascience.com/opencv-face-detection/ (Last accessed: 11.12.2023).

14. Raja R. 2017. Face Recognition Using OpenCV and Python URL: https://www.superdatascience.com/opencv-facerecognition/. (Last accessed: 10.01.2024).

15. Rosebrock A. Facial landmarks with dlib, OpenCV, and Python. 2017. URL: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/ (Last accessed: 15.01.2024).

16. Turk, M., Pentland, A.: ‘Eigenfaces for recognition', J. Cogn. Neurosci., 1991. pp. 71–86. (дата звернення 15.01.2024).

17. Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: ‘Face recognition by independent component analysis', IEEE Trans. Neural Netw., 2002, 13, (6), pp. 1450–1464.

18. Belhumeur P.N., Hespanha J.P, Kriegman D.J.: ‘Eigenfaces vs. _sherfaces: recognition using class speci_c linear projection', IEEE Trans. Pattern Anal. Mach. Intell., 1997. 19, (7), pp. 711–720.

19. Hu H., Zhang P., De la Torre F.: ‘Face recognition using enhanced linear discriminant analysis', IET Comput. Vis., 2010, 4, (3), pp. 195–208.

20. Yang J., Zhang D., Frangi A.F., Yang J.-Y.: ‘Two-dimensional PCA: a new approach to appearance-based face representation and recognition', IEEE Trans. Pattern Anal. Mach. Intell., 2004, 26, (1), pp. 131–137.

21. Naseem I., Togneri R., Bennamoun M.: ‘Linear regression for face recognition', IEEE Trans. Pattern Anal. Mach. Intell., 2010, 32, (11), pp. 2106–2112.

22. Lu J., Plataniotis K.N., Venetsanopoulos A.N.: ‘Face recognition using kernel direct discriminant analysis algorithms', IEEE Trans. Neural Netw., 2003, 14, (1), pp. 117–126.

23. He X., Yan S., Hu Y., Niyogi P., Zhang H.-J.: ‘Face recognition using Laplacian faces', IEEE Trans. Pattern Anal. Mach. Intell., 2005, 27, (3), pp. 328–340.

24. Perlibakas V.: ‘Distance measures for PCA-based face recognition', Pattern 42 Recognit. Lett., 2004, 25, (6), pp. 711–724

25. Kim K.I., Jung K., Kim H.J.: ‘Face recognition using kernel principal component analysis', IEEE Signal Process. Lett., 2002, 9, (2), pp. 40–42.

26. Bach F., Jordan M.: ‘Kernel independent component analysis', J. Mach. Learn. Res., 2003, 3, pp. 1–48

27. Ji S., Ye J.: ‘Generalized linear discriminant analysis: a uni_ed framework and ef_cient model selection', IEEE Trans. Neural Netw., 2008, 9, (10), pp. 1768–1782.

28. Roweis S., Saul L.: ‘Nonlinear dimensionality reduction by locally linear embedding', Science, 2000, 290, (5500), pp. 2323–2326.

29. Xiaofei H., Partha N.: ‘Locality preserving projections'. Int. Conf. on Advances in Neural Information Processing Systems (NIPS'03), 2003, pp. 153–161.

30. Ahonen T., Hadid A., Pietikäinen M.: ‘Face description with local binary patterns: application to face recognition', IEEE Trans. Pattern Anal. Mach. Intell., 2006, 28, (12), pp. 2037–2041.

32. Face Mask Detection Dataset. URL: https://www.kaggle.com/datasets/ andrewmvd/face-mask-detection (Last accessed: 17.01.2024).

33. LFW - People (Face Recognition) Dataset. URL: https://www.kaggle.com/datasets/atulanandjha/lfwpeople (Last accessed: 17.01.2024).

ДОДАТОК А

ДОВІДКА

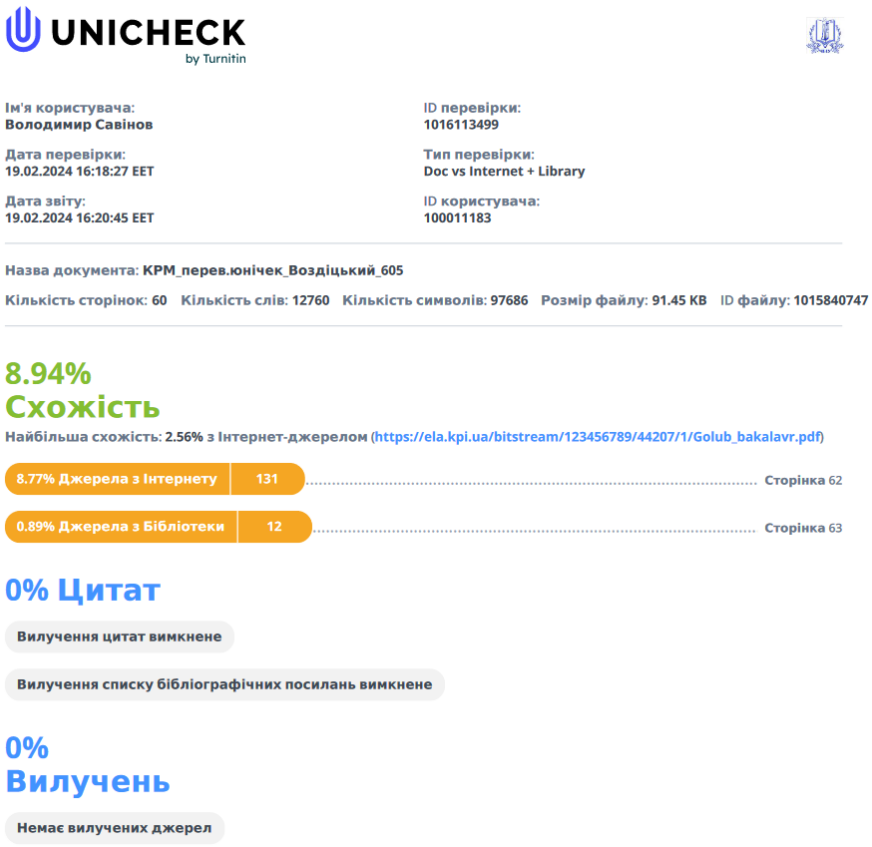
про перевірку на унікальність пояснювальної записки

кваліфікаційної магістерської роботи

на тему: «Програмно-апаратний комплекс детектування рухів та розпізнавання обличчя
на базі одноплатного комп'ютеру Raspberry Pi»
студента спеціальності 123 «Комп'ютерна інженерія», групи 605
Воздіцького Сергія Юрійовича
прізвище, ім'я, по-батькові

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck

Результат перевірки тексту кваліфікаційної роботи: схожість складає 8.94 %.


UNICHECK
by Turnitin

Ім'я користувача:
Володимир Савінов

ID перевірки:
1016113499

Дата перевірки:
19.02.2024 16:18:27 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.02.2024 16:20:45 EET

ID користувача:
100011183

Назва документа: КРМ_перев.юнічек_Воздіцький_605

Кількість сторінок: 60 Кількість слів: 12760 Кількість символів: 97686 Розмір файлу: 91.45 KB ID файлу: 1015840747

8.94%
Схожість

Найбільша схожість: 2.56% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/44207/1/Golub_bakalavr.pdf)

8.77% Джерела з Інтернету 131 Сторінка 62

0.89% Джерела з Бібліотеки 12 Сторінка 63

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Студент: Керівник: к. т. н., доцент

_____С. Ю. Воздіцький _____В. Ю. Савінов
підпис ініціали, прізвище підпис ініціали, прізвище

Дата: «___» _____ 2024 р.

ДОДАТОК Б

Код програми

```
from imutils.video import VideoStream
import argparse
import datetime
import imutils
import time
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

if args.get("video", None) is None:
      vs = VideoStream(src=0).start()
      time.sleep(2.0)

else:
      vs = cv2.VideoCapture(args["video"])

firstFrame = None
while True:

frame = vs.read()
frame = frame if args.get("video", None) is None else frame[1]
text

if frame is None:
      break
frame = imutils.resize(frame, width=500)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (21, 21), 0)

      if firstFrame is None:
            firstFrame = gray
            continue

frameDelta = cv2.absdiff(firstFrame, gray)
thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]


thresh = cv2.dilate(thresh, None, iterations=2)
      cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
      cnts = imutils.grab_contours(cnts)
 for c in cnts:
            if cv2.contourArea(c) < args["min_area"]:
                  continue
            (x, y, w, h) = cv2.boundingRect(c)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            text = "Occupied"
motion detection and tracking with Python and OpenCVPython
            cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
      cv2.putText(frame, datetime.datetime.now().strftime
            (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
1)
```

```
      cv2.imshow("Security Feed", frame)
      cv2.imshow("Thresh", thresh)
      cv2.imshow("Frame Delta", frameDelta)
      key = cv2.waitKey(1) & 0xFF

      if key == ord("q"):
            break
vs.stop() if args.get("video", None) is None else vs.release()
cv2.destroyAllWindows()

      cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
      cv2.putText(frame, datetime.datetime.now().strftime ,
            (10,frame.shape[0]- 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
1)

      cv2.imshow("Security Feed", frame)
      cv2.imshow("Thresh", thresh)
      cv2.imshow("Frame Delta", frameDelta)
      key = cv2.waitKey(1) & 0xFF

      if key == ord("q"):
            break

vs.stop() if args.get("video", None) is None else vs.release()
cv2.destroyAllWindows()
```

Main.py

```
from flask import Flask , redirect, url_for, request, render_template,
make_response, jsonify
from flask import send_file, abort
import requests as req
import json
from PIL import Image
from datetime import *
import base64
from io import *
import jsonpickle
import mysql.connector as mariadb
config = {
'host': '127.0.0.1',
'port': 3306,
'user': 'root',
'password': 'root',
'database': 'face_recognition'
}
app = Flask(__name__)
@app.route('/persons', methods=['GET', 'POST', 'DELETE'])
def persons():
conn = mariadb.connect(**config)
cur = conn.cursor()
if (request.method == 'GET'):
cur.execute("SELECT name FROM person")
# serialize results into JSON
row_headers=[x[0] for x in cur.description]
rv = cur.fetchall()
json_data=[]
for result in rv:
json_data.append(dict(zip(row_headers,result)))
# return the results!
```

```
return json.dumps(json_data)
elif (request.method == 'POST'):
person = request.get_json()
cur.execute("INSERT INTO person(name) values('{}')".format(person.get('name')))
conn.commit()
return "Data has been stored"
elif (request.method == 'DELETE'):
person = request.get_json()
cur.execute("DELETE FROM person WHERE name = ('{}')".format(person.get('name')))
conn.commit()
return "Data has been deleted"
else :
return 400
def myconverter(o):
if isinstance(o, datetime):
return o.__str__()
@app.route('/story', methods=['GET', 'POST'])
def story():
conn = mariadb.connect(**config)
cur = conn.cursor()
if (request.method == 'GET'):
cur.execute("SELECT datetime, name FROM story, person WHERE story.person_id =
person.id
ORDER BY datetime DESC")
# serialize results into JSON
row_headers=[x[0] for x in cur.description]
rv = cur.fetchall()
json_data=[]
for result in rv:
json_data.append(dict(zip(row_headers,result)))
# return the results!
return json.dumps(json_data, default = myconverter)
elif (request.method == 'POST'):
story = request.get_json()
cur.execute("SELECT id FROM person WHERE person.name =
'{}'".format(story.get('name')))
person_id = cur.fetchall()[0][0]
cur.execute("INSERT INTO story(datetime, person_id) values('{}',
'{}')".format(story.get('datetime'),
person_id))
conn.commit()
return "Data has been stored"
@app.route('/story/name')
def storyByName():
name = request.args.get('name', type = str)
conn = mariadb.connect(**config)
cur = conn.cursor()
cur.execute("SELECT datetime, name FROM story, person WHERE person.name = '{}' AND
story.person_id = person.id ORDER BY datetime DESC".format(name))
row_headers=[x[0] for x in cur.description]
rv = cur.fetchall()
json_data=[]
for result in rv:
json_data.append(dict(zip(row_headers,result)))
# return the results!
return json.dumps(json_data, default = myconverter)
@app.errorhandler(404)68
def page_not_found(e):
return "Sorry, end-point source not available.", 404
@app.route('/')
def index():
return "Door lock"
```

```
if __name__ == '__main__':
app.run(host='0.0.0.0', debug=True)
```

facial_req.py

```
# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
from datetime import *
import requests as req
import face_recognition
import imutils
import pickle
import time
import json
import cv2
import gpiozero
led = gpiozero.LED(17)
def myconverter(o):
if isinstance(o, datetime):
return o.__str__()
#Initialize 'currentname' to trigger only when a new person is identified.
currentname = "unknown"
#Determine faces from encodings.pickle file model created from train_model.py
encodingsP = "encodings.pickle"
#use this xml file
cascade = "haarcascade_frontalface_default.xml"
tripleCheck = 0
# load the known faces and embeddings along with OpenCV's Haar
# cascade for face detection
print("[INFO] loading encodings + face detector...")
data = pickle.loads(open(encodingsP, "rb").read(), encoding='iso-8859-1')
detector = cv2.CascadeClassifier(cascade)70
# initialize the video stream and allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
#vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
# start the FPS counter
fps = FPS().start()
def classify(self,id,confidence=101):
# Check if confidence is less them 100 ==> "0" is perfect match
if (confidence < 75):
label = self.names[id]
print(confidance)
elif (confidence > 75 and confidence < 101):
label = str(self.names[id]) + " no match"
print(confidance)
else:
label = "unknown"
print(confidance)
return label
# loop over frames from the video file stream
while True:
# grab the frame from the threaded video stream and resize it
# to 500px (to speedup processing)
frame = vs.read()
frame = imutils.resize(frame, width=500)
# convert the input frame from (1) BGR to grayscale (for face
# detection) and (2) from BGR to RGB (for face recognition)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
# detect faces in the grayscale frame
rects = detector.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30),
flags=cv2.CASCADE_SCALE_IMAGE)
# OpenCV returns bounding box coordinates in (x, y, w, h) order
# but we need them in (top, right, bottom, left) order, so we
# need to do a bit of reordering
boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
# compute the facial embeddings for each face bounding box
encodings = face_recognition.face_encodings(rgb, boxes)
names = []
# loop over the facial embeddings
for encoding in encodings:
# attempt to match each face in the input image to our known
# encodings
matches = face_recognition.compare_faces(data["encodings"],
encoding, tolerance=0.4)
name = "Unknown" #if face is not recognized, then print Unknown
# check to see if we have found a match
if True in matches:
# find the indexes of all matched faces then initialize a
# dictionary to count the total number of times each face
# was matched
matchedIdxs = [i for (i, b) in enumerate(matches) if b]
counts = {}
# loop over the matched indexes and maintain a count for
# each recognized face face
for i in matchedIdxs:
name = data["names"][i]
counts[name] = counts.get(name, 0) + 1
# determine the recognized face with the largest number
# of votes (note: in the event of an unlikely tie Python
# will select first entry in the dictionary)
name = max(counts, key=counts.get)
#If someone in your dataset is identified, print their name on the screen
if currentname != name:
currentname = name
print("Identified person with name: " + name)
dictionary = {"datetime": datetime.now().strftime("%Y-%m-%d %H:%M:%S"), "name":
name}
req.post("http://127.0.0.1:5000/story", headers = {'Content-type':
'application/json'},
json=dictionary)
# update the list of names
names.append(name)
led.on()
time.sleep(3)
led.off()
# loop over the recognized faces
for ((top, right, bottom, left), name) in zip(boxes, names):
# draw the predicted face name on the image - color is in BGR
cv2.rectangle(frame, (left, top), (right, bottom),
(0, 255, 225), 2)
y = top - 15 if top - 15 > 15 else top + 15
cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
.8, (0, 255, 255), 2)
# display the image to our screen
cv2.imshow("Facial Recognition is Running", frame)
key = cv2.waitKey(1) & 0xFF
# quit when 'q' key is pressed
if key == ord("q"):
break
```

```
# update the FPS counter
fps.update()
# stop the timer and display FPS information
fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

ДОДАТОК В

Публікації за темою роботи

УДК 004.89

Воздіцький С. Ю.,
студент 6-го курсу,
Савінов В. Ю.,
канд. техн. наук, доцент,
ЧНУ ім. Петра Могили, м. Миколаїв, Україна

ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЕТЕКТУВАННЯ РУХІВ ТА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ НА БАЗІ ОДНОПЛАТНОГО КОМП'ЮТЕРУ RASPBERRY PI

У сучасному світі зростає популярність використання технологій комп'ютерного зору для вирішення різноманітних завдань, таких як відеоспостереження, безпека та розпізнавання облич. Однак існуючі рішення часто потребують великих обчислювальних ресурсів та великих витрат на обладнання. Одним з підходів до вирішення цих задач є використання одноплатних комп'ютерів. Одноплатні комп'ютери є компактними і доступними пристроями, які можна використовувати для створення мобільних систем.

У цій роботі розроблено програмно-апаратний комплекс детектування рухів та розпізнавання обличчя на базі одноплатного комп'ютера Raspberry Pi. У цьому контексті він пропонує доступне та потужне рішення для створення програмно-апаратного комплексу детектування рухів та розпізнавання облич.

Для детектування рухів та розпізнавання обличчя існує безліч підходів. Для детектування рухів можна використовувати такі методи, як:

1) Обробка відео по кадрах: цей метод полягає у порівнянні послідовних кадрів відео для виявлення змін. Він є простим у реалізації, але може бути неефективним для виявлення швидких рухів або рухів, що відбуваються на тлі складних текстур.
2) Обчислення оптичного потоку: цей метод використовує математичне поняття оптичного потоку для виявлення руху між сусідніми кадрами відео. Він є більш ефективним, ніж обробка відео по кадрах, але може бути чутливим до шуму та артефактів.
3) Використання машинного навчання: цей метод використовує моделі машинного навчання для навчання на наборі даних відео з рухами та без рухів. Він є найбільш точним методом, але може бути складним у реалізації та потребувати значної кількості даних для навчання.

Для розпізнавання обличчя існують такі методи:

1) Використання геометричних ознак: Цей метод використовує геометричні характеристики облич, такі як відстані між очима, носом і ротом. Він є простим у реалізації, але може бути сприйнятливим до змін освітлення та поворотів голови.
2) Використання текстурних ознак: Цей метод використовує текстурні характеристики облич, такі як розподіл пікселів і текстурні шаблони. Він є більш точним, ніж використання геометричних ознак, але може бути складним у реалізації.
3) Використання машинного навчання: Цей метод використовує моделі машинного навчання для навчання на наборі даних з фотографій і відео облич. Він є найбільш точним методом, але може бути складним у реалізації та потребувати значної кількості даних для навчання.

Посилаючись на [1] автори пропонують метод детектування рухів на основі згорткової нейронної мережі. Метод був протестований на наборі даних з відео з рухомими людьми і показав високу точність. Метод складається з двох основних етапів:

1) Обробка відео: Відео розбивається на послідовні кадри, які потім використовуються для навчання CNN.
2) Детектування рухів: CNN використовується для детектування рухів на кожному кадрі відео.

На рисунку 1 показана вся запропонована структура. Система складається з двох частин: виявлення руху та класифікація об'єктів і регресія. По-перше, виявлення руху здійснюється за допомогою модуля виявлення руху з модулем фільтра та математичний морфологічний модуль (операція відкриття). Потім вилучення пов'язаної області виконується для отримання переміщення регіонів. Відповідно до інформації в рухомих регіонах, оригінальне зображення вирізається та обрізається, регіони вводяться в нейронну мережу, щоб отримати межі та категорію кожного об'єкта. Нарешті, згідно з інформацією про положення об'єкта в області, що рухається, координати відображаються на вихідному зображенні. Ключові модулі в загальну архітектуру включають виявлення руху, регіон вилучення та виявлення об'єктів. На першому етапі відео розбивається на послідовні кадри. Кожен кадр потім перетворюється в вектор розміром 1024, який представляє характеристики кадру. На другому етапі CNN використовується для детектування рухів на кожному кадрі відео. CNN складається з трьох основних шарів:

1) Згортковий шар: Цей шар використовує згортки для вилучення характерних ознак з кадру.
2) Фільтрувальний шар: Цей шар використовує фільтри для відсікання шуму та артефактів.
3) Підсумковий шар: Цей шар об'єднує результати згорточного шару та фільтрувального шару для визначення наявності руху.

Автори статті протестували свій метод на наборі даних з відео з рухомими людьми. Набір даних містить 100 відео з 200 000 кадрами. Метод авторам показав точність 95%. Це означає, що метод правильно детектував рухи на 95% кадрів набору даних.

Як правило, кожен кадр відео розділений на передній план і фон для виявлення рухомих об'єктів відповідно до різниці в інтенсивності пікселів або розподілі кольорів. Дослідники пропонують різні методи, наприклад штучний дизайн для виділення переднього плану. Як найпоширеніший метод, різниця кадрів базується на різниці рівнів сірого між двома сусідніми кадрами відео. Без фону моделювання його легко реалізувати, але він вразливий до шуму або складних сцен з локальним рухом у фоновому режимі. Тим часом виходячи з [3], модель шуму Гаусса (GMM) є більш надійною, але вона потребує кілька кадрів для моделювання та оновлення фону, що страждає від високої обчислювальної складності.

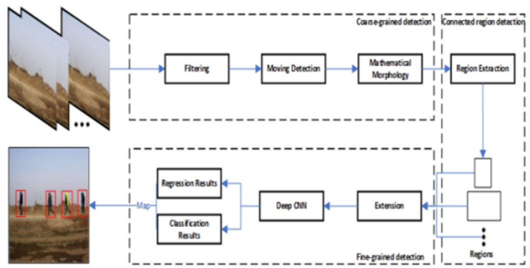


FIGURE 1: Framework schema

Рисунок 1 – структура згорткової мережі

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Haidi Zhu, Xin Yan, Hongying Tang, Yuchao Chang. Moving object detection with Deep CNNs. 2015. DOI: https://www.researchgate.net/publication/339174353_Moving_object_detection_with_Deep_CNNs
2. Andrew Shepley. 2019. Deep Learning For Face Recognition: A Critical Analysis. DOI: https://www.researchgate.net/publication/334783128_Deep_Learning_For_Face_Recognition_A_Critical_Analysis
3. Yilun Wang, Xiangyu Zhang, Xiaogang Wang, Jian Sun. 2016. Face recognition with deep learning. DOI: https://arxiv.org/
4. Caroline Dunn. 2022. DOI: https://www.tomshardware.com/how-to/raspberry-pi-facial-recognition