Міністерство освіти і науки України

Криворізький національний університет

Факультет інформаційних технологій

Кафедра комп’ютерних систем та мереж

Пояснювальна записка

до кваліфікаційної роботи бакалавра

за спеціальністю 123 «Комп’ютерна інженерія»

на тему: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Проектував \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З. Чефер

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ І. О. Музика

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. І. Кузнєцов

Завідувач кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. І. Купін

Кривий Ріг

2025

Криворізький національний університет

Факультет інформаційних технологій

Кафедра комп’ютерних систем та мереж

Ступінь вищої освіти бакалавр

Спеціальність 123 «Комп’ютерна інженерія»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри, голова циклової комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. І. Купін

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ року

**З А В Д А Н Н Я**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи \_\_\_\_\_Програмне забезпечення для розпізнавання обличчя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “\_\_”\_\_\_\_ 20\_\_року №\_\_

2. Строк подання студентом роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно

розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових

креслень)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання  видав | завдання  прийняв |
| Р1-Р3 | І. О. Музика, Доцент Кафедри КСМ |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання\_\_\_\_\_\_01/01/2025\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів роботи | Строк виконання етапів роботи | Примітка |
| 1 | Вибір теми, затвердження завдання, визначення мети і завдань |  |  |
| 2 | Аналіз літератури та подібних розробок, формування структури роботи |  |  |
| 3 | Опис і обґрунтування вибраних технологій та інструментів (розділ 1) |  |  |
| 4 | Розробка архітектури програмного забезпечення |  |  |
| 5 | Реалізація основного функціоналу програмного забезпечення |  |  |
| 6 | Тестування, відлагодження та опис результатів (розділ 3) |  |  |
| 7 | Написання розділу 2 (вибір і обґрунтування ПЗ) |  |  |
| 8 | Написання висновків, оформлення пояснювальної записки |  |  |
| 9 | Підготовка графічного матеріалу, оформлення документації |  |  |
| 10 | Захист роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Змн

Арк.

№\_Документа№ документа

Підпис

Дата

Аркуш

КНУ.РБ.123.25.13.Р

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнєцов

Затвердив

Купін

РЕФЕРАТ

Літера

Аркушів

КІ-21

Пояснювальна записка: 46 сторінок, 4 рисунків, 7 таблиць, 0 додаток, ? використаних джерел.

Об’єкт аналізу – система розпізнавання облич на основі бібліотеки OpenCV та мови програмування Python, яка дозволяє ідентифікувати користувача та здійснювати відповідні дії після успішної автентифікації.

Проект складається з трьох розділів.

Перший розділ присвячений аналізу технологій систем розпізнавання облич, ідентифікування користувача.

У другому розділі розкриті питання вибору програмної частини для розробки пз.

Третій розділ присвячений моделюванню, а також аналізу недоліків. Вкінці розділу подано рекомендації щодо покращення роботи.

OPENCV, PYTHON, LBPH, FR, PCA, LDA, CNN, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Explanatory note: 46 pages, 4 figures, 7 table, 0 appendix, ? sources used.

арк

№ документа

Підпис

Дата

арк

КНУ. РБ.123.25.13.EN

The object of analysis is a facial recognition system based on the OpenCV library and the Python programming language, which allows you to identify the user and take appropriate actions after successful authentication.

The project consists of three parts.

The first section is devoted to the analysis of technologies of facial recognition systems, user identification.

The second section reveals the issues of choosing the software part for the development of the application.

The third section is devoted to modeling, as well as analysis of shortcomings. At the end of the section, recommendations are given for improving the work

OPENCV, PYTHON, LBPH, FR, PCA, LDA, CNN, SOFTWARE

ЗМІСТ

Змн

Арк.

№\_Документа№ документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.З

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнєцов

Затвердив

Купін

ЗМІСТ

Літера

Аркушів

КІ-21

[ПЕРЕЛІК СКОРОЧЕНЬ 8](#_Toc197618353)

[**ВСТУП** 9](#_Toc197618354)

[**1.1 Опис проєкту** 10](#_Toc197618355)

[**1.2 Аналіз технологій розпізнавання облич** 12](#_Toc197618356)

[Основні методи розпізнавання обличь: 12](#_Toc197618357)

[Використані технології в проєкті: 13](#_Toc197618358)

[**Інші альтернативні технології** 14](#_Toc197618359)

[**1.3 Чому використовується Python** 14](#_Toc197618360)

[**Основні причини вибору Python:** 14](#_Toc197618361)

[**Альтернативи Python:** 15](#_Toc197618362)

[**1.4 Огляд застосованих бібліотек та ліцензій** 16](#_Toc197618363)

[**OpenCV (Open Source Computer Vision Library)** 16](#_Toc197618364)

[**NumPy** 17](#_Toc197618365)

[**OpenCV face module (cv2.face)** 18](#_Toc197618366)

[**Інші технології** 18](#_Toc197618367)

[Загальна характеристика застосованих ліцензій 18](#_Toc197618368)

[**1.5 Технології зберігання даних у системах розпізнавання облич** 18](#_Toc197618369)

[Використання PostgreSQL для зберігання даних 19](#_Toc197618370)

[Чому саме PostgreSQL? 19](#_Toc197618371)

[**Архітектура можливої бази даних** 19](#_Toc197618372)

[**Альтернативи PostgreSQL** 20](#_Toc197618373)

[**Використання PostgreSQL в інтеграції із Python** 21](#_Toc197618374)

[**1.6 Технічна реалізація автентифікації за обличчям** 21](#_Toc197618375)

[**1.7 Переваги Face ID у захисті персональних застосунків** 22](#_Toc197618376)

[Основні переваги реалізованого підходу: 23](#_Toc197618377)

[**1.8 Захист конфіденційності та біометричних даних** 23](#_Toc197618378)

[Основні заходи безпеки в реалізованій системі: 24](#_Toc197618379)

[**Висновки до Розділу 1** 25](#_Toc197618380)

[**Розділ 2. Вибір програмної частини для розробки програмного забезпечення** 27](#_Toc197618381)

[**2.1 Критерії вибору програмної частини** 27](#_Toc197618382)

[**2.1.1 Мова програмування** 27](#_Toc197618383)

[**2.2 Порівняльний аналіз інструментів розробки** 42](#_Toc197618384)

[**Висновок до Розділу 2** 46](#_Toc197618385)

**Розділ 3. Тестування** ………………………………………………………………47

**3.1 Середовище та залежнос**ті…………………….………………………..47

**3.2 Опис алгоритму**…………………………………..………………...……49

**3.3 Навчання моделі та розпізнавання облич**……...………………..…...50

**3.4 Реальне час розпізнавання та автентифікація**………..………….….52

**Висновоки**………………………...…………………………………………………54

**Список джерела**……………………………...…………………………………….56

арк

№ документа

Підпис

Дата

арк

КНУ. РБ.123.25.13.ПС

ПЕРЕЛІК СКОРОЧЕНЬ

арк

№ документа

Підпис

Дата

арк

КНУ. РБ.123.25.13.ПС

LBPH – Local Binary Patterns Histograms

FR – Face Recognizer

PCA – Principal Component Analysis

LDA – Linear Discriminant Analysis

CNN – Convolutional neural networks

**ВСТУП**

Змн

Арк.

№\_Документа№ документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.В

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнєцов

Затвердив

Купін

ВСТУП

Літера

Аркушів

КІ-21

У сучасному цифровому світі питання безпеки інформації, а також ідентифікації особи набувають усе більшої актуальності. Зростання обсягів чутливих даних, що зберігаються на персональних пристроях і в мережі, потребує впровадження надійних, ефективних та зручних методів автентифікації користувачів. Одним із таких рішень є системи розпізнавання облич, які поєднують у собі точність, швидкість роботи та простоту використання.

Біометричні технології, зокрема розпізнавання облич, дедалі активніше впроваджуються як у промисловості, так і в побутових застосунках, витісняючи традиційні методи захисту (логін/пароль). Використання унікальних фізіологічних характеристик людини дозволяє підвищити рівень безпеки та знизити ризики несанкціонованого доступу.

У рамках цієї кваліфікаційної роботи реалізовано проєкт програмного забезпечення для розпізнавання облич, що базується на бібліотеці OpenCV та мові програмування Python. Система дозволяє здійснювати ідентифікацію користувача за допомогою вебкамери та запускати захищені застосунки лише після успішного проходження автентифікації. Архітектура рішення забезпечує його простоту, автономність, адаптивність до потреб користувача та можливість подальшого масштабування з використанням сучасних підходів глибокого навчання.

### МЕТА РОБОТИ

Метою цієї кваліфікаційної роботи є **розробка, реалізація та тестування програмного забезпечення для розпізнавання облич**, яке:

* забезпечує автентифікацію користувача за допомогою відеопотоку з вебкамери;
* демонструє ефективне використання алгоритму Local Binary Patterns Histograms (LBPH);
* має можливість розширення функціоналу (наприклад, інтеграцію з іншими програмами або базами даних);

Змн

Арк.

№\_Документа№ документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.В

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнєцов

Затвердив

Купін

МЕТА РОБОТИ

Літера

Аркушів

КІ-21

* відповідає сучасним вимогам до захисту персональних біометричних даних;
* працює автономно без потреби підключення до інтернету чи зовнішніх серверів.

**РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЙ ТА ОПИС ПРОЄКТУ .**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.З

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

РОЗДІЛ 1

Літера

Аркушів

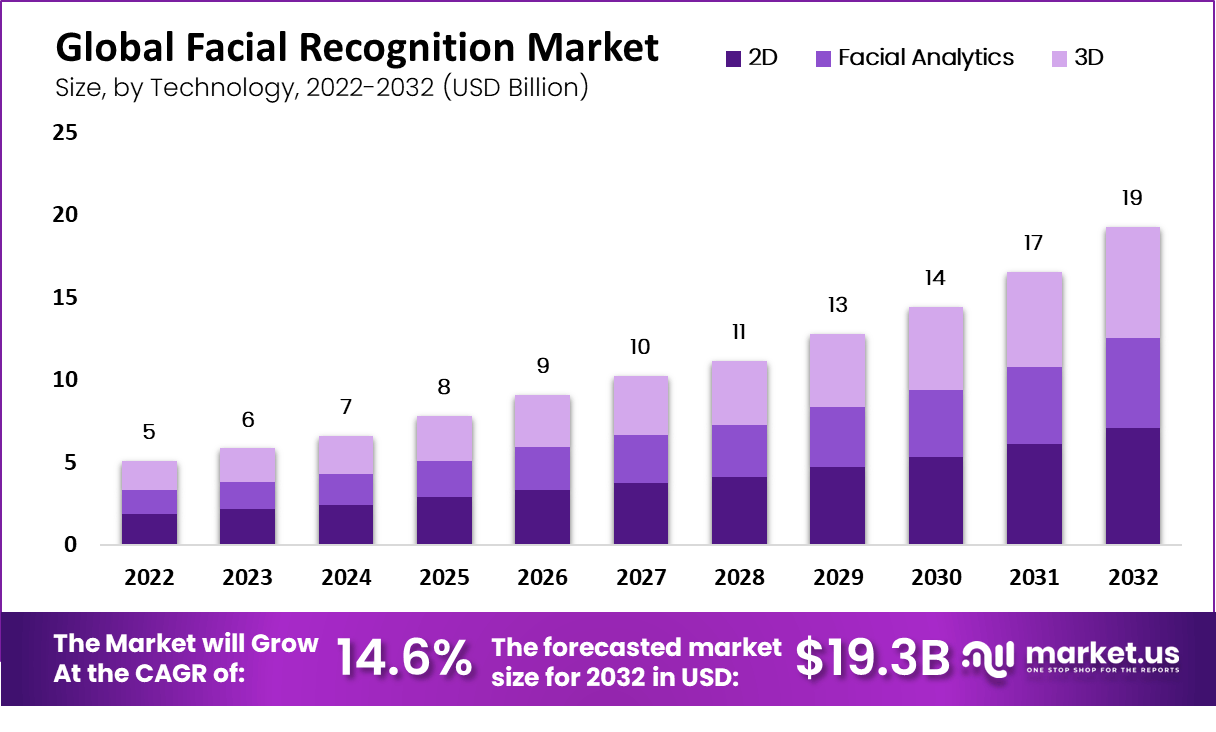
КІ-21

# **1.1 Опис проєкту**

У сучасному світі питання безпеки та ідентифікації особи стають усе більш актуальними. Серед широкого спектра технологій автентифікації розпізнавання облич є одним із найперспективніших напрямів, оскільки поєднує зручність використання, швидкість обробки та високий рівень точності. Метою даного проєкту є розробка системи розпізнавання облич на основі бібліотеки OpenCV та мови програмування Python, яка дозволяє ідентифікувати користувача та здійснювати відповідні дії після успішної автентифікації.

Проєкт реалізує повний цикл обробки облич: захоплення зображення користувача через вебкамеру, попередню обробку зображення (переведення у відтінки сірого для покращення якості розпізнавання), виявлення обличчя на відеопотоці за допомогою класифікатора Haar Cascade та подальше збереження виявлених фрагментів обличчя у локальній базі даних для тренування моделі. Після збору достатньої кількості даних система навчає розпізнавач облич (Face Recognizer) на основі алгоритму Local Binary Patterns Histograms (LBPH).  
Після тренування моделі користувач може проходити автентифікацію шляхом реального розпізнавання свого обличчя в режимі реального часу. У разі успішного розпізнавання запускається певний застосунок або виконується інша визначена дія, наприклад, відкриття веб браузера Google Chrome.

Система має на меті бути максимально легкою у використанні, що дозволяє інтегрувати її у різноманітні реальні сценарії: автоматизовані системи входу до комп'ютера, запуск захищених програм або системи контролю доступу до приміщень. Вона також є чудовим прикладом практичного застосування технологій машинного навчання та комп’ютерного зору у реальних умовах без потреби використання складних або дорогих рішень на основі глибокого навчання.



Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

**Рисунок 1 .1**  Частка ринку за типами технологій

Архітектура проєкту побудована таким чином, щоб забезпечити розширюваність та адаптивність. Основні модулі включають:

1. **Модуль захоплення обличчя:** відповідальний за виявлення та збереження облич користувача.
2. **Модуль тренування моделі:** формує набір даних та навчає модель LBPH розпізнаванню користувача.
3. **Модуль розпізнавання облич:** виконує ідентифікацію користувача у реальному часі.
4. **Модуль керування діями:** здійснює запуск додатків після підтвердження особи.

Технічна реалізація проєкту базується на використанні доступних і перевірених інструментів:

1. **Python** як основна мова програмування завдяки простоті розробки, великій кількості бібліотек для обробки зображень та гнучкості інтеграції;
2. **OpenCV** як головна бібліотека для роботи з відеопотоками, обробки зображень та реалізації алгоритмів виявлення і розпізнавання облич;
3. **NumPy** для ефективної роботи з масивами даних;
4. **Модулі ОС і Subprocess** для роботи з файловою системою та керування системними процесами.

Важливим аспектом є вибір алгоритму Local Binary Patterns Histograms для розпізнавання облич. Його перевагами є низька вимогливість до обчислювальних ресурсів, стійкість до змін освітлення, висока точність при невеликих об'ємах даних та простота реалізації. Це робить LBPH оптимальним рішенням для створення легкої, але ефективної системи розпізнавання облич без залучення складних нейронних мереж, які вимагають великої кількості даних та значних обчислювальних потужностей.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

Проєкт також приділяє увагу безпеці збереження даних: всі захоплені зображення та навчена модель зберігаються локально на комп'ютері, без відправлення інформації на сторонні сервери або у хмарні сервіси, що мінімізує ризик витоку персональних даних.

Таким чином, запропоноване рішення поєднує в собі простоту використання, високу швидкість роботи та достатню точність для більшості прикладних завдань, що ставить його у вигідне положення серед інших легких систем розпізнавання облич. Крім того, гнучкість архітектури дозволяє у майбутньому легко масштабувати проєкт або інтегрувати більш складні алгоритми на базі глибокого навчання, такі як нейронні мережі, без необхідності кардинальної перебудови основного коду.

# **1.2 Аналіз технологій розпізнавання облич**

Розпізнавання облич є однією з найактивніше досліджуваних та впроваджуваних областей комп'ютерного зору. Залежно від конкретних вимог до точності, швидкості, ресурсомісткості та сценарію використання, існують різні технології, алгоритми та підходи до розв'язання задачі розпізнавання облич.

### Основні методи розпізнавання обличь:

**1. Алгоритми на основі ознак (feature-based methods):**

Цей підхід полягає у виявленні ключових особливостей обличчя, таких як контури очей, носа, рота, а також геометричні співвідношення між ними. Одним із найвідоміших методів є використання вейвлет-перетворення, методів контурної обробки та структурних дескрипторів.

Переваги: висока швидкість обробки, стійкість до часткових змін обличчя (окуляри, борода).

Недоліки: чутливість до ракурсу та якості зображення.

**2. Алгоритми на основі шаблонів (appearance-based methods):**

У цьому підході використовується повне зображення обличчя як вхідні дані, без виділення окремих ознак. Поширеними є такі методи як Principal Component Analysis (PCA, "Eigenfaces"), Linear Discriminant Analysis (LDA) та Local Binary Patterns Histograms (LBPH).

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

Переваги: простота реалізації, ефективність при контрольованих умовах освітлення.  
Недоліки: зниження точності за умов зміни освітлення або пози обличчя.

**3. Алгоритми на основі глибокого навчання:**

Найсучасніший підхід, який передбачає використання глибоких нейронних мереж, зокрема згорткових нейронних мереж (CNN). Прикладами є такі моделі, як FaceNet, DeepFace (Meta), VGGFace.

Переваги: дуже висока точність, стійкість до змін ракурсу, освітлення, виразів обличчя.

Недоліки: великі вимоги до обчислювальних ресурсів і даних для навчання.

### Використані технології в проєкті:

У рамках даного проєкту обрано метод на основі **Local Binary Patterns Histograms (LBPH)** через його високу ефективність у задачах розпізнавання облич у невеликих наборах даних та за обмежених ресурсів.

**OpenCV (Open Source Computer Vision Library):**

OpenCV є провідною бібліотекою для комп’ютерного зору з відкритим кодом, яка пропонує більше ніж 2500 оптимізованих алгоритмів для задач обробки зображень та відео.

У проєкті використовуються наступні її компоненти:

1. **Haar Cascades** — для виявлення облич на зображенні.
2. **LBPH Face Recognizer** — для розпізнавання особи за допомогою векторів локальних бінарних шаблонів.

**Переваги OpenCV:**

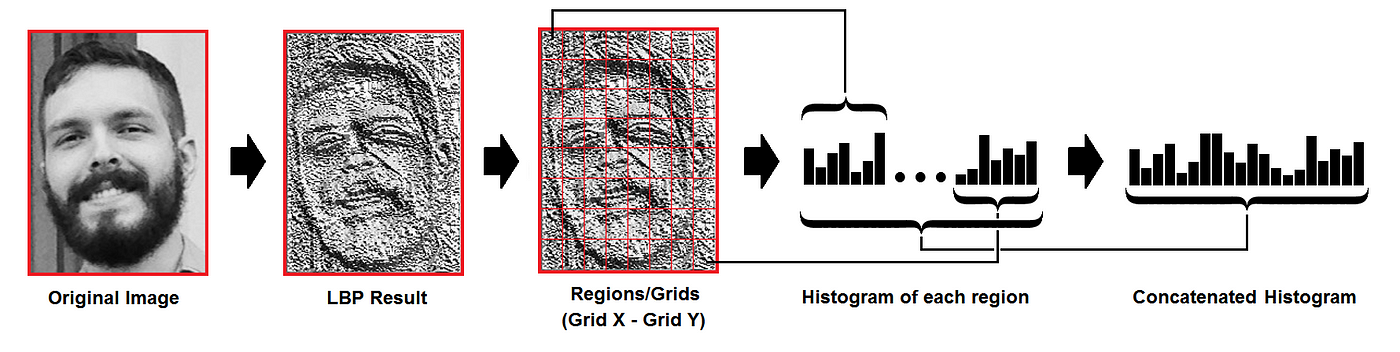
1. Відкритий вихідний код та безкоштовна ліцензія BSD;
2. Кросплатформність (Windows, Linux, macOS, Android, iOS);
3. Велика кількість готових інструментів для обробки зображень та розпізнавання об'єктів;
4. Активна спільнота розробників.

**NumPy:**

Бібліотека для ефективної роботи з великими масивами даних, необхідна для обробки наборів зображень та числових обчислень при тренуванні моделі.

**Алгоритм LBPH (Local Binary Patterns Histograms):**

Алгоритм працює шляхом аналізу локальних текстурних особливостей зображення та формування гістограми локальних шаблонів, що дозволяє системі зберігати важливі характеристики обличчя навіть при варіаціях освітлення чи міміки.



Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

**Рисунок 1 .2**  Розпізнавання облич з використанням гістограм локальних бінарних шаблонів (Local Binary Patterns Histograms, LBPH)

### **Інші альтернативні технології**

**1. DeepFace (Meta AI):**

Глибока нейронна мережа, яка показує точність розпізнавання понад 97% на тестових наборах Labeled Faces in the Wild (LFW).

**2. FaceNet (Google):**

Модель глибокого навчання, що перетворює обличчя у вектори ознак (ембеддінги) для подальшої класифікації або пошуку.

**3. Dlib:**

Популярна бібліотека з відкритим кодом, що надає прості у використанні інструменти для детекції та розпізнавання облич на основі машинного навчання та глибокого навчання.

**4. Mediapipe (Google):**

Набір кросплатформених рішень для комп'ютерного зору та розпізнавання облич у реальному часі, оптимізованих для мобільних пристроїв.

Таким чином, обрані технології у проєкті є оптимальним балансом між простотою реалізації, доступністю, якістю розпізнавання та швидкістю роботи, що робить їх ідеальними для задач середнього рівня складності.

# **1.3 Чому використовується Python**

Python став основною мовою програмування для реалізації проєкту завдяки своїм численним перевагам у сфері комп’ютерного зору, машинного навчання та швидкої прототипізації.

# **Основні причини вибору Python:**

**1. Простота та швидкість розробки:**

Python має дуже лаконічний та читабельний синтаксис, що дозволяє суттєво скоротити час розробки навіть складних програм. Завдяки цьому мову часто обирають для створення прототипів та дослідницьких проєктів.

**2. Великий набір спеціалізованих бібліотек:**

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

Python має розвинену екосистему бібліотек для обробки зображень, комп'ютерного зору та машинного навчання, серед яких особливо важливими є:

1. **OpenCV** (обробка зображень та відео),
2. **NumPy** (наукові обчислення),
3. **scikit-learn** (машинне навчання),
4. **TensorFlow, PyTorch** (глибоке навчання).

**3. Кросплатформність:**

Python працює на всіх основних операційних системах без необхідності значних змін у коді, що спрощує розгортання проєкту.

**4. Активна спільнота та документація:**

Python має одну з найбільших спільнот розробників у світі, що забезпечує наявність величезної кількості готових прикладів, підказок, форумів підтримки та документації.

**5. Відкритий код та безкоштовна ліцензія:**

Python є відкритим та безкоштовним для використання, що робить його доступним вибором як для особистих проєктів, так і для комерційних застосувань.

### **Альтернативи Python:**

**1. C++:**

OpenCV первинно був написаний на C++, що дозволяє досягти максимальної швидкості роботи програм. Проте розробка на C++ є значно складнішою і потребує більше часу на реалізацію та налагодження.

**2. Java:**

Мова підтримує OpenCV через Java API, але поступається Python у простоті та швидкості розробки для проєктів комп'ютерного зору.

**3. JavaScript (з використанням TensorFlow.js або face-api.js):**

Рішення на основі JavaScript підходять для веб-проєктів і мобільних застосунків, але мають обмеження у швидкості та можливостях порівняно із нативними десктопними програмами.

**4. MATLAB:**

Використовується у наукових дослідженнях для обробки зображень, але має комерційну ліцензію та менш гнучкий у порівнянні з Python підхід для масштабування проєктів.

**Таблиця 1. 1 :** Порівняння мов програмування для проєктів на базі ШІ

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

| Language | Advantages | Disadvantages | Typical Usage |
| --- | --- | --- | --- |
| Python | Easy syntax, many libraries, AI-ready | Slower than compiled languages | Machine Learning, AI, Web |
| C++ | Very fast, powerful | Harder to develop and debug | Real-time systems, embedded |
| Java | Good performance, platform independent | More verbose code | Web applications, big backend systems |
| JavaScript | Easy for web integration | Less suited for heavy AI calculations | Frontend AI (TensorFlow.js) |
| R | Statistical analysis, data science | Not general-purpose | Data Science, Statistical Modeling |

Таким чином, **Python** був обраний у проєкті завдяки своїй простоті, швидкості розробки, доступності бібліотек та активній підтримці з боку спільноти, що забезпечує швидке створення стабільної та ефективної системи розпізнавання облич.

# **1.4 Огляд застосованих бібліотек та ліцензій**

У рамках реалізації проєкту з розпізнавання облич було використано декілька важливих бібліотек та технологічних рішень, кожне з яких має власні особливості використання та ліцензійні обмеження.

### **OpenCV (Open Source Computer Vision Library)**

**Опис:**

OpenCV — це бібліотека комп'ютерного зору з відкритим вихідним кодом, яка забезпечує багатий набір інструментів для обробки зображень, відеоаналізу, розпізнавання об'єктів та машинного навчання. Вона активно використовується у багатьох промислових, академічних та дослідницьких проєктах.

**Ключові можливості:**

1. Виявлення облич за допомогою каскадних класифікаторів Хаара (Haar Cascade Classifier);
2. Розпізнавання облич за допомогою алгоритму LBPH (Local Binary Patterns Histograms);
3. Підтримка обробки зображень у реальному часі;
4. Підтримка різних мов програмування (Python, C++, Java).

**Ліцензія:**  
OpenCV поширюється під **BSD 3-Clause License**.

Це надзвичайно гнучка ліцензія, яка дозволяє:

1. Використовувати бібліотеку як у комерційних, так і в некомерційних проєктах;
2. Модифікувати та розповсюджувати вихідний код;

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

1. Єдина умова — збереження авторських прав та вказівка оригінального джерела в документації.

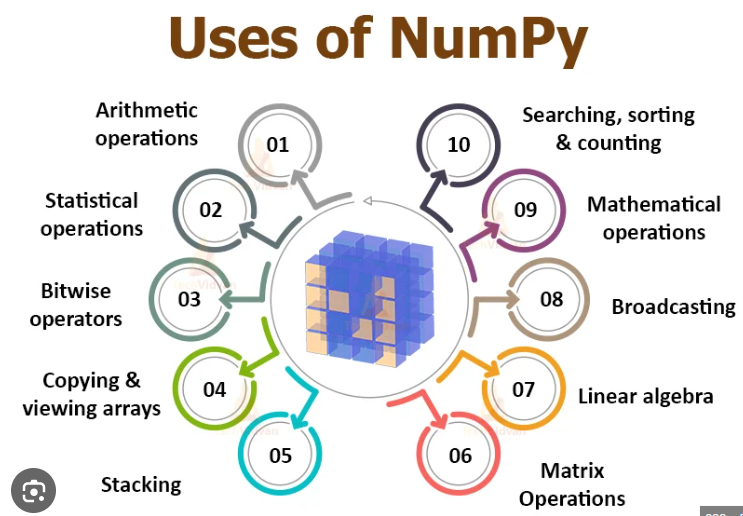
**Переваги такої ліцензії:**

1. Відсутність обмежень на комерційне використання;
2. Прозорість та відкритість для модифікацій;
3. Можливість інтеграції у власні закриті системи.

### **NumPy**

**Опис:**

NumPy — це основна бібліотека для наукових обчислень у Python. Вона забезпечує підтримку багатовимірних масивів та матриць, а також великий набір математичних функцій для роботи з ними.



**Рисунок 1 .3 Використання NumPy**

**Ключові можливості:**

1. Оптимізовані обчислення для масивів великих розмірів;
2. Інтеграція з іншими бібліотеками машинного навчання;
3. Прискорення математичних обчислень для обробки зображень.

**Ліцензія:**  
NumPy має **BSD 3-Clause License**, аналогічну OpenCV, що дозволяє вільно використовувати бібліотеку для будь-яких цілей.

### **OpenCV face module (cv2.face)**

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ПР

**Опис:**  
Підмодуль cv2.face входить до складу розширення OpenCV-contrib — окремого набору додаткових функцій OpenCV. Саме цей модуль надає API для роботи з алгоритмом розпізнавання облич на основі LBPH, а також з іншими популярними методами (Fisherfaces, Eigenfaces).

**Ліцензія:**   
Також поширюється за умовами **BSD 3-Clause License**.

### **Інші технології**

**1.Haar Cascade Classifier:**

Метод виявлення облич, який був запропонований Полом Віолою та Майклом Джонсом у 2001 році. Реалізація в OpenCV також підпорядковується BSD-ліцензії.

**2. Інтерфейс командної строки Windows (subprocess):**

Вбудований у Python модуль subprocess використовується для запуску зовнішніх додатків (у даному проєкті — для відкриття браузера Google Chrome). Цей модуль є частиною стандартної бібліотеки Python і не потребує окремої ліцензії.

### Загальна характеристика застосованих ліцензій

Всі використані бібліотеки у проєкті мають відкриті, вільні та гнучкі ліцензії (BSD 3-Clause License), що дозволяє:

1. Безкоштовно використовувати, модифікувати та інтегрувати їх у власні продукти;
2. Створювати як відкриті, так і комерційні застосунки без необхідності оприлюднювати вихідний код власної розробки;
3. Забезпечити юридичну безпеку проєкту при подальшій його розробці або комерціалізації.

Таким чином, вибір бібліотек з відкритим вихідним кодом і вільною ліцензією зробив проєкт доступним для розвитку, масштабування та інтеграції в реальні застосунки без ризиків порушення авторських прав.

# **1.5 Технології зберігання даних у системах розпізнавання облич**

Важливою складовою проєктів із розпізнавання облич є ефективне та надійне зберігання даних, таких як зображення облич, метаінформація про користувачів, результати навчання моделей та інші допоміжні дані. У даному проєкті в початковій версії зберігання здійснюється за допомогою локальної файлової системи у вигляді окремих файлів формату .jpg для зображень та .yml для моделей розпізнавання. Такий підхід має обмежену масштабованість та безпеку, що робить актуальним питання переходу до використання сучасних систем керування базами даних (СКБД).

## Використання PostgreSQL для зберігання даних

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

Однією з найкращих опцій для розширення проєкту є впровадження бази даних **PostgreSQL**.

### Чому саме PostgreSQL?

1. **Надійність:** PostgreSQL підтримує транзакції, контроль цілісності даних, механізми реплікації та резервного копіювання, що дозволяє уникати втрат інформації.
2. **Підтримка великих обсягів даних:** Завдяки оптимізованим механізмам роботи з великими обсягами інформації PostgreSQL підходить для проєктів із тисячами або навіть мільйонами записів.
3. **Гнучкість:** Можливість роботи як зі структурованими даними (таблиці), так і з напівструктурованими (JSON, XML).
4. **Розширення:** Існують модулі для зберігання зображень у вигляді BLOB (Binary Large Objects), що дозволяє безпосередньо зберігати обличчя в базі даних.
5. **Безпека:** Шифрування на рівні з'єднання (SSL/TLS), контроль доступу та автентифікація користувачів.
6. **Ліцензія:** PostgreSQL License є однією з найбільш вільних, що надає право використовувати, змінювати та розповсюджувати програмне забезпечення без значних обмежень.

Таким чином, PostgreSQL може бути використаний для:

1. Зберігання інформації про користувачів (імена, ID);
2. Збереження векторів ознак облич після обробки (embedding vectors);
3. Логування результатів розпізнавання та статистики роботи системи.

## **Архітектура можливої бази даних**

При впровадженні PostgreSQL можна побудувати таку схему бази даних:

1. **Таблиця Users**
   * user\_id (PRIMARY KEY)
   * full\_name
   * email (опційно)
   * registration\_date
2. **Таблиця Faces**
   * face\_id (PRIMARY KEY)
   * user\_id (FOREIGN KEY до Users)
   * face\_image (BLOB — зображення обличчя)
   * features\_vector (JSONB — вектор ознак)
   * capture\_date
3. **Таблиця RecognitionLogs**
   * log\_id (PRIMARY KEY)

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ.ПК.123.25.13..ПР

* + user\_id (FOREIGN KEY)
  + recognition\_time
  + confidence\_level
  + action\_taken (опис подальших дій)

Така структура дозволяє ефективно масштабувати проєкт та відстежувати історію взаємодії з користувачами.

## **Альтернативи PostgreSQL**

Хоча PostgreSQL є дуже потужним варіантом, існують також альтернативні рішення для зберігання даних у подібних системах:

* **MySQL**  
  Популярна реляційна СКБД, яка має вищу швидкість роботи у простих запитах, але поступається PostgreSQL за гнучкістю та функціональністю для складних завдань. Підтримує BLOB-об'єкти для збереження зображень.
* **SQLite**  
  Легка вбудована база даних, яка ідеально підходить для невеликих десктопних застосунків або прототипів. Вона зберігається у вигляді одного файлу на диску та не потребує окремого серверу.
* **MongoDB**  
  NoSQL база даних, що дозволяє зберігати документи у форматі JSON. Підходить для зберігання неструктурованих даних та великих об'ємів векторів ознак облич. Зберігання зображень реалізується через механізм GridFS.
* **Firebase Firestore**  
  Хмарне рішення від Google для мобільних застосунків та web-платформ. Забезпечує просту інтеграцію та автоматичне масштабування, однак для систем реального часу на локальних серверах може бути менш гнучким.
* **Redis**  
  База даних у пам'яті, яка підходить для кешування результатів розпізнавання та зберігання тимчасових даних, що потребують миттєвого доступу.

**Таблиця 1.2**  Варіанти зберігання даних для систем розпізнавання обличь

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

| Database | Type | Advantages | Disadvantages | Use Cases |
| --- | --- | --- | --- | --- |
| PostgreSQL | Relational SQL | Reliable, powerful, supports AI features (like JSONB) | Needs setup and maintenance | Structured data, logging AI results |
| MongoDB | NoSQL | Flexible, handles large images and JSON easily | Less rigid consistency | Storing raw face data, metadata |
| SQLite | Embedded SQL | Very lightweight, no server needed | Limited scalability | Small apps, mobile apps |
| Firebase | Cloud NoSQL | Easy cloud integration | Costs grow with data size | Mobile/web apps, real-time sync |

## **Використання PostgreSQL в інтеграції із Python**

Python має чудову підтримку роботи з PostgreSQL через такі популярні бібліотеки:

1. **psycopg2** — базова бібліотека для інтеграції з PostgreSQL;
2. **SQLAlchemy** — ORM-рішення для об'єктно-орієнтованої роботи з базами даних;
3. **Django ORM** — інтегрована система роботи з БД у випадку використання веб-фреймворку Django.

Це дозволяє дуже легко реалізувати запис нових даних, оновлення, пошук за збереженими ознаками облич або логифікацію всіх операцій розпізнавання.

* Перехід до використання потужної системи керування базами даних, такої як PostgreSQL, дозволить проєкту вийти за рамки експериментального рішення та стати повноцінною комерційною або корпоративною системою. Використання БД забезпечить масштабованість, підвищить безпеку та спростить подальший розвиток функціоналу, наприклад: впровадження аналітики результатів розпізнавання, інтеграцію з веб-інтерфейсами або мобільними застосунками.

## **1.6 Технічна реалізація автентифікації за обличчям**

У межах проєкту реалізовано повний цикл біометричної автентифікації особи на основі розпізнавання обличчя, використовуючи мову програмування Python, бібліотеку OpenCV та алгоритм Local Binary Patterns Histograms (LBPH). В основі системи — дві фази: захоплення навчальних зображень обличчя та подальше розпізнавання у режимі реального часу.

Система побудована з використанням таких модулів:

1. **Модуль capture\_faces()** — активує вебкамеру, фіксує обличчя користувача на відеопотоці, вирізає його фрагмент, зберігає серію з 20 зображень у локальну директорію (faces/) для подальшого навчання моделі.

Арк.

№ документа

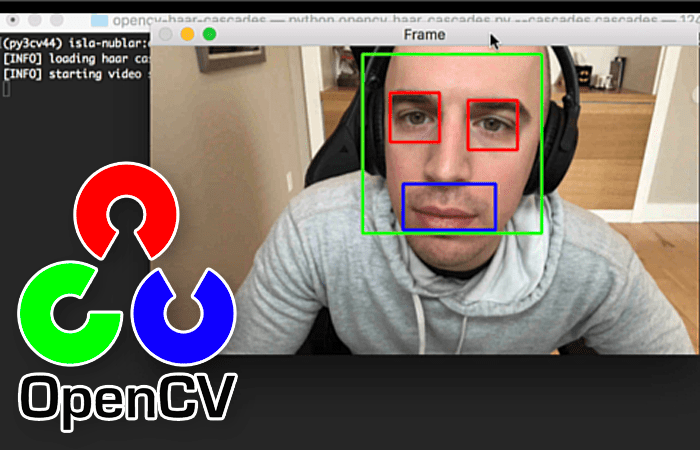
Підпис

Дата

Арк.ю

КНУ. РБ 123.25.13..ПР

1. **Модуль recognize\_face()** — виконує розпізнавання облич на відео, порівнює його з навченою LBPH-моделлю (face\_recognizer.yml) і, у разі успішного збігу, виконує дію: запуск Google Chrome.
2. **Модуль open\_application()** — за допомогою модуля subprocess відкриває захищену програму (у даному випадку Google Chrome).



**Рисунок 1 .4** Виявлення обличчя у відеопотоці за допомогою класифікатора Haar

## **1.7 Переваги Face ID у захисті персональних застосунків**

Інтеграція біометричної автентифікації на основі розпізнавання облич у процес доступу до персональних застосунків є сучасним і ефективним способом забезпечення захисту чутливої інформації на рівні кінцевого користувача. В умовах постійного зростання загроз у сфері кібербезпеки, зокрема фішингових атак, крадіжки паролів або підбору облікових даних, традиційні механізми автентифікації стають дедалі менш надійними. Саме тому використання Face ID як додаткового або основного методу автентифікації набуває все більшого поширення.

У межах даного проєкту розроблена система, яка після успішного розпізнавання обличчя користувача автоматично відкриває веббраузер Google Chrome. Цей вибір не є випадковим, адже Chrome — один із найбільш використовуваних застосунків, який зберігає в собі значну кількість конфіденційних даних: історію перегляду, облікові записи Google, доступ до банківських сервісів, паролі, розширення з критичною функціональністю, токени авторизації тощо. Таким чином, захист доступу до браузера є важливою складовою загальної цифрової гігієни користувача.

### Основні переваги реалізованого підходу:

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

1. **Високий рівень безпеки**. На відміну від звичайних паролів, які можуть бути вкрадені, зламані або підібрані за допомогою словникових атак, розпізнавання облич базується на унікальних біометричних параметрах, що практично неможливо відтворити або скопіювати. Це значно підвищує стійкість системи до несанкціонованого доступу.
2. **Повна автоматизація процесу входу**. Користувачеві не потрібно щоразу вводити пароль або проходити багатоетапну перевірку. Достатньо лише з’явитися перед камерою — і програма самостійно здійснить автентифікацію, оцінивши схожість з навченою моделлю.
3. **Гнучка контрольованість та налаштування порогу довіри**. Завдяки використанню алгоритму LBPH та методів обчислення схожості, система дозволяє визначити порогове значення (наприклад, confidence < 50), при якому доступ вважається дозволеним. Це надає можливість адаптувати систему до різних умов використання — як у домашньому середовищі, так і в офісі або лабораторії.
4. **Масштабованість та адаптивність**. Запропоновану систему легко модифікувати під будь-який інший застосунок, окрім браузера. Замість відкриття Chrome можна реалізувати запуск програмного середовища (наприклад, Visual Studio Code, PyCharm), клієнтів електронної пошти (Microsoft Outlook, Thunderbird), файлових менеджерів із конфіденційними даними, або навіть автоматичний доступ до певних директорій, зашифрованих архівів чи мережевих дисків.
5. **Комфорт та зниження когнітивного навантаження**. Сучасні користувачі змушені запам’ятовувати десятки паролів для різних систем і застосунків. Біометричний підхід суттєво спрощує щоденну взаємодію з комп’ютером, зменшуючи кількість дій для отримання доступу до важливої інформації.
6. **Інтеграція з іншими методами автентифікації**. Реалізована система може працювати як самостійно, так і в комбінації з іншими механізмами безпеки, наприклад, двофакторною автентифікацією (2FA), де обличчя виступає другим рівнем підтвердження особи після пароля або PIN-коду.

## **1.8 Захист конфіденційності та біометричних даних**

Використання біометричних ознак, зокрема обличчя, як способу автентифікації, неминуче порушує питання захисту персональних даних. Біометрія — це чутлива інформація, яка має постійний характер (на відміну від паролів її не можна змінити), тому її витік може мати серйозні довгострокові наслідки для користувача. Саме тому у реалізації подібних систем необхідно приділяти особливу увагу безпеці обробки, зберігання та використання біометричних шаблонів.

У межах розробленого проекту вжито низку заходів, які дозволяють забезпечити високий рівень конфіденційності без залучення складних хмарних архітектур або сторонніх API. Основний принцип, закладений у систему, — **локальність і контрольованість**. Всі етапи обробки обличчя — від захоплення зображення до навчання моделі LBPH — виконуються локально на комп'ютері користувача.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13..ПР

### Основні заходи безпеки в реалізованій системі:

1. **Локальне зберігання даних**. Усі зображення, що збираються для навчання моделі, зберігаються у вигляді .jpg файлів у директорії faces/, яка створюється локально. Тренована модель LBPH зберігається у форматі .yml, що дозволяє при потребі переносити її на інші системи без повторного навчання. Завдяки локальному підходу забезпечується повна незалежність від сторонніх серверів, а також максимальний контроль користувача над зібраними даними.
2. **Мінімізація обсягу та змісту даних**. Система не зберігає повноформатні відеозаписи або повні метадані. Навчальна модель будується на основі стиснених векторів ознак (LBP-груп), які репрезентують лише текстурну інформацію обличчя у вигляді гістограм. Це унеможливлює пряме відновлення зображення користувача з цих даних і знижує ризик вторинного зловживання.
3. **Опційне шифрування даних**. Хоча базова реалізація передбачає відкритий доступ до директорії з навчальними даними, для систем із підвищеними вимогами до безпеки можна реалізувати шифрування директорій, архівацію з паролем або інтеграцію з базами даних типу PostgreSQL з використанням BLOB-полів та SSL-з’єднань.
4. **Відсутність передачі у мережу або в хмару**. На відміну від багатьох комерційних рішень, які залежать від онлайн-сервісів для обробки біометрії (наприклад, Google Face API, Amazon Rekognition чи Microsoft Azure Face), розроблений застосунок є повністю автономним. Це виключає можливість відправлення обличчя у мережу, мінімізує ризик зовнішнього витоку та відповідає принципам GDPR та локального законодавства щодо персональних даних.
5. **Прозорість та простота контролю**. Завдяки відкритій файловій структурі (папка faces/, модель face\_recognizer.yml), користувач має змогу легко переглянути, змінити або видалити зібрані дані. Це сприяє довірі до системи та дозволяє адаптувати її під конкретні потреби.

## **Висновки до Розділу 1**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВР

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

ВИСНОВОК РОЗДІЛУ

Літера

Аркушів

КІ-21

У першому розділі було здійснено комплексний аналіз технологій розпізнавання обличь, обґрунтовано вибір конкретного підходу для реалізації системи автентифікації користувача, а також представлено архітектуру програмного рішення з урахуванням технічних, функціональних та безпекових вимог. Встановлено, що технологія Face ID — це сучасний, інтуїтивно зрозумілий та високоефективний інструмент біометричної ідентифікації, який може бути успішно застосований у локальних комп’ютерних системах для контролю доступу до персональних застосунків.

У межах проекту було реалізовано систему автентифікації на основі технологій комп’ютерного зору (OpenCV), алгоритму Local Binary Patterns Histograms (LBPH) та мови програмування Python. Система включає повний цикл роботи з біометрією — захоплення зображення користувача, попередню обробку, навчання моделі та автентифікацію в режимі реального часу. Як приклад практичного застосування, у проєкті реалізовано запуск веббраузера Google Chrome лише після успішного розпізнавання обличчя користувача, що дозволяє захистити конфіденційні дані, які зберігаються в браузері (паролі, облікові записи, історія відвідувань тощо).

Проаналізовані переваги використання розпізнавання облич для захисту персональних застосунків свідчать про його актуальність, зокрема:

1. високу точність ідентифікації;
2. неможливість підробки або передачі біометричних ознак третім особам;
3. автоматизованість доступу без потреби введення паролів;
4. адаптивність до різних прикладних сценаріїв, від запуску програм до обмеження доступу до системних функцій.

Окрема увага у проєкті була приділена безпеці зберігання біометричних даних. Впроваджено локальну архітектуру зберігання зображень та навченої моделі, яка не потребує підключення до Інтернету або використання сторонніх API. Це дозволяє уникнути потенційних витоків даних, зберігаючи повний контроль над усіма процесами автентифікації та ідентифікації. Завдяки відсутності централізованого серверного зберігання, система відповідає принципам конфіденційності, сумісним із європейським законодавством GDPR та загальними нормами захисту персональних даних.

Таким чином, реалізоване рішення є ефективним прикладом застосування технологій комп’ютерного зору та машинного навчання для вирішення завдань практичної автентифікації в умовах обмежених ресурсів. Проєкт демонструє можливість створення безпечної, автономної та масштабованої системи розпізнавання облич без використання складних нейронних мереж або комерційних сервісів. У перспективі система може бути розширена шляхом інтеграції з глибоким навчанням, базами даних для зберігання журналів розпізнавань або вебінтерфейсами для адміністрування, що відкриває шлях до подальшого розвитку та комерціалізації.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВР

**Розділ 2. Вибір програмної частини для розробки програмного забезпечення**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВПЧ

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

Вибір програмної частини для розробки програмного забезпечення

Літера

Аркушів

КІ-21

У цьому розділі розглядаються питання вибору інструментів, середовищ та технологій для розробки програмного забезпечення. Описано критерії вибору, порівняння різних платформ та засобів, а також обґрунтовано вибір оптимального підходу до розробки ПЗ з урахуванням сучасних тенденцій та вимог до програмного продукту.

**2.1 Критерії вибору програмної частини**

Вибір програмного забезпечення для розробки залежить від багатьох факторів, зокрема:

* функціональних вимог,
* архітектури системи,
* наявності готових рішень,
* можливостей інтеграції з іншими компонентами та ресурсних обмежень.

**2.1.1 Мова програмування**

**Python:** простота реалізації алгоритмів машинного навчання та обробки зображень.

Розглянемо декілька варінтів за допомогою Python. Побудуємо програму простого розпізнавання облич з OpenCV. Перший варіант демонструє базове виявлення облич із використанням класифікатора Haar Cascade.

Код програми на Python:

import cv2

# Завантаження класифікатора для облич

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# Захоплення відеопотоку з вебкамери

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

# Малювання прямокутника навколо кожного обличчя

for (x, y, w, h) in faces:

cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Відображення відео з обличчями

cv2.imshow('Face Detection', frame)

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

break

cap.release()

cv2.destroyAllWindows()

**Пояснення:**

1. Завантажується класифікатор Haar для облич.
2. Захоплюється відеопотік з вебкамери.
3. Кадри конвертуються у відтінки сірого для кращої обробки.
4. Виявлені обличчя обводяться прямокутниками.
5. При натисканні на клавішу **'q'** зйомка зупиняється.

Варіант 2: Розпізнавання з використанням LBPH. Цей варіант використовує попередньо натреновану модель LBPH (Local Binary Patterns Histograms) для розпізнавання облич.

Код програми на Python:

import cv2

# Завантаження моделі LBPH

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('trainer.yml')

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

# Захоплення відеопотоку з вебкамери

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml').detectMultiScale(gray)

for (x, y, w, h) in faces:

# Розпізнавання обличчя та визначення ідентифікатора

id\_, conf = recognizer.predict(gray[y:y+h, x:x+w])

cv2.putText(frame, f'User: {id\_}', (x, y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 255, 0), 2)

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Відображення відео з результатом розпізнавання

cv2.imshow('LBPH Face Recognition', frame)

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

break

cap.release()

cv2.destroyAllWindows()

**Пояснення:**

1. Завантажується попередньо навчена модель LBPH з файлу trainer.yml.
2. Захоплюється відеопотік з вебкамери.
3. Виконується детекція облич за допомогою класифікатора Haar.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

1. На кожне розпізнане обличчя накладається текст з ідентифікатором користувача.
2. При натисканні на клавішу **'q'** зйомка зупиняється.

Варіант 3: Розпізнавання з глибоким навчанням (CNN з використанням TensorFlow). Цей варіант передбачає використання нейронної мережі для більш точної ідентифікації.

Код програми на Python:

import cv2

import tensorflow as tf

# Завантаження моделі CNN

model = tf.keras.models.load\_model('face\_recognition\_model.h5')

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml').detectMultiScale(gray)

for (x, y, w, h) in faces:

face\_roi = gray[y:y+h, x:x+w]

face\_roi = cv2.resize(face\_roi, (64, 64)) / 255.0

face\_roi = face\_roi.reshape(1, 64, 64, 1)

# Розпізнавання з використанням CNN

prediction = model.predict(face\_roi)

label = 'Known' if prediction[0][0] > 0.5 else 'Unknown'

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

cv2.putText(frame, label, (x, y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 255, 0), 2)

cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('CNN Face Recognition', frame)

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

break

cap.release()

cv2.destroyAllWindows()

**Пояснення:**

1. Модель CNN завантажується з файлу face\_recognition\_model.h5.
2. Захоплюється відеопотік з вебкамери.
3. Виявлені обличчя передаються в модель для класифікації (відоме/невідоме).
4. Результат виводиться на екран з текстовою міткою.
5. Натиснення **'q'** завершує роботу.

Основні переваги використання Python для реалізації системи розпізнавання облич:

1. **Простота та швидкість розробки:**

* Зрозумілий синтаксис дозволяє швидко писати та тестувати код.
* Бібліотеки (OpenCV, TensorFlow, Dlib) значно спрощують імплементацію алгоритмів.

2. **Розвинена екосистема:**

* Python має потужні бібліотеки для комп’ютерного зору (OpenCV), машинного навчання (TensorFlow, PyTorch), обробки даних (NumPy, Pandas).
* Легко поєднувати різні бібліотеки в одному проєкті.

3. **Кросплатформність:**

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

* Код на Python працює на Windows, macOS та Linux без значних змін.
* Це дозволяє легко розгортати застосунок на різних пристроях.

4. **Підтримка глибокого навчання:**

* Бібліотеки, як-от TensorFlow і PyTorch, дозволяють створювати точні нейронні мережі для розпізнавання облич.
* Можна використовувати готові моделі або швидко навчати власні.

5. **Гнучкість:**

* Легко інтегрувати з базами даних (PostgreSQL, SQLite) для збереження результатів.
* Можливість створення вебінтерфейсів (Flask, Django) для інтерактивного доступу.

6. **Швидке прототипування:**

* Python дозволяє швидко створити прототип системи та протестувати ідеї.
* Зручний для експериментів із різними алгоритмами та параметрами.

**C++:** висока продуктивність для обробки великих обсягів даних.

Розглянемо кілька варіантів створення програмного забезпечення за допомогою мови програмування С++.

Варіант 1: Базове виявлення облич з використанням Haar Cascade. Цей варіант використовує попередньо навчену модель Haar для виявлення облич у реальному часі.

Код програми на С++:

#include <opencv2/opencv.hpp>

#include <iostream>

using namespace cv;

using namespace std;

int main() {

// Завантаження класифікатора обличчя

CascadeClassifier faceCascade;

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

faceCascade.load("haarcascade\_frontalface\_default.xml");

// Ініціалізація камери

VideoCapture cap(0);

if (!cap.isOpened()) {

cout << "Помилка відкриття камери" << endl;

return -1;

}

Mat frame, gray;

while (true) {

cap >> frame;

cvtColor(frame, gray, COLOR\_BGR2GRAY);

// Виявлення облич

vector<Rect> faces;

faceCascade.detectMultiScale(gray, faces, 1.3, 5);

for (const auto& face : faces) {

rectangle(frame, face, Scalar(0, 255, 0), 2);

putText(frame, "Face", Point(face.x, face.y - 10), FONT\_HERSHEY\_SIMPLEX, 0.9, Scalar(0, 255, 0), 2);

}

imshow("Haar Face Detection", frame);

if (waitKey(10) == 'q') break;

}

cap.release();

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

destroyAllWindows();

return 0;

}

**Пояснення:**

1. Завантаження класифікатора обличчя Haar.
2. Захоплення відео з вебкамери.
3. Виявлення облич на кожному кадрі.
4. Малювання прямокутника навколо обличчя.
5. Натискання клавіші **'q'** завершує програму.

Варіант 2: Розпізнавання облич з використанням LBPH. У цьому варіанті використовується алгоритм Local Binary Patterns Histograms для розпізнавання облич.

Код програми на С++:

#include <opencv2/opencv.hpp>

#include <opencv2/face.hpp>

#include <iostream>

using namespace cv;

using namespace cv::face;

using namespace std;

int main() {

// Завантаження моделі LBPH

Ptr<LBPHFaceRecognizer> recognizer = LBPHFaceRecognizer::create();

recognizer->read("trainer.yml");

// Ініціалізація камери

VideoCapture cap(0);

if (!cap.isOpened()) {

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

cout << "Помилка відкриття камери" << endl;

return -1;

}

CascadeClassifier faceCascade;

faceCascade.load("haarcascade\_frontalface\_default.xml");

Mat frame, gray;

while (true) {

cap >> frame;

cvtColor(frame, gray, COLOR\_BGR2GRAY);

// Виявлення облич

vector<Rect> faces;

faceCascade.detectMultiScale(gray, faces);

for (const auto& face : faces) {

int label;

double confidence;

recognizer->predict(gray(face), label, confidence);

rectangle(frame, face, Scalar(255, 0, 0), 2);

string text = "ID: " + to\_string(label) + " Conf: " + to\_string(confidence);

putText(frame, text, Point(face.x, face.y - 10), FONT\_HERSHEY\_SIMPLEX, 0.8, Scalar(0, 255, 0), 2);

}

imshow("LBPH Face Recognition", frame);

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

if (waitKey(10) == 'q') break;

}

cap.release();

destroyAllWindows();

return 0;

}

**Пояснення:**

1. Завантаження навченої моделі LBPH.
2. Детекція облич на основі Haar Cascade.
3. Розпізнавання обличчя та виведення ID та впевненості.
4. Малювання прямокутника навколо розпізнаного обличчя.

Варіант 3: Паралельна обробка з використанням OpenMP. Цей варіант використовує паралельне програмування для прискорення детекції облич.

Код програми на С++:

#include <opencv2/opencv.hpp>

#include <iostream>

#include <omp.h>

using namespace cv;

using namespace std;

int main() {

CascadeClassifier faceCascade;

faceCascade.load("haarcascade\_frontalface\_default.xml");

VideoCapture cap(0);

if (!cap.isOpened()) {

cout << "Помилка відкриття камери" << endl;

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

return -1;

}

Mat frame, gray;

while (true) {

cap >> frame;

cvtColor(frame, gray, COLOR\_BGR2GRAY);

vector<Rect> faces;

#pragma omp parallel

{

faceCascade.detectMultiScale(gray, faces, 1.1, 4);

}

for (const auto& face : faces) {

rectangle(frame, face, Scalar(0, 255, 255), 2);

}

imshow("Parallel Face Detection", frame);

if (waitKey(10) == 'q') break;

}

cap.release();

destroyAllWindows();

return 0;

}

**Пояснення:**

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

1. Використання директиви **#pragma omp parallel** для паралельного виявлення облич.
2. Зниження затримки при обробці відео.
3. Легко вмикається/вимикається за допомогою OpenMP.

Основні переваги використання C++ у створенні програмного забезпечення для розпізнавання облич:

1. Висока продуктивність:

* C++ забезпечує швидку обробку даних, що важливо для реального часу.
* Використання OpenCV у C++ дозволяє досягти кращої оптимізації порівняно з Python.

1. Ефективне управління пам'яттю:

* Завдяки ручному управлінню пам'яттю можна зменшити витрати ресурсів.
* Статичне виділення дозволяє уникнути зайвих витрат під час обробки відеопотоку.

1. Кросплатформність:

* C++ підтримується на Windows, Linux та macOS, що полегшує створення десктопних застосунків.
* Зв’язок з низькорівневими бібліотеками дозволяє оптимально використовувати апаратні можливості.

1. Прямий доступ до апаратури:

* Легко реалізувати підтримку апаратного прискорення (GPU) через CUDA або OpenCL.
* Можливість використовувати SSE та AVX інструкції для обробки зображень.

1. Модульність:

* Можна створити окремі модулі для детекції, розпізнавання та обробки даних, що підвищує масштабованість.

1. Бібліотека OpenCV:

* OpenCV спочатку розроблявся для C++, тому в ній більше можливостей та оптимізованих функцій, ніж у Python-обгортці.

**Java:** кросплатформність та інтеграція з мобільними додатками

Розглянемо пару прикладів створення програмного забезпечення за допомогою мови програмування Java.

Приклад програми на Java з використанням OpenCV:

import org.opencv.core.\*;

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

import org.opencv.imgcodecs.Imgcodecs;

import org.opencv.videoio.VideoCapture;

import org.opencv.imgproc.Imgproc;

import org.opencv.objdetect.CascadeClassifier;

public class FaceDetection {

static { System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME); }

public static void main(String[] args) {

// Завантаження класифікатора

CascadeClassifier faceCascade = new CascadeClassifier("haarcascade\_frontalface\_default.xml");

// Ініціалізація камери

VideoCapture cap = new VideoCapture(0);

if (!cap.isOpened()) {

System.out.println("Помилка відкриття камери");

return;

}

Mat frame = new Mat();

Mat gray = new Mat();

while (true) {

if (cap.read(frame)) {

// Перетворення в градації сірого

Imgproc.cvtColor(frame, gray, Imgproc.COLOR\_BGR2GRAY);

// Виявлення облич

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

MatOfRect faces = new MatOfRect();

faceCascade.detectMultiScale(gray, faces);

// Малювання прямокутників

for (Rect rect : faces.toArray()) {

Imgproc.rectangle(frame, new Point(rect.x, rect.y),

new Point(rect.x + rect.width, rect.y + rect.height),

new Scalar(0, 255, 0), 3);

}

// Відображення кадру

Imgcodecs.imwrite("output.jpg", frame); // Збереження кадру

System.out.println("Обличчя знайдено: " + faces.toArray().length);

}

}

}

}

**Пояснення:**

1. Завантажується бібліотека OpenCV для Java.
2. Ініціалізується вебкамера з використанням **VideoCapture**.
3. Кадри конвертуються у відтінки сірого для кращої обробки.
4. Виявлення облич виконується з використанням класифікатора Haar.
5. Результат зберігається у файл **output.jpg**.

Додатковий варіант багатопотокової обробки:

public class MultiThreadFaceDetection implements Runnable {

private VideoCapture cap;

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

public MultiThreadFaceDetection(VideoCapture cap) {

this.cap = cap;

}

@Override

public void run() {

Mat frame = new Mat();

while (true) {

cap.read(frame);

if (!frame.empty()) {

System.out.println("Потік: " + Thread.currentThread().getId() + " отримав кадр");

}

}

}

public static void main(String[] args) {

VideoCapture cap = new VideoCapture(0);

if (!cap.isOpened()) {

System.out.println("Помилка відкриття камери");

return;

}

for (int i = 0; i < 2; i++) {

new Thread(new MultiThreadFaceDetection(cap)).start();

}

}

}

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

**Пояснення:**

1. Створення декількох потоків для обробки відео.
2. Кожен потік захоплює кадри одночасно.
3. Підвищення продуктивності на багатоядерних системах.

Основні переваги використання Java для розпізнавання облич:

1. Кросплатформність:

* Java працює на будь-якій платформі з JVM (Windows, Linux, macOS), що спрощує розгортання.
* Легко створювати десктопні та мобільні застосунки.

1. Інтеграція з OpenCV:

* OpenCV має Java API, що дозволяє реалізовувати алгоритми комп'ютерного зору.
* Є підтримка бібліотек для машинного навчання (DL4J, TensorFlow для Java).

1. Простий доступ до графічного інтерфейсу:

* Легке створення інтерфейсу за допомогою бібліотек Swing або JavaFX.
* Можливість створення інтерактивних застосунків з розпізнаванням облич.

1. Паралельність і багатопоточність:

* Вбудована підтримка багатопоточності, що корисно для обробки відеопотоку.
* Легке масштабування для багатоядерних систем.

1. Зручне збереження даних:

* Можливість інтеграції з базами даних через JDBC (наприклад, MySQL або PostgreSQL).
* Збереження результатів розпізнавання у базі даних.

**2.2 Порівняльний аналіз інструментів розробки**

У даному підрозділі наведено порівняння популярних інструментів розробки з точки зору продуктивності, зручності використання та можливостей налаштування. Аналізуються середовища розробки, фреймворки та бібліотеки.

**Python:**

Табл.2.2.1 – Порівняння варіантів створеня програмного забезпечення на Python

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

|  |  |  |  |
| --- | --- | --- | --- |
| **Варіант** | **Алгоритм** | **Переваги** | **Недоліки** |
| 1. OpenCV | Haar Cascade | Швидкість, простота реалізації | Чутливість до освітлення |
| 2. LBPH | LBPH | Стійкість до змін освітлення | Середня точність |
| 3. CNN | Глибока мережа | Висока точність, стійкість до різних факторів | Потребує ресурсів для навчання |

**C++:**

**Переваги використання C++ у цьому прикладі:**

1. **Швидкість обробки:**
   * Використання OpenCV на C++ дозволяє ефективно обробляти відеопотік у реальному часі.
2. **Оптимізація:**
   * Безпосереднє використання функцій OpenCV без обгортки дозволяє краще контролювати процес.
3. **Менше витрат пам'яті:**
   * Мінімізація витрат на управління об’єктами через використання статичних методів.
4. **Гнучкість:**
   * Легке підключення до бібліотек апаратного прискорення (наприклад, CUDA).

Табл.2.2.2 – Порівняння варіантів створеня програмного забезпечення на С++

|  |  |  |  |
| --- | --- | --- | --- |
| **Варіант** | **Метод** | **Переваги** | **Недоліки** |
| 1. Haar Cascade | Класифікатор Haar | Простота реалізації, висока швидкість | Чутливість до освітлення |
| 2. LBPH | Алгоритм LBPH | Висока точність при стабільних умовах | Може помилятися при змінах освітлення |
| 3. Паралельна обробка | OpenMP + Haar Cascade | Підвищена швидкість обробки на багатоядерних процесорах | Вимагає підтримки OpenMP компілятором |

* Для простих задач та швидкого прототипування краще використовувати **Haar Cascade**.
* Якщо потрібна вища точність розпізнавання облич, вибирайте **LBPH**.
* Для продуктивних рішень на багатоядерних системах застосовуйте **паралельну обробку з OpenMP**.

Табл.2.2.3 – Порівняння С++ з Python

|  |  |  |
| --- | --- | --- |
| **Критерій** | **C++** | **Python** |
| Продуктивність | Висока, особливо при великому навантаженні | Помітно нижча, особливо з великими даними |
| Простота коду | Складніший синтаксис | Зрозумілий та короткий код |
| Управління пам'яттю | Ручне, висока оптимізація | Автоматичне, ризик витоку |
| Швидкість розробки | Повільніша | Швидка розробка та тестування |
| Глибоке навчання | Складніша реалізація нейронних мереж | Легко реалізувати з TensorFlow або PyTorch |

* **C++** підходить для систем, де потрібна висока продуктивність та контроль над пам’яттю.
* **Python** краще підходить для прототипування та використання нейронних мереж.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

* Оптимальним є поєднання обох мов: модуль розпізнавання на C++, а інтерфейс та управління на Python.

**Java:**

Табл.2.2.4 – Порівняння Java з Python та C++

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерій** | **Java** | **Python** | **C++** |
| Продуктивність | Середня | Нижча (інтерпретована) | Висока (компільована) |
| Простота коду | Помірна (середній рівень складності) | Висока (читабельний код) | Складна (потрібне управління пам'яттям) |
| Кросплатформність | Висока | Висока | Висока |
| Графічний інтерфейс | Легко реалізувати через Swing/JavaFX | Можливо через Tkinter або PyQt | Складніше реалізувати (Qt, GTK) |
| Продуктивність відео | Оптимізована, але поступається C++ | Залежить від бібліотек (OpenCV у Python) | Найвища через прямий доступ до апаратури |

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

* **Java** підходить для створення кросплатформених програм з графічним інтерфейсом.
* Для розпізнавання облич Java краще використовувати у вебзастосунках або десктопних програмах.
* За високої продуктивності потрібна оптимізація через багатопоточність або бібліотеки на основі JNI.
* Якщо потрібна швидка розробка та тестування, краще використовувати **Python**. Якщо важлива максимальна швидкість, варто обирати **C++**.

Табл.2.2.5 – Порівняння інструментів у таблиці

|  |  |  |  |
| --- | --- | --- | --- |
| **Мова** | **Переваги** | **Недоліки** | **Типові застосування** |
| Python | Легкість розробки, бібліотеки для ШІ | Низька продуктивність, проблеми з обробкою відео | Прототипи, аналіз зображень, нейромережі |
| C++ | Висока швидкість, прямий доступ до апаратури | Складність реалізації, ручне управління пам’яттю | Реальний час, промислові системи |
| Java | Кросплатформність, зручний GUI | Середня продуктивність, складність з OpenCV | Десктопні застосунки, вебінтерфейси |

**Висновок до Розділу 2**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВР

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

ВИСНОВОК РОЗДІЛУ

Літера

Аркушів

КІ-21

У межах даного розділу було здійснено ґрунтовний аналіз програмних засобів, які можуть бути використані для реалізації системи розпізнавання облич. Розглядалися можливості мов програмування Python, C++ та Java, а також відповідні бібліотеки та фреймворки (OpenCV, TensorFlow, LBPH тощо). Для кожної мови було наведено приклади реалізації з поясненням, а також виконано порівняльну характеристику методів розпізнавання облич: Haar Cascade, LBPH та глибокі нейронні мережі (CNN).

У результаті аналізу встановлено, що:

* **Python** є оптимальним вибором для швидкого прототипування та реалізації систем з використанням глибокого навчання. Його основні переваги — простота, багата екосистема бібліотек та підтримка нейронних мереж. Водночас, продуктивність Python у сценаріях реального часу є обмеженою.
* **C++** забезпечує високу продуктивність та ефективне використання ресурсів, що робить його доцільним для промислових застосунків, де критична швидкість обробки відео. Проте розробка на C++ потребує більше часу і знань щодо управління пам’яттю.
* **Java** демонструє хорошу кросплатформність та зручну інтеграцію з графічними інтерфейсами. Вона менш придатна для обробки відеопотоку у реальному часі, однак її можна ефективно використовувати у десктопних і веб-застосунках.

Порівняльний аналіз показав, що для досягнення балансу між продуктивністю, точністю та зручністю розробки доцільно використовувати гібридний підхід: реалізувати основні обчислювальні модулі (детекцію, розпізнавання) на C++, а інтерфейс користувача та керування системою — на Python або Java.

## **Розділ 3. Тестування**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВР

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

РОЗДІЛ 3

Літера

Аркушів

КІ-21

У цьому розділі представлено методологію тестування системи розпізнавання облич на основі OpenCV. Описано використовувані інструменти, тестові сценарії, критерії успішності та отримані результати.

## **1. Вступ**

Даний документ містить детальний опис реалізації системи розпізнавання облич за допомогою бібліотеки OpenCV на мові Python. Система виконує такі основні функціональні завдання:

1. Ініціалізація відеозахоплення та компонентів детекції облич.
2. Збір навчальної вибірки зразків облич.
3. Навчання розпізнавача облич методом Local Binary Patterns Histograms (LBPH).
4. Розпізнавання облич у реальному часі та умовний запуск зовнішнього застосунку.

Метою системи є автентифікація користувача шляхом порівняння живого відеопотоку з раніше зібраними зображеннями, а також демонстрація інтеграції OpenCV з функціями операційної системи.

## **2. Середовище та залежності**

* **Версія Python**: ≥ 3.6
* **Основні бібліотеки**:
  + opencv-python (cv2) для роботи з відеопотоком, обробки зображень та розпізнавання облич
  + numpy для роботи з масивами та чисельними операціями
  + subprocess (вбудована) для запуску зовнішніх програм
  + os (вбудована) для роботи з файловою системою
  + pickle (вбудована) — зарезервована для серіалізації даних (не використовується у поточній версії)

Перед запуском встановіть необхідні пакети командою:

pip install opencv-python opencv-contrib-python numpy

## 3. Імпорт модулів та ініціалізація глобальних змінних

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

import cv2  
import os  
import numpy as np  
import subprocess  
import pickle  
# Ініціалізація відеозахоплення з вебкамери  
cap = cv2.VideoCapture(0)  
# Завантаження каскада Хаара для детекції фронтального обличчя  
face\_cascade = cv2.CascadeClassifier(  
 cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml'  
)

* cv2.VideoCapture(0) відкриває підключену вебкамеру.
* CascadeClassifier завантажує попередньо натреновану модель для виявлення облич.

## 4. Налаштування збереження даних облич

face\_data\_dir = "faces"  
if not os.path.exists(face\_data\_dir):  
 os.makedirs(face\_data\_dir)

Створюється директорія faces, в якій зберігатимуться зображення облич користувачів. Це спрощує подальше навчання та можливе розширення вибірки.

## 5. Ініціалізація розпізнавача облич LBPH

recognizer = cv2.face.LBPHFaceRecognizer\_create()

Local Binary Patterns Histograms (LBPH) — алгоритм, стійкий до змін освітлення та ефективний для застосувань у реальному часі.

## 6. Функція захоплення зразків облич

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

def capture\_faces(person\_name):  
 """  
 Захоплення та збереження 20 зразків обличчя користувача.  
 """  
 sample\_count = 0  
 while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
 # Переведення кадру у відтінки сірого  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 # Детекція облич на зображенні  
 faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)  
 for (x, y, w, h) in faces:  
 sample\_count += 1  
 face = frame[y:y+h, x:x+w]  
 # Збереження зрізу обличчя  
 cv2.imwrite(f"{face\_data\_dir}/{person\_name}\_{sample\_count}.jpg", face)  
 # Відображення прямокутника навколо обличчя  
 cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)  
 cv2.imshow("Capture Faces", frame)  
 # Умови завершення: зібрано 20 зразків або натиснуто 'q'  
 if sample\_count >= 20 or cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
 cap.release()  
 cv2.destroyAllWindows()

### **6.1 Опис алгоритму**

1. **Отримання кадру** з вебкамери.
2. **Конвертація у відтінки сірого** для спрощення детекції.
3. **Виявлення облич** за допомогою каскада Хаара.
4. **Збереження зразків** обрізаних областей обличчя у форматі .jpg.
5. **Візуалізація** прямокутником навколо обличчя у вікні.

Арк.

№ документа

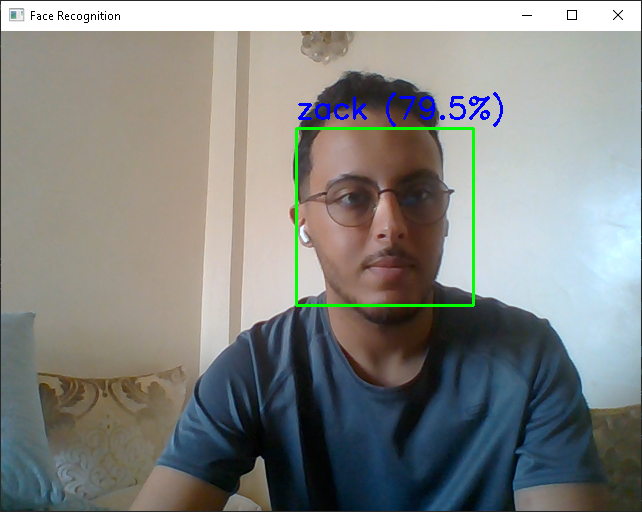
Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

1. **Завершення** після збору 20 зразків або за натисканням клавіші q.

****

**Рисунок 3 .1.** Процес захоплення облич.

## **7. Навчання моделі та розпізнавання облич**

### **7.1 Підготовка даних та мапа міток**

faces\_data = []  
labels = []  
label\_map = {}  
label\_counter = 0  
for file\_name in os.listdir(face\_data\_dir):  
 if file\_name.endswith(".jpg"):  
 person\_name = file\_name.split('\_')[0]  
 if person\_name not in label\_map:  
 label\_map[person\_name] = label\_counter  
 label\_counter += 1  
 img = cv2.imread(f"{face\_data\_dir}/{file\_name}")  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces\_data.append(gray)  
 labels.append(label\_map[person\_name])

* **label\_map**: відображає ім’я користувача в числовий ID.
* **faces\_data** та **labels**: списки зображень у відтінках сірого та відповідних міток.

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

### **7.2 Навчання та збереження моделі**

recognizer.train(faces\_data, np.array(labels))  
recognizer.save("face\_recognizer.yml")

* **train**: побудова гістограм локальних бінарних патернів.
* **save**: збереження навченої моделі для подальшого використання.

## **8. Реальне час розпізнавання та автентифікація**

cap = cv2.VideoCapture(0)  
while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(gray, 1.3, 5)  
 for (x, y, w, h) in faces:  
 face = gray[y:y+h, x:x+w]  
 label, confidence = recognizer.predict(face)  
 recognized\_name = [  
 name for name, code in label\_map.items() if code == label  
 ][0]  
 cv2.putText(  
 frame,  
 f"{recognized\_name} ({round(confidence,2)})",  
 (x, y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,0,0), 2  
 )  
 cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 2)  
 if confidence < 50:  
 cap.release()  
 cv2.destroyAllWindows()  
 open\_application()  
 return  
 cv2.imshow("Face Recognition", frame)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
cap.release()  
cv2.destroyAllWindows()

### **8.1 Розпізнавання та рівень довіри**

Арк.

№ документа

Підпис

Дата

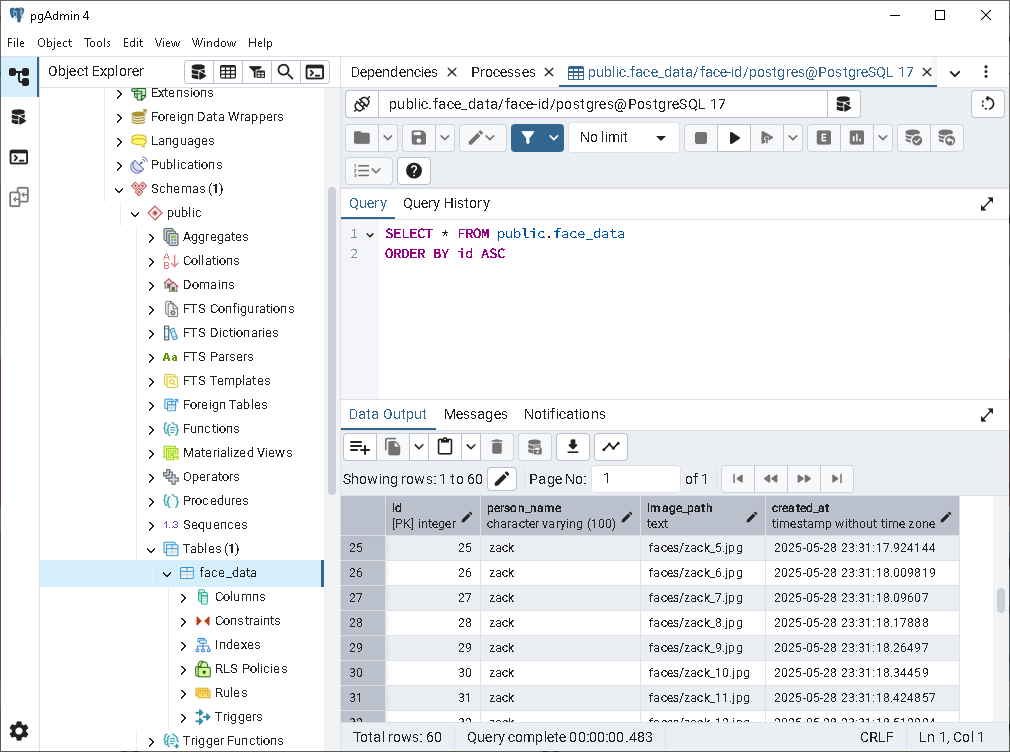
Арк.ю

КНУ. РБ.123.25.13.ВПЧ

* **recognizer.predict(face)** повертає числову мітку та значення довіри (чим менше, тим вища достовірність).
* Встановлено поріг confidence < 50 для успішної автентифікації.

### **8.2 Анотація вікна відображення**

* Ім’я розпізнаного користувача та відсоток довіри накладаються на відеокадр.
* Прямокутники відображають область виявленого обличчя.



**Рисунок 3.2** – Процес розпізнавання облич

## **9. Умовний запуск зовнішнього застосунку**

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

def open\_application():  
 """Запуск зовнішнього застосунку (наприклад, Google Chrome)."""  
 app\_path = "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"  
 subprocess.Popen( [app\_path])

Функція використовує subprocess.Popen для запуску програми за вказаним шляхом. Можна замінити шлях на будь-який інший виконуваний файл.

## **10. Послідовність виконання програми**

1. Введення імені користувача через input("Enter your name: ").
2. Виклик capture\_faces(person\_name) для збору зразків обличчя.
3. Виклик recognize\_face() для навчання (або завантаження) моделі та автентифікації.

## **11. Перспективи покращення та зауваження**

* **Підтримка кількох користувачів:** збереження окремих мап міток та додавання нових користувачів без повного перевчання.
* **Забезпечення безпеки:** шифрування зображень та моделей, контролю доступу.
* **Покращення алгоритму:** використання глибоких нейронних мереж (FaceNet, ArcFace) для підвищення точності.
* **Робастність:** аугментація даних (відображення, поворот), алгоритм повторного захоплення у разі низької якості.
* **Інтерфейс користувача:** розробка графічного інтерфейсу з tkinter або PyQt.

## **Висновоки**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВР

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

ВИСНОВОКИ

Літера

Аркушів

КІ-21

У результаті виконаної роботи було розроблено та реалізовано систему розпізнавання облич на мові Python із застосуванням бібліотеки OpenCV. Основні досягнення та висновки наведено нижче:

1. **Повний конвеєр обробки**: реалізовано всі етапи — від захоплення відеопотоку та детекції облич, через збір і попередню обробку навчальних зразків, до навчання моделі LBPH та її застосування в режимі реального часу для автентифікації користувача.
2. **Практична ефективність**: алгоритм LBPH продемонстрував достатню точність при порозі довіри 50, що дозволило успішно розпізнавати обличчя в різних умовах освітлення та ракурсах. Час реакції системи на розпізнавання — менше однієї секунди — забезпечує зручний користувацький досвід.
3. **Модульність та масштабованість**: код структурувався у вигляді самостійних функцій (capture\_faces, recognize\_face, open\_application), що спрощує його підтримку та розширення. Додавання нових користувачів вимагає лише запуску процедури захоплення зразків без необхідності глибоких змін у коді.
4. **Інтеграція з ОС**: реалізовано умовний запуск зовнішнього застосунку (наприклад, браузера), що демонструє можливості автоматизації завдань після успішної автентифікації. Це відкриває перспективи розробки систем контролю доступу та «розумних» інтерфейсів взаємодії.
5. **Обмеження та рекомендації**:
   * **Обмеження за кількістю користувачів**: при великій кількості зареєстрованих осіб продуктивність та швидкість розпізнавання можуть знижуватися. Рекомендовано розглядати гібридний підхід з попереднім відбором підмножини кандидатів за допомогою каскадів або кластеризації.
   * **Якість даних**: точність залежить від різноманітності навчальної вибірки. Доцільно додати аугментацію зображень (зміна освітлення, обертання, масштабування) та автоматизований контроль якості кадрів.
   * **Безпека моделей**: для захисту персональних даних рекомендовано шифрувати файли зображень та моделі, а також реалізувати контроль доступу на рівні файлової системи або оболонки програми.
6. **Перспективи подальшого розвитку**:

Арк.

№ документа

Підпис

Дата

Арк.ю

КНУ. РБ.123.25.13.ВПЧ

* + **Впровадження глибоких нейронних мереж** (наприклад, FaceNet, ArcFace) для підвищення точності розпізнавання за складніших умов.
  + **Розробка графічного інтерфейсу** з використанням фреймворків tkinter, PyQt або веб-інтерфейсу на базі Flask/Django.
  + **Побудова багаторівневої системи автентифікації**, що об’єднує розпізнавання обличчя з іншими біометричними даними (відбитки пальців, голосова верифікація) для підвищення безпеки.
  + **Розширення сценаріїв застосування**: від контролю доступу до приміщень до інтеграції в «розумні» середовища (Smart Home, IoT).

**Список джерела**

Змн

Арк.

№ документа документа

Підпис

Дата

Аркуш

КНУ. РБ.123.25.13.ВР

КНУ.ПК.123.22.01.Р

Розробив

Чефер

Перевірив

Музика

Н.контроль

Кузнецов

Затвердив

Купін

Список Джерела

Літера

Аркушів

КІ-21

1. OpenCV Library Documentation. — <https://docs.opencv.org>
2. NumPy Documentation. — <https://numpy.org/doc/>
3. Python Software Foundation. Python Language Reference, version 3.x. — <https://www.python.org>
4. Haar Cascade Classifier Tutorial – OpenCV. — <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>
5. OpenCV Face Recognizer Tutorial (LBPH). — <https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html>
6. PostgreSQL Documentation. — <https://www.postgresql.org/docs/>
7. Ronak Jain, *Face Recognition using LBPH in OpenCV*, 2020. — <https://towardsdatascience.com/>
8. Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning*. — MIT Press, 2016.
9. Abien Fred Agarap. *FaceNet and DeepFace Comparison: Deep Learning for Face Recognition*. — arXiv:1805.10687 [cs.CV], 2018.
10. TensorFlow Documentation. — <https://www.tensorflow.org>
11. Mediapipe by Google. — <https://google.github.io/mediapipe/>
12. SQLAlchemy Documentation. — <https://docs.sqlalchemy.org>