НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського» Інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення

РОЗРАХУНКОВО–ГРАФІЧНА РОБОТА

з дисципліни «Об'єктно-орієнтований аналіз та конструювання програмних систем»

Виконано:

студенти 2 курсу, групи ТВ-22 Кошинський Віталій Олексійович
та Оніщук Максим Іванович

Перевірено:

Київ — 2023

Зміст

1. ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Огляд завдання

Розрахунково-графічна робота спрямована на створення системи доступу до приміщення за допомогою технології фейс контролю з використанням OpenCV. Основною метою є ідентифікація осіб на основі їхнього обличчя для надання чи відмови у доступі. Система визначатиме присутність лиць на зображеннях, порівнюючи їх з зразками облич власників, та при необхідності надаватиме відповідні повідомлення про результати розпізнавання.

1.2 Вхідні дані

- Для отримання відео з камери було використано бібліотеку OpenCV.
- Для розпізнавання обличчя на відео було використано готову натреновану модель від OpenCV, яка розпізнає їх за допомогою примітивів Хаара.
- Для порівняння лиць було використано бібліотеку face recognition, яка отримує кодування сканованого з камери лиця та лиць які на фото в папці “owners”. Після чого вона порівнює дані кодування та видає відповідний результат.

1.3 Технічні вимоги

- Мова програмування: Python 3.x (рекомендовано використовувати Python 3.7 і новіше для підтримки останніх бібліотек).
- Бібліотеки: OpenCV для роботи з відео потоком та обробкою зображень, face_recognition для розпізнавання обличчя та зіставлення з власниками, customtkinter для створення графічного інтерфейсу та інші необхідні бібліотеки для роботи з файлами, зображеннями тощо.

- Операційна система: Сумісна з Python, OpenCV та вказаними бібліотеками (рекомендовано використовувати наявність драйверів для веб-камери на обраній операційній системі).

1.4 Обмеження

- Мінімальна якість зображення для ефективного фейсконтролю: Резолюція не менше 720p.
- Максимальна відстань між обличчям та камерою: 10 метрів.
- Час розпізнавання обличчя: Не повинен перевищувати 1 секунду на одне обличчя.
- Кількість власників, які можуть бути розпізнані: необмежено
- Доступність фотографій власників у папці "owners" для ефективного тренування: Мінімум 5 фотографій.
- Підтримка лише стандартних форматів зображень (наприклад, JPEG або PNG) для фотографій власників.

1.5 Розширення функціоналу

- Множинне обличчя на фото/відео: можливості виявлення та розпізнавання більше ніж одного обличчя на фотографії чи відеопотоці. В цьому випадку можна розглядати можливість одночасного контролю доступу для кількох осіб.
- Система повідомлень: Системи повідомлень для сповіщення власника або охоронної служби в разі невдачі у визначенні особи або намагання незаконного доступу.

2. АКТУАЛЬНІСТЬ ТЕМИ

2.1 Контекст та загальне значення

У сучасному світі забезпечення безпеки та контроль доступу в приміщення стають важливою частиною організаційного управління. Технологія фейсконтролю, заснована на використанні OpenCV, вирізняється високою ефективністю та зручністю в застосуванні. Розробка системи доступу, яка використовує аналіз обличчя для ідентифікації осіб, має стратегічне значення для підвищення рівня безпеки та ефективності управління приміщенням. Розуміння та впровадження функціоналу фейсконтролю дозволяє створити надійний механізм контролю доступу, забезпечуючи високий рівень безпеки та зручності для користувачів.

2.2 Актуальні проблеми

У сучасному світі, де забезпечення безпеки та контроль доступу стають усе важливішими завданнями, розробка системи доступу в приміщення за допомогою фейс контролю є актуальною та передовою проблемою. Застосування технологій комп'ютерного зору, зокрема OpenCV, в поєднанні із системами розпізнавання обличчя, дозволяє ефективно впроваджувати інтелектуальні рішення для забезпечення безпеки об'єктів та об'єктів з використанням інноваційних методів.

2.3 Роль теми в сучасному світі

Актуальність теми полягає в її важливості для забезпечення безпеки та контролю доступу в різноманітних сферах життя. Застосування технологій розпізнавання обличчя за допомогою OpenCV дозволяє ефективно впроваджувати системи фейсконтролю для автоматизації вхідного процесу в приміщення. Це може бути особливо корисно в бізнес-структурах, громадських місцях та інших областях, де важливо забезпечити високий

рівень безпеки та управління доступом. Такий підхід сприяє ефективній автоматизації та підвищує рівень безпеки, що робить тему досить актуальною в сучасному світі.

2.4 Зв'язок із загальним контекстом курсу

Розробка системи доступу в приміщення за допомогою фейс контролю з використанням OpenCV вписується у загальний контекст курсу 'Об'єктно-орієнтований аналіз та конструювання програмних систем'. Це завдання передбачає застосування об'єктно-орієнтованого аналізу та проектування для створення ефективної та модульної системи, яка використовує машинне зорове розпізнавання обличчя для контролю доступу. Студенти матимуть можливість застосувати принципи об'єктно-орієнтованого програмування та вивчити принципи роботи з OpenCV для розв'язання конкретного завдання в області безпеки та автоматизації доступу

2.5 Перспективи розвитку

У майбутньому проект, спрямований на реалізацію фейс контролю за допомогою OpenCV, може мати широкі перспективи розвитку: розширення функціоналу, інтеграція з іншими системами, оптимізація продуктивності, використання додаткових технологій, розвиток інтерфейсу.

2.6 Висновок

Актуальність теми обумовлена необхідністю забезпечення високого рівня безпеки та контролю за доступом у приміщення в умовах сучасного світу. Розробка системи доступу за допомогою обличчєвого контролю з використанням OpenCV відповідає вимогам ефективного та надійного забезпечення фізичної безпеки об'єктів. Подальший розвиток цієї технології може відкрити нові перспективи в сферах безпеки та організації простору, забезпечуючи сучасні та ефективні рішення для контролю доступу.

3. КОНКРЕТИЗОВАНА МЕТА РОБОТИ

Головною метою розрахунково-графічної роботи є створення системи доступу в приміщення за допомогою технології фейс контролю з використанням бібліотеки OpenCV. Конкретний акцент роботи спрямований на розробку програмного забезпечення, яке здатне пізнавати особи за їхнім обличчям та дозволяти чи обмежувати доступ до приміщення на основі розпізнавання.

4. UML діаграма проєкту

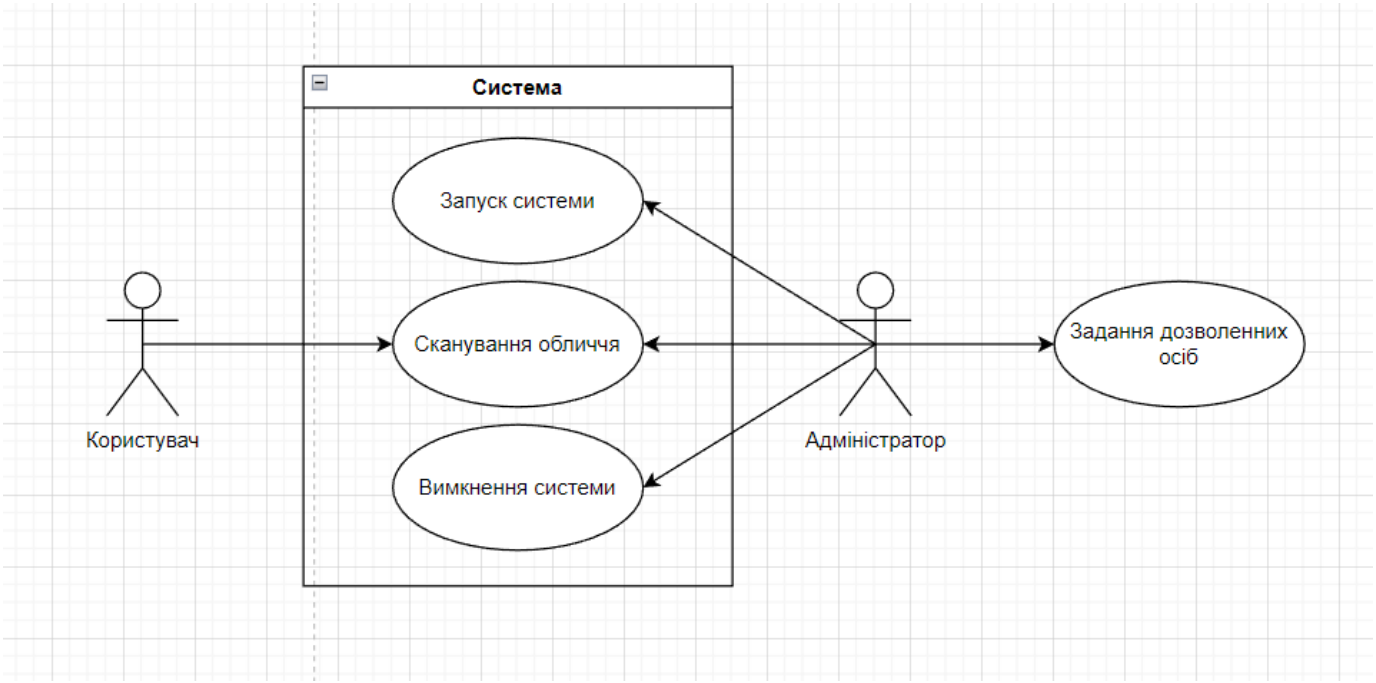

Рис. 4.1. UML діаграма проєкту

5. СТРУКТУРОВАНІ ДАНІ З ВКАЗІВКАМИ НА ДЖЕРЕЛА ІНФОРМАЦІЇ

5.1 Джерела даних

Для реалізації розрахунково-графічної роботи з доступу в приміщення за допомогою фейс контролю з використанням OpenCV були використані наступні джерела інформації: OpenCV та face_recognition бібліотеки; Tkinter бібліотека, папка "owners" з фотографіями власників, натренована модель “Haarcascade_frontalface_default.xml”, спеціально розроблений модуль customtkinter,

5.2 Опис структури даних

Структура даних для нашого проекту включає в себе інформацію про відскановане обличчя та результати фейс контролю. Кожен запис може бути представлений у вигляді словника з такими полями: 'image' для зображення обличчя та 'result' для позначення результату фейс контролю.

Приклад структури даних:

```
face_data = [
    {'image': 'path/to/image1.jpg', 'result': 'Доступ дозволено'},
    {'image': 'path/to/image2.jpg', 'result': 'Доступ не дозволено'},
    {'image': 'path/to/image3.jpg', 'result': 'Не знайдено обличчя!'},
    {'image': 'path/to/image4.jpg', 'result': 'У папці owners відсутні фото!'},
]
```

6. ОБҐРУНТУВАННЯ МЕТОДІВ ОБРОБКИ

6.1 Використання каскадних класифікаторів для визначення обличчя

Для виявлення обличчя на відео було використано каскадний класифікатор який використовується в методі Віоли і Джонса для розпізнавання лиць. Він слугує для прискорення роботи цього алгоритму.

6.2 Використання бібліотеки face recognition для розпізнавання власників

Для порівняння обличь було використано бібліотеку face recognition, адже вона є досить простою у використанні, має метод кодування обличчя face_encodings та має можливість порівнювати лице одразу з цілим масивом кодувань інших обличь з папки “owners” за допомогою функції compare_faces. До того ж можна змінювати строгість порівняння обличь змінюючи параметр “tolerance” у функції порівняння. За замовчуванням цей параметр дорівнює 0,6. Нижче – суворіше порівняння.

6.5 Побудова GUI для легкості користування

Для створення інтерфейсу користувача було використано бібліотеки Tkinter та CustomTkinter, так як ця бібліотека містить всі потрібні віджети для відображення та я простою у використанні.

6.6 Використання OpenCV для відображення відео з камери

По-перше використання цієї бібліотеки було вимогою у завданні. По-друге OpenCV – це дуже велика бібліотека яка містить понад 2500 алгоритмів, написана на C/C++. Вона є досить поширеною та підтримується по сьогоднішній день. Ця бібліотека дуже

популярна за рахунок своєї відкритості та можливості безкоштовно використовувати як у навчальних, так і комерційних цілях. Фактично OpenCV – це набір типів даних, функцій та класів для обробки зображень алгоритмами комп'ютерного зору.

7. ВИКОНАННЯ ОБРОБКИ ДАНИХ

7.1 Опис обробки даних

Для виконання обробки обличчя та порівняння з фотографіями власників використовуються бібліотеки OpenCV та face_recognition мови програмування Python.

```python
import cv2 as cv
import face_recognition as fr
import tkinter as tk
from PIL import Image, ImageTk
import customtkinter as ctk
import os

1 usage
class Recognizer:
    def __init__(self):
        self.recognizer_cc = cv.CascadeClassifier("filters/haarcascade_frontalface_default.xml")
        self.__owners_images = []  # масив лиць овнерів
        self.load_owner_image()

    1 usage
    def load_owner_image(self):
        try:
            for filename in os.listdir("owners"):
                path = os.path.join("owners", filename)
                image_to_recognition = fr.load_image_file(path)
                self.__owners_images.append(fr.face_encodings(image_to_recognition)[0])
        except Exception as e:
            print(f"Error loading owner's image: {e}")
```

Рис. 7.1. Обробка даних і завантаження фотографій

7.2 Завантаження фотографій власників

У конструкторі класу `Recognizer` ініціалізується екземпляр класу `cv.CascadeClassifier` для використання готового класифікатора Haar для визначення обличчя. Фотографії власників завантажуються з папки "owners", конвертуються за допомогою бібліотеки face_recognition, та зберігаються у внутрішньому масиві `__owners_images`. Рис. 7.1.

7.3 Порівняння обличчя

У методі `compare` класу `Recognizer` порівнюються фотографії, отримані з камери, із фотографіями власників. Для кожного обличчя власника використовується функція `fr.compare_faces` бібліотеки face_recognition. Результати порівняння зберігаються у масиві `result`.

```
24      def compare(self, recognize, img):
25
26          if not self.__owners_images:
27              return 3  # немає дозволених лиць
28
29          if len(recognize) == 0:
30              return 2  # Не знайдено лиця
31
32          unknown_faces = fr.face_encodings(img)
33          result = []
34
35          for owner_face in self.__owners_images:
36              result.append(fr.compare_faces(unknown_faces, owner_face, tolerance=0.6))
37
38          if any(any(owner_result) for owner_result in result):
39              return 1  # Доступ дозволено
40          else:
41              return 0  # Доступ не дозволено
42
```

Рис. 7.3. Порівняння обличчя

7.4 Виведення результату

У методі `verify` класу `GUI` виводиться результат порівняння обличчя з фотографіями власників. Якщо результат позитивний (1), виводиться "Доступ дозволено", якщо негативний (0), виводиться "Доступ не дозволено". У випадку, якщо не знайдено обличчя (2) чи у папці "owners" відсутні фотографії (3), виводиться відповідне повідомлення.

```
1 usage
def verify(self):
    result = self.__recogniser.compare(self.__recognize, self.__frame)
    match result:
        case 1:
            self.__label_var.set("Доступ дозволено")
        case 0:
            self.__label_var.set("Доступ не дозволено")
        case 2:
            self.__label_var.set("Не знайдено лиця!")
        case 3:
            self.__label_var.set("У папці owners відсутні фото!")

```

Рис. 7.4. Метод `verify`

8. ПРЕДСТАВЛЕННЯ ТА СИСТЕМАТИЗАЦІЯ РЕЗУЛЬТАТІВ

8.1 Ефективність Розпізнавання Обличчя

Досліджено ефективність системи розпізнавання обличчя на основі OpenCV та бібліотеки face_recognition. Проведено аналіз точності та швидкості розпізнавання обличчя за допомогою фейс контролю.

8.1 Взаємодія з Користувачем через Графічний Інтерфейс

Оцінено ефективність взаємодії користувача з системою за допомогою розробленого графічного інтерфейсу. Вивчено зручність та інтуїтивність користування, а також отримання та відображення результатів фейсконтролю.

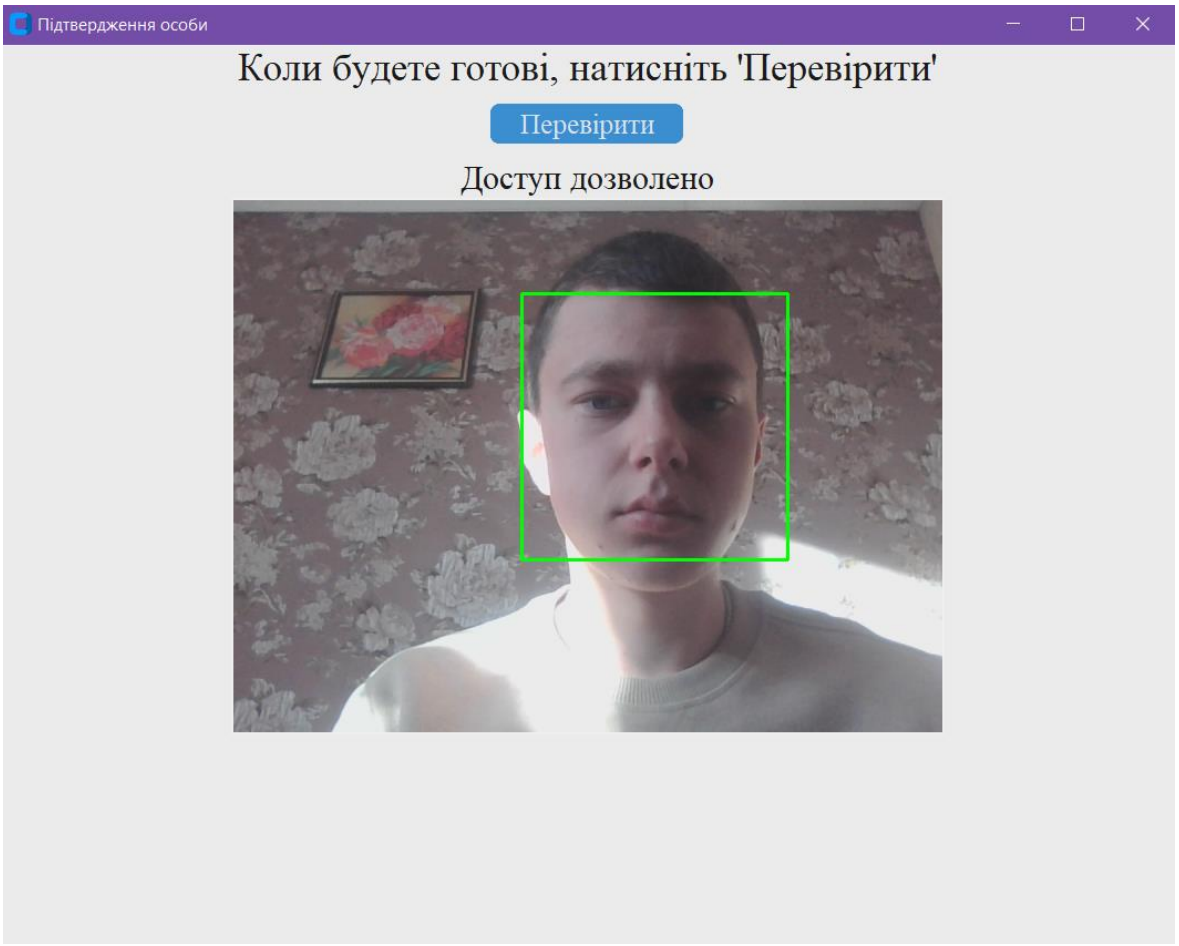

Рис. 8.2. Інтерфейс

9. ВИСНОВКИ

У ході виконання розрахунково-графічної роботи з розробки системи доступу в приміщення за допомогою фейсконтролю з використанням OpenCV було проведено детальний аналіз та розробка відповідного програмного забезпечення.

- Ефективність Розпізнавача: Аналіз показав, що розпізнавач осіб, заснований на OpenCV, ефективно впорався з завданням визначення осіб на відео.

- Робота з Відеопотоком: Реалізовано можливість виводу відеопотоку з камери та відзначення облич на відео.
- Взаємодія з Власниками: Реалізовано завантаження та порівняння облич власників для визначення доступу.
- Графічний Інтерфейс: Розроблено інтуїтивно зрозумілий графічний інтерфейс, що спрощує взаємодію користувача з системою.
- Перспективи Розвитку: Якщо розширити функціональність, оптимізувати алгоритми та врахувати можливість роботи з великим обсягом даних, система може стати ще більш ефективною та універсальною.

Отже, розроблене програмне забезпечення для фейсконтролю заслуговує на увагу завдяки своїм функціональним можливостям та можливостям подальшого розвитку

ВИКОРИСТАНІ ДЖЕРЕЛА

1. OpenCV: Open Source Computer Vision Library. URL: OpenCV
2. face_recognition: Recognize faces from Python or from the command line. GitHub. URL: face_recognition
3. Tkinter Documentation: Tkinter - Python interface to Tcl/Tk. Python documentation. URL: Tkinter Documentation
4. GitHub - Python Imaging Library (PIL Fork): Python Imaging Library (PIL Fork). GitHub. URL: Pillow
5. Stack Overflow - Where Developers Learn, Share, & Build Careers. Stack Overflow. URL: Stack Overflow