

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Петуховський Владислав Олександрович

**Розробка алгоритму розпізнавання облич у відеопотоці
кваліфікаційна робота**

здобувача вищої освіти першого (бакалаврського) рівня
освітньої програми «Інженерія програмного забезпечення»
за спеціальністю 121 Інженерія програмного забезпечення

Особистий підпис – _____

Науковий керівник – _____
(підпис)

доцент кафедри ІТС, кандидат
педагогічних наук, доцент
С.О. Переяславська
(посада, науковий ступінь, наукове звання,
ініціали, прізвище)

Зав. кафедри – _____
(підпис)

зав. кафедри ІТС, кандидат
педагогічних наук, доцент,
М.А. Семенов
(посада, науковий ступінь, наукове звання,
ініціали, прізвище)

Міністерство освіти і науки України

Державний заклад

„Луганський національний університет імені Тараса Шевченка”

Навчально-науковий інститут математики та інформаційних технологій

Інформаційних технологій та систем

Бакалавр

121 Інженерія програмного забезпечення

(код, назва)

12, Інформаційні технології

(код, назва)

Завідувач кафедри ІТС

(підпис)

(ініціали, прізвище)

“ ” 2025 p.

НА БАКАЛАВРСЬКУ РОБОТУ

(прізвище, ім'я, по батькові)

- Керівник кваліфікаційної роботи Переяславська С.А., к.пед.н., доцент

(прізвище, ініціали, науковий ступінь, вчене звання)

Від

- 3. Вихідні дані до роботи:** Програмний модуль «Face Recognition System», створений для розпізнавання облич у режимі реального часу з використанням бібліотеки OpenCV на мові програмування Python.

- 4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):** поняття та особливості технологій комп'ютерного зору, огляд алгоритмів розпізнавання облич, реалізація додатку «Face Recognition System»

- 5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):** Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „_____” _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 24 жовтня	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	До 1 лютого	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 лютого	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 1 квітня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень квітня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 31 квітня	
7.	Попередній захист роботи на кафедрі	травень	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

Керівник проекту (роботи)

підпис

підпис

М.О.Вередін

(ініціали, прізвище)

С.О. Переяславська

(ініціали, прізвище)

АНОТАЦІЯ

Петуховський В.О.

Тема: Алгоритм розпізнавання облич у відеопотоці

Спеціальність: 121 «Інженерія програмного забезпечення».

Установа: ДЗ Луганський національний університет імені Тараса Шевченка, 2025 р.

Кваліфікаційна робота: 63 с., 19 рис., 4 табл., 20 джерел.

Мета роботи – розробити алгоритм, що дозволяє здійснювати розпізнавання облич у відеопотоці з використанням сучасних технологій комп'ютерного зору.

Об'єкт дослідження – процес розпізнавання облич у цифрових відеопотоках.

Предмет дослідження – методи та алгоритми комп'ютерного зору, що використовуються для розпізнавання облич.

Методи дослідження: аналіз літературних джерел, моделювання, розробка програмного забезпечення, тестування, порівняння ефективності алгоритмів.

Результати роботи: Розроблена система демонструє стабільну роботу в режимі реального часу та може бути використана як основа для подальшого розвитку в галузі безпеки, контролю доступу або інтерактивних застосунків.

Висновки : Кваліфікаційна робота на тему: «Алгоритм розпізнавання облич у відеопотоці» присвячена розробці системи для автоматичного розпізнавання облич в режимі реального часу. У роботі проведено аналіз існуючих методів виявлення та розпізнавання облич, обґрунтовано вибір технологій реалізації, розроблено архітектуру системи та реалізовано її у вигляді програмного продукту з використанням мови програмування Python та бібліотеки OpenCV.

Ключові слова: РОЗПІЗНАВАННЯ ОБЛИЧ, ВІДЕОПОТІК, КОМП'ЮТЕРНИЙ ЗІР, АЛГОРИТМИ, PYTHON, OPENCV.

ABSTRACT

Petukhovskiy V.O.

Topic: Face Recognition Algorithm in Video Stream

Specialty: 121 "Software Engineering".

Institution: DZ Luhansk Taras Shevchenko National University, 2024

Qualification work: 63 pages, 19 figures, 4 tables, 20 sources.

The purpose of the work: to develop an algorithm that enables face recognition in a video stream using modern computer vision technologies.

Object of research: the process of face recognition in digital video streams.

Subject of research: methods and algorithms of computer vision used for face recognition.

Research methods: literature review, modeling, software development, testing, and performance comparison of algorithms.

Results of the work: the developed system demonstrates stable performance in real-time mode and can be used as a basis for further development in the field of security, access control, or interactive applications.

Conclusion: the qualification thesis entitled "Face Recognition Algorithm in Video Stream" is dedicated to the development of a system for automatic face recognition in real-time.

The work includes analysis of existing methods of face detection and recognition, justification of chosen implementation technologies, system architecture development, and implementation of a software product using the Python programming language and OpenCV library.

The system was tested, and its accuracy, performance, and compliance with specified requirements were evaluated.

Keywords: FACE RECOGNITION, VIDEO STREAM, COMPUTER VISION, ALGORITHMS, PYTHON, OPENCV.

ДОДАТКОК В ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Державний заклад «Луганський національний університет
імені Тараса Шевченка»

Факультет (інститут)

Навчально-науковий інститут математики та
інформаційних технологій
(повна назва)

Кафедра

Інформаційних технологій та систем
(повна назва)

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання програмної розробки:

" РОЗРОБКА АЛГОРИТМУ РОЗПІЗНАВАННЯ ОБЛИЧ У
ВІДЕОПОТОЦІ "

ПОГОДЖЕНО

Керівник кваліфікаційної роботи

Переяславська С.О.

“ _____ ” 2025р

ВИКОНАВЕЦЬ

Студент групи 4 ІПЗ

Петуховський В.О.

“ _____ ” 2025р

Полтава - 2025

ЗМІСТ

1. Загальні відомості	3
2. Призначення та характеристика програмного додатку.....	3
3. Вимоги до системи.....	4
3.1 Функціональні вимоги:.....	4
3.2 Нефункціональні вимоги:.....	4
3.3 Вимоги до програмного забезпечення:	4
3.4 Технічні вимоги:.....	4
4. Порядок контролю і приймання роботи	5

1. Загальні відомості

Повна назва системи: Додаток реалізації алгоритм для виявлення облич у відеопотоці

Замовник: Кафедра інформаційних технологій та систем, ЛНУ імені Тараса Шевченка, спеціальності 121 "Інженерія програмного забезпечення".

Планові терміни виконання:

початок — жовтень 2024 р.,

завершення — червень 2025 р.

Фінансування: навчальний проєкт, фінансування не передбачено. Результати робіт оформлюються у вигляді дипломної роботи з додатком програмного коду та звіту.

2. Призначення та характеристика програмного додатку

Система призначена для автоматизованого виявлення облич у відеопотоці з використанням бібліотеки OpenCV.

Область застосування — Розроблена система може застосовуватись у навчальних цілях для демонстрації алгоритмів комп'ютерного зору та практичного вивчення OpenCV. Також її можна адаптувати для систем контролю доступу, відеоспостереження та базових біометричних рішень. Система придатна для фіксації облич у кадрі в реальному часі та зберігання невідомих осіб для подальшого аналізу. Метою є створення прототипу інформаційної системи, здатної в реальному часі визначати наявність облич, порівнювати їх з базою даних та фіксувати нові виявлені обличчя.

Об'єкт автоматизації — процес виявлення та реєстрації облич за допомогою вебкамери. Середовище експлуатації — настільні комп'ютери або ноутбуки з ОС Windows 10/11.

Необхідні умови — наявність підключеної вебкамери.

3. Вимоги до системи

3.1 Функціональні вимоги:

- Захоплення відеопотоку з вебкамери. Виявлення облич за допомогою алгоритму Віолі-Джонса (каскади Хаара).
- Порівняння виявлених облич із базою даних (графічні гістограми).
- Інтерфейс керування програмою (Tkinter).
- Збереження нових облич у базу даних за допомогою кнопок у інтерфейсі Tkinter(Вибрати файл\Перемістити файл).
- Вивід на інтерфейс імені виявлених облич.
- Створення лог-журналів для автоматичного запису подій, що відбуваються в комп'ютерній системі.

3.2 Нефункціональні вимоги:

- Система має працювати у реальному часі.
- Програмне забезпечення має бути сумісним з Windows 10/11.
- Продуктивність: детекція на стандартній камері — не рідше 50 кадрів на секунду.

3.3 Вимоги до програмного забезпечення:

- Мова програмування: Python 3.x.
- Зовнішні бібліотеки: OpenCV, Pillow, NumPy, Tkinter.

3.4 Технічні вимоги:

- Мінімальні системні вимоги: 4 ГБ ОЗУ, процесор 1.5 ГГц, вебкамера.

- Рекомендовані: 8 ГБ ОЗУ, процесор 2.5 ГГц+.

4. Порядок контролю і приймання роботи

Система вважається прийнятою, якщо:

- успішно розпізнає обличчя з камери в режимі реального часу;
- дозволяє зберігати нові зображення облич;
- не містить критичних помилок при стандартному використанні.

Приймання здійснюється замовником на основі демонстрації та захисту проєкту.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**ДЕРЖАВНИЙ ЗАКЛАД „ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА”**

Навчально-науковий інститут математики та інформаційних технологій

Кафедра інформаційних технологій та систем

Пояснювальна записка
до кваліфікаційної роботи
за першим (бакалаврським) рівнем освіти

Розробка алгоритму розпізнавання облич у відеопотоці

Виконав: студент 4 курсу
напряму підготовки (спеціальності)
121 «Інженерія програмного
забезпечення»
(шифр і назва напряму підготовки, спеціальності)

Петуховський В.О.
(прізвище та ініціали)

Керівник Переяславська С.О.
(прізвище та ініціали)

Рецензент Козуб Ю.Г.
(прізвище та ініціали)

Полтава – 2025 року

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КОМП'ЮТЕРНОГО ЗОРУ ТА РОЗПІЗНАВАННЯ ОБЛИЧ	5
1.1. Технологія комп'ютерного зору	5
1.2 Розпізнавання образів. Методи та алгоритми розпізнавання образів	7
1.3 Детектування облич у відеопотоці	10
1.4 Засоби програмування систем комп'ютерного зору	20
Висновки до розділу 1	25
РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗПІЗНАВАННЯ ОБЛИЧ У ВІДЕОПОТОЦІ	26
2.1. Характеристика програмного додатку, постановка завдання та визначення вимог	26
2.2 Архітектура та моделювання проєкта	27
2.3 Програмна реалізація додатку.	30
2.4 Тестування і демонстрація роботи алгоритму.	37
Висновки до розділу 2	46
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	49

ВСТУП

У сучасних інформаційних системах дедалі більшого значення набувають технології комп'ютерного зору, що дозволяють автоматизовано аналізувати візуальні дані та приймати рішення на їх основі. Одним із ключових напрямів застосування комп'ютерного зору є розпізнавання облич, яке використовується у системах безпеки, контролю доступу, медичних додатках, сервісах персоналізації тощо.

У процесі виконання проєкту було проведено аналіз сучасних алгоритмів розпізнавання облич, реалізовано прототип інформаційної системи, проведено тестування працездатності програми та сформульовано висновки щодо її ефективності.

Метою даної роботи є розробка алгоритму розпізнавання облич у відеопотоці в реальному часі.

Об'єкт дослідження: процес розпізнавання облич у відеопотоці в реальному часі.

Предмет дослідження: алгоритми та методи комп'ютерного зору, машинного навчання та обробки зображень, що застосовуються для виявлення та ідентифікації облич у відеопотоці в реальному часі.

Відповідно до предмета дослідження і мети, були виділені основні **завдання дослідження:**

- Провести аналіз сучасних методів та алгоритмів розпізнавання облич у відеопотоці.
- Провести аналіз і обґрунтувати вибір оптимального алгоритму для реалізації системи розпізнавання облич.
- Розробити архітектуру програмного забезпечення для автоматизованого розпізнавання облич у реальному часі.

- Реалізувати прототип інформаційної системи для захоплення відеопотоку, детекції та ідентифікації облич.
- Забезпечити функціонал збереження нових виявлених облич у базу даних для подальшого аналізу.
- Реалізувати механізм ведення журналу подій (логування) для моніторингу роботи системи.
- Провести тестування працездатності програмного додатку за різними сценаріями використання.

До складу роботи входять два розділи, які висвітлюють теоретичні й практичні питання:

- у **першому розділі** розглянуто теоретичні аспекти комп'ютерного зору, методи та алгоритми розпізнавання облич у відеопотоці, здійснено аналіз сучасних підходів та обґрунтовано вибір оптимального алгоритму для реалізації системи;
- у **другому розділі** представлено практичну реалізацію розробленого програмного додатку: описано архітектуру та функціональність системи, наведено результати тестування працездатності алгоритму та демонстрацію роботи додатку у реальному часі.

Практична цінність роботи полягає у створенні простої та гнучкої програмної системи, яка може бути використана для навчальних цілей, демонстрації роботи алгоритмів комп'ютерного зору, а також для подальшого розширення та інтеграції в більш складні системи.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ КОМП'ЮТЕРНОГО ЗОРУ ТА РОЗПІЗНАВАННЯ ОБЛИЧ

1.1. Технологія комп'ютерного зору

Актуальність комп'ютерного зору зростає з кожним роком, оскільки технології продовжують розвиватися, дозволяючи вирішувати все більше завдань із більшою точністю та ефективністю, що відкриває нові можливості для його застосування в майбутньому.

Комп'ютерний зір (Computer Vision, CV) — це галузь штучного інтелекту, що займається аналізом, обробкою та інтерпретацією зображень і відео. Ця технологія дозволяє комп'ютерам "бачити" та розуміти візуальну інформацію подібно до людського зору. [1], [2]

Основні задачі, де застосовується комп'ютерний зір

1. Обробка зображень та відео

- Робота з двовимірними (2D) та тривимірними (3D) даними.
- Підтримка різних форматів зображень (JPEG, PNG, BMP, TIFF тощо).
- Використання попередньої обробки зображень: нормалізація, шумозаглушення, фільтрація тощо.[15]

2. Глибоке навчання та нейронні мережі

- Використання згорткових нейронних мереж (CNN) для аналізу зображень.[8]
- Підтримка моделей, таких як ResNet, VGG, YOLO, Faster R-CNN.[16]
- Навчання на великих наборах даних (ImageNet, COCO, Open Images).

3. Розпізнавання об'єктів та класифікація

- Ідентифікація та категоризація об'єктів на зображеннях.
- Використання алгоритмів сегментації (Mask R-CNN, U-Net).
- Визначення меж і положення об'єктів у просторі.

4. Відстеження об'єктів

- Реалізація алгоритмів для відстеження руху (SORT, DeepSORT).
- Використання багатокадрового аналізу для розпізнавання траєкторій руху.

5. Розпізнавання осіб та біометрія

- Виявлення облич на зображеннях та відео.
- Ідентифікація осіб за допомогою моделей FaceNet, Dlib, OpenCV.
- Використання біометричних даних для доступу та безпеки.

6. Тривимірна реконструкція та глибина зображення

- Використання стереозору та LiDAR для побудови 3D-моделей.
- Визначення глибини сцени та відстані до об'єктів.

7. Оптимізація продуктивності

- Використання апаратного прискорення (GPU, TPU, FPGA).
- Підтримка фреймворків TensorFlow, PyTorch, OpenCV, OpenVINO.[17,18,20]
- Реалізація методів прискорення обчислень: квантування, прунінг, компресія моделей.

8. Автоматизація та інтеграція

- Інтеграція з IoT-пристроями, робототехнікою та автономними системами.
- Використання комп'ютерного зору в промисловості, медицині, автомобільній сфері.

Комп'ютерний зір продовжує розвиватися завдяки зростанню обчислювальних потужностей, вдосконаленню алгоритмів машинного навчання та збільшенню доступу до великих обсягів даних.

1.2 Розпізнавання образів. Методи та алгоритми розпізнавання образів

Основні поняття теорії образів. Класифікація систем розпізнавання образів. Прості і складні системи. Принципи класифікації. Системи з навчанням, без навчання, що самонавчаються. Алгоритми: структурні, ймовірнісні, логічні, неромережіві, детерміновані, гібридні.

Основи теорії образів у контексті розпізнавання облич

- *Образ* — зображення обличчя в певному кадрі.
- *Ознаки* — координати очей, форми рота, носа, текстура шкіри, вектори ознак з нейромереж.
- *Простір ознак* — багатовимірний простір, де кожне обличчя представлене числовим вектором.
- *Клас* — певна особа або категорія (наприклад: відома людина, незнайомиць).
- *Розпізнавання* — визначення, хто зображений на обличчі.

Класифікація систем розпізнавання у відеопотоці

За складністю

- *Прості системи:*
 - Виявляють обличчя за шаблонами (наприклад, алгоритм Хаара).
 - Працюють лише на статичних зображеннях або низькій частоті кадрів.
 - Обмежені в можливостях (не стежать за рухом, не розрізняють осіб).

- *Складні системи:*
 - Працюють у реальному часі.
 - Включають виявлення, трекінг, нормалізацію обличчя та ідентифікацію.
 - Використовують глибокі нейронні мережі (CNN, FaceNet, Dlib, MTCNN).
 - Інтеграція з базами даних, самонавчання, адаптація до змін освітлення, пози і віку.

У таблиці 1.1 наведено типи систем за принципами навчання.

Таблиця 1.1

Типи систем за принципами навчання

Тип системи	Застосування у відео
Без навчання	Виявлення облич за фіксованими шаблонами (Haar cascades, HOG + SVM).
З навчанням	Використовують попередньо натреновані моделі CNN для векторизації та класифікації осіб.
Самонавчання	Моделі оновлюють власні уявлення про нові обличчя в процесі використання (реідентифікація, кластеризація в реальному часі).

В таблиці 1.2 наведено алгоритми в системах розпізнавання облич.

Таблиця 1.2

Типи алгоритмів в системах розпізнавання облич

Тип алгоритму	Роль у відеопотоці	Приклад

Структурні	Виявлення форми обличчя, очей, рота (ключові точки)	Dlib, OpenCV shape predictor
Ймовірнісні	Оцінка ймовірності належності до особи	Байєсівські фільтри, Kalman-фільтр
Логічні	Прийняття рішень у системах доступу, охорони	"Якщо особа = X, тоді відкрити двері"
Неромережеві	Глибинна обробка облич: векторизація, класифікація	FaceNet, DeepFace, ArcFace
Детерміновані	Постійна логіка трекінгу або логічні правила	Постійний ідентифікатор обличчя в системі
Гібридні	Поєднання CNN + логіки або трекінг + класифікація	MTCNN + трекер + FaceNet

У таблиці 1.3 наведені етапи для типової архітектури системи розпізнавання облич у відео.

Таблиця 1.3

Етапи для типової архітектури систем розпізнавання облич

Етап	Опис / Прикладні технології
Отримання відеопотоку	Камера в реальному часі або відеофайл
Детекція облич	MTCNN, Haar cascades
Трекінг облич	SORT, Deep SORT, Kalman filter
Виділення ознак	CNN-екстрактори: FaceNet, VGGFace
Класифікація / Ідентифікація	Алгоритми: k-NN, SVM, softmax classifier
Реакція системи	Логіка дій: надання доступу, сигналізація, збереження даних тощо

1.3 Детектування облич у відеопотоці

Детектування облич у потоці (в режимі реального часу) є ключовою задачею комп'ютерного зору, яка використовується в системах відеоспостереження, розпізнавання осіб, безпеці та взаємодії людини з комп'ютером.

Детектування облич у потоці є основою для багатьох застосувань, таких як автоматизовані системи безпеки, відстеження емоцій, ідентифікація осіб, а також взаємодія людини з пристроями.

Етапи розпізнавання облич у відеопотоці (рис.1.1):

1. Ініціалізація відеопотоку

- Відкриття камери або відеофайлу.
- Підготовка ресурсів: моделі, класифікатори, бази облич.

2. Захоплення кадру

- Кожен кадр (frame) витягується з відеопотоку для обробки.

3. Попередня обробка

- Зменшення розміру кадру (для швидкості).
- Переведення в сірий колір (для Haar/LBP).
- Нормалізація освітлення (опційно).

4. Детекція облич

- Використання алгоритму (наприклад, Haar Cascade, MTCNN, SSD або YOLO) для знаходження координат облич у кадрі.

5. Вирізання та обробка облич

- Виділення регіонів із зображенням облич.
- Масштабування до стандартного розміру.
- Перетворення до формату, який очікує модель розпізнавання.

6. Розпізнавання (ідентифікація)

- Отримання ознак обличчя (embeddings) через нейронну мережу (наприклад, FaceNet, DeepFace).
- Порівняння з базою даних відомих облич.
- Присвоєння імені або мітки.

7. Відображення результатів

- Малювання рамок навколо облич.
- Вивід імен / ID на екрані.
- Можливе логування, збереження кадрів або подій.

8. Повторення для наступного кадру

- Перехід до наступного кадру і повторення циклу.

9. Завершення

- Закриття відеопотоку.
- Звільнення ресурсів.

Опис системи роботи алгоритмів

Для розпізнавання об'єктів на фото або відео необхідно мати натреновану нейронну мережу. У нашому випадку потрібно мати нейронну мережу, що натренована розпізнавати обличчя. Тренування нейронної мережі - довгий і складний процес. Полягає він у тому, щоб дати нейронці десятки тисяч фото і дати їй можливість самій знаходити на них обличчя. Ми, своєю чергою, щоразу в разі неправильного варіанту повинні говорити їй про це. Після кількох сотень тисяч спроб і помилок діапазон для знаходження правильної відповіді у нейронної мережі буде скориговано. Ваги для розпізнавання правильної відповіді буде налаштовано, і надалі під час її використання вона буде вірно знаходити потрібний об'єкт. Тренувати нейронну систему можна і вдома, але таке зробити не просто, а головне не потрібно. Річ у тім, що вже існують готові натреновані

моделі, які можна використовувати для своїх цілей. Саме таким чином ми і вчинимо.

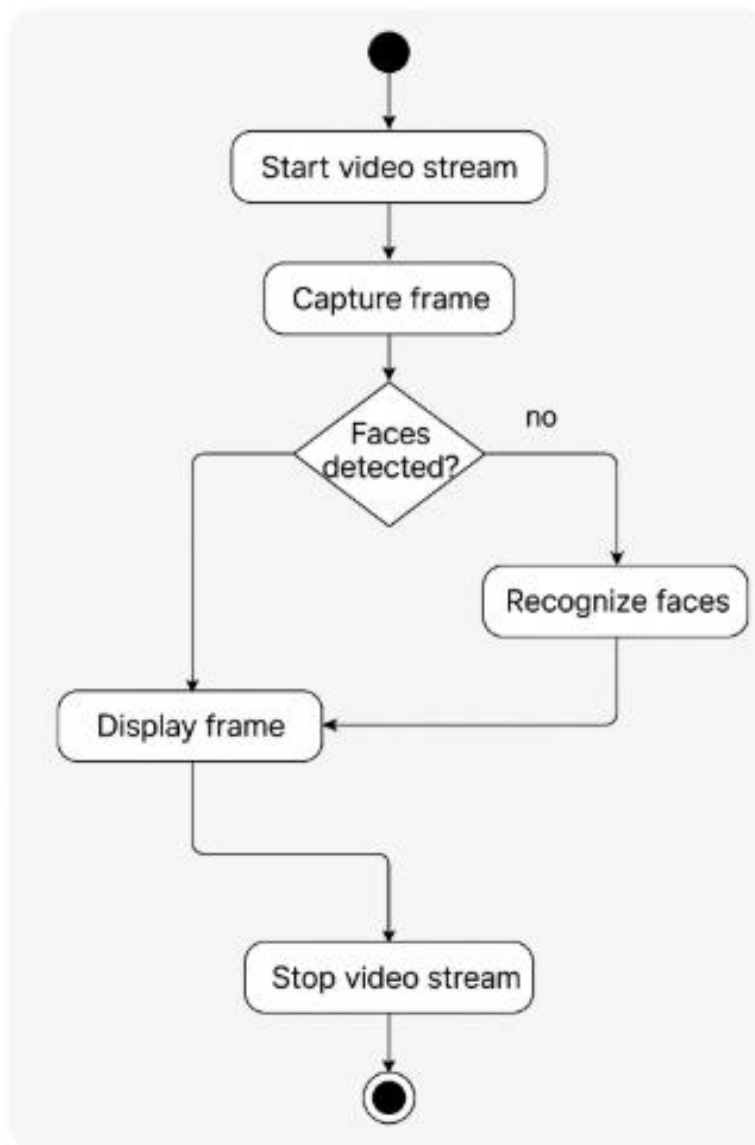


Рисунок 1.1 – Структура алгоритму розпізнавання облич

Прямокутниками ми позначаємо межі об'єктів під час розмітки датасетів для систем розпізнавання, підписуємо лейбли, щоб наочно показати, які об'єкти розташовані на зображенні, де саме і в якій кількості. Можна малювати найрізноманітніші геометричні фігури - не тільки квадрати - а також

зафарбовувати їх, змінювати товщину і колір ліній. Приклад виділення прямокутником наведено на (рис 1.2)

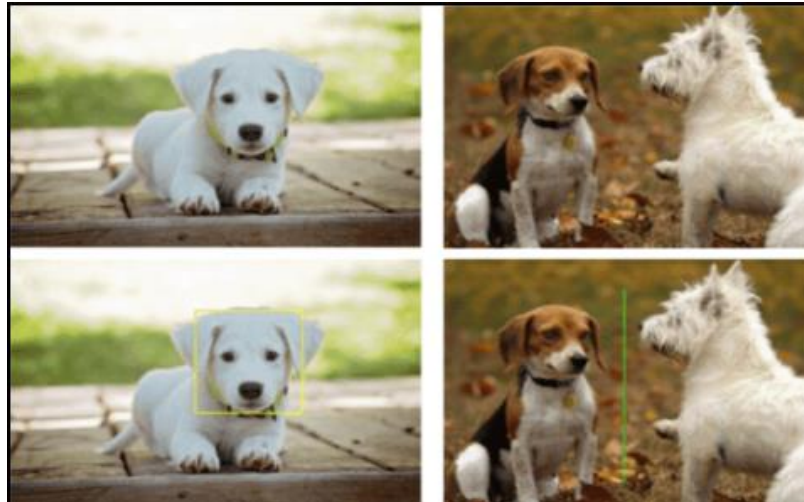


Рисунок 1.2 - Метод виділення прямокутником

У OpenCV є доволі потужний функціонал із детекту (визначення) об'єктів на зображенні. Реалізовано це завдяки попередньо навченим моделям, заснованим на нейромережах. У прикладі, який ми розглянемо нижче, використовувалася навчена модель визначення фронтального зображення обличчя на фото (допускається поворот голови до приблизно 30 градусів). OpenCV дає змогу дуже швидко знаходити обличчя. При цьому ми не ідентифікуємо особу людини, але з певною часткою впевненості можемо припустити, що обличчя знаходиться саме в цій частині кадру. Приклад виділення обличчя людини зображено на (рис 1.3)



Рисунок 1.3 – Виділення обличчя людини

Існуючі алгоритми

1. Haar Cascade Classifier - це ефективний метод виявлення об'єктів, який був запропонований Полом Віолою та Майклом Джонсом у їхній роботі “Rapid Object Detection using a Boosted Cascade of Simple Features” у 2001 році. Це підхід на основі машинного навчання, де каскадна функція навчається з великої кількості позитивних (зображення з обличчями) та негативних (зображення без облич) зображень. [6]

Ось основні моменти алгоритму Haar Cascade Classifier:

- Особливості Наар: Для виявлення об'єктів використовуються особливості Наар, які є простими формами, такими як прямокутники, що використовуються для визначення різниці в інтенсивності пікселів між чорними та білими областями.

- Інтегральне зображення: Для швидкого обчислення особливостей Наар використовується концепція інтегрального зображення, яка дозволяє обчислити суму значень пікселів у будь-якій прямокутній області за дуже короткий час.

- Adaboost: Для вибору найкращих особливостей з великої кількості можливих використовується алгоритм Adaboost. Він дозволяє класифікувати обличчя як позитивні та негативні зображення з мінімальною кількістю помилок.

- Каскадні класифікатори: Навчений класифікатор складається з каскаду етапів, де кожен етап використовує декілька особливостей для визначення, чи є об'єкт на зображенні. Це дозволяє швидко відкинути негативні зразки на ранніх етапах, що забезпечує високу швидкість виявлення.

Haar Cascade Classifier широко використовується для виявлення облич у реальному часі через його швидкість та ефективність, хоча він може бути схильний до хибнопозитивних виявлень і потребує налаштування параметрів при застосуванні. Цей метод все ще актуальний і корисний, особливо при роботі з обмеженими ресурсами, коли не можна використовувати більш обчислювально витратні детектори об'єктів.

2.MTCNN, або Multi-Task Cascaded Convolutional Networks, - це передовий алгоритм для виявлення облич та вирівнювання особливостей обличчя, який був розроблений Кайпенгом Чжаном, Жанпенгом Чжаном, Чжифенгом Лі та Ю Кяо у 2016 році. [9] Основною ідеєю MTCNN є використання каскадної структури з трьох етапів глибоких згорткових нейронних мереж, які прогнозують розташування облич та орієнтирів (таких як очі, ніс і рот) від грубого до точного.

Ось ключові особливості MTCNN:

- Каскадна структура: MTCNN складається з трьох стадій згорткових нейронних мереж, кожна з яких виконує певну задачу: виявлення облич (P-Net), вирівнювання орієнтирів (R-Net) та відшукування облич (O-Net).

- Багатозадачність: Кожна стадія виконує кілька завдань одночасно, таких як виявлення облич, вирівнювання орієнтирів та визначення ймовірності наявності обличчя.

- Висока точність: MTCNN показав високу точність на складних наборах даних, таких як Fddb та WIDER FACE для виявлення облич, а також AFLW для вирівнювання орієнтирів.

- Реальний час: Незважаючи на високу точність, MTCNN зберігає здатність працювати в реальному часі, що робить його придатним для застосувань, які вимагають швидкої обробки.

- Стратегія онлайн-вибору складних зразків: У процесі навчання MTCNN використовує нову стратегію онлайн-вибору складних зразків, яка автоматично покращує продуктивність без ручного вибору зразків.

MTCNN є одним з найпопулярніших методів для виявлення облич та вирівнювання орієнтирів, завдяки його точності та ефективності.

3.YOLO, що розшифровується як “You Only Look Once” (Ти дивишся лише один раз), - це алгоритм виявлення об’єктів у реальному часі, який був вперше представлений Джозефом Редмоном, Сантошем Діввала, Россом Гіршиком та Алі Фархаді у 2015 році. Цей алгоритм відомий своєю швидкістю та точністю, що робить його популярним вибором у застосуваннях комп’ютерного зору. [10]

Ось деякі ключові особливості YOLO:

- Швидкість: YOLO може обробляти зображення зі швидкістю 45 кадрів за секунду (FPS), що робить його дуже швидким і придатним для реального часу.

- Точність виявлення: YOLO має високу точність виявлення об’єктів з дуже низькою кількістю помилок на фоні.

- Загальність: YOLO добре узагальнює від природних зображень до мистецтва, показуючи високу продуктивність на різноманітних наборах даних.

- Об’єднана архітектура: Відмінною особливістю YOLO є те, що він використовує єдину згорткову нейронну мережу для прямого прогнозування обмежувальних рамок і класових ймовірностей безпосередньо з повних зображень.

·Регресійна задача: На відміну від попередніх робіт, які перетворювали класифікатори для виконання виявлення, YOLO розглядає виявлення об'єктів як задачу регресії для просторово відокремлених обмежувальних рамок і пов'язаних з ними класових ймовірностей.

YOLO продовжує розвиватися, і його нові версії (наприклад, YOLOv3) включають покращення, які збільшують точність і швидкість, зберігаючи при цьому здатність до роботи в реальному часі .

4.SSD, або Single Shot MultiBox Detector, - це алгоритм виявлення об'єктів у реальному часі, який може ідентифікувати об'єкти на зображеннях швидко та точно. Він був представлений Вей Лю та іншими у 2015 році і з тих пір став одним з найпопулярніших методів для виявлення об'єктів у різних застосуваннях. [11]

Ось деякі ключові особливості SSD:

·Єдина глибока нейронна мережа: SSD використовує одну глибоку нейронну мережу для прямого прогнозування обмежувальних рамок і класових ймовірностей безпосередньо з повних зображень.

·Дискретизація простору виводу: SSD дискретизує простір виводу обмежувальних рамок у набір за замовчуванням з різними співвідношеннями сторін і масштабами на кожному місці карти ознак.

· Піраміда ознак: Алгоритм використовує метод виявлення піраміди ознак, що дозволяє йому виявляти об'єкти різних масштабів.

· Простота навчання: Модель SSD простіша відносно методів, які вимагають пропозицій об'єктів, оскільки вона повністю відмовляється від етапу генерації пропозицій і включає всі обчислення в одній мережі.

· Швидкість та точність: SSD показує порівнянну точність з методами, які використовують додатковий етап пропозиції об'єктів, і є набагато швидшим, надаючи єдиний фреймворк для навчання та виводу. SSD широко використовується у різних застосуваннях, таких як спостереження, сільське

господарство та медична візуалізація, завдяки його здатності швидко та точно ідентифікувати об'єкти.

Таблиця 1.4

Порівняльний аналіз алгоритмів розпізнавання обличчя

Характеристика	Віола-Джонса	MTCNN	YOLO	SSD
Принцип роботи	Каскад слабких класифікаторів (Haar-фічі та AdaBoost)	Каскадна CNN для локалізації облич і точок (landmarks)	Єдина CNN, що ділить зображення на сітку й передбачає об'єкти в один прохід	CNN із декількома виходами для різних масштабів об'єктів
Призначення	Переважно лише детекція облич	Детекція облич + визначення ключових точок	Загальне виявлення об'єктів (обличчя — частковий випадок)	Загальне виявлення об'єктів
Швидкість	Дуже швидкий для невеликих зображень	Повільніший за Віолу-Джонса, але досить швидкий	Дуже швидкий (особливо в оптимізованих версіях YOLOv4, YOLOv5)	Теж дуже швидкий, хоча трішки повільніший за YOLO
Точність	Низька за сучасними мірками, особливо на складних даних	Висока для облич у різних положеннях та освітленні	Висока, але потребує якісного навчання під конкретну задачу	Висока, особливо для середніх об'єктів
Переваги	Легкий, не потребує GPU, чудовий для простих застосувань	Одночасна локалізація облич і ключових точок	Реальний час на відео, підходить для великих наборів даних	Баланс між швидкістю та точністю

Характеристика	Віола-Джонса	MTCNN	YOLO	SSD
Недоліки	Погано працює зі складними поворотами, масштабами	Потребує більше обчислень, ніж Viola-Jones	Може погано локалізувати дрібні обличчя	Складніший для налаштування та оптимізації
Чутливість до освітлення, повороту	Висока чутливість	Стійкий до освітлення та повороту	Може втратити точність при великій варіативності	Краще справляється з різними умовами, ніж YOLO
Потреби в ресурсах	Низькі	Помірні (потрібен GPU для найкращої роботи)	Від GPU залежить точність і швидкість	Переважно потрібен GPU

Проведений аналіз дозволив встановити:

- **Viola-Jones** — застарілий, але простий та швидкий для базових задач.
- **MTCNN** — дуже добрий для розпізнавання облич і визначення ключових точок, гарний баланс швидкості та точності.
- **YOLO** — краще для швидкої детекції багатьох типів об'єктів; можна адаптувати під обличчя, особливо якщо потрібна реальна робота в режимі відео.
- **SSD** — альтернатива YOLO з трохи кращою обробкою дрібних об'єктів, гарний вибір, якщо потрібен компроміс між точністю і швидкістю.[13], [14]

Для вирішення поставлених задач з урахування вимог будемо використовуватись алгоритм Haar Cascade Classifier.

1.4 Засоби програмування систем комп'ютерного зору

Під час роботи потрібно зробити програму для детекції облич у відеопотоці, мову програмування для втілення роботи обрали Python [3], [4], [5]. Основними бібліотеками будуть OpenCV і os, за допомогою OpenCV буде виконано основну частину програми, що здатна виявляти обличчя та їхні частини у відеопотоці, користувачеві нададуть вибір між веб-камерою та заздалегідь записаним відео, основою алгоритму для виконання програми буде алгоритм Віюлі-Джонса, що ґрунтується на застосуванні каскадів Хаара. Для комфортнішої інтеграції алгоритму знадобиться встановлення пресетів навчання каскаду для уникнення тривалого і витратного процесу машинного нейромережевого навчання.

Бібліотека os слугуватиме підпорою для введення додаткової функції, як-от збереження або "запам'ятовування" виявлених облич, потрібно створити директорію і задати її розташування в коді, або створити її шляхом введення функції. Збереження здійснюватиметься шляхом спрацьовування функції в момент, коли основна програмна частина з функцією детекції облич та їхніх частин виявить усі елементи на відео, результат функції збереження буде надіслано в заздалегідь створену алгоритмом директорію у вигляді png-зображення.(рис. 1.4)

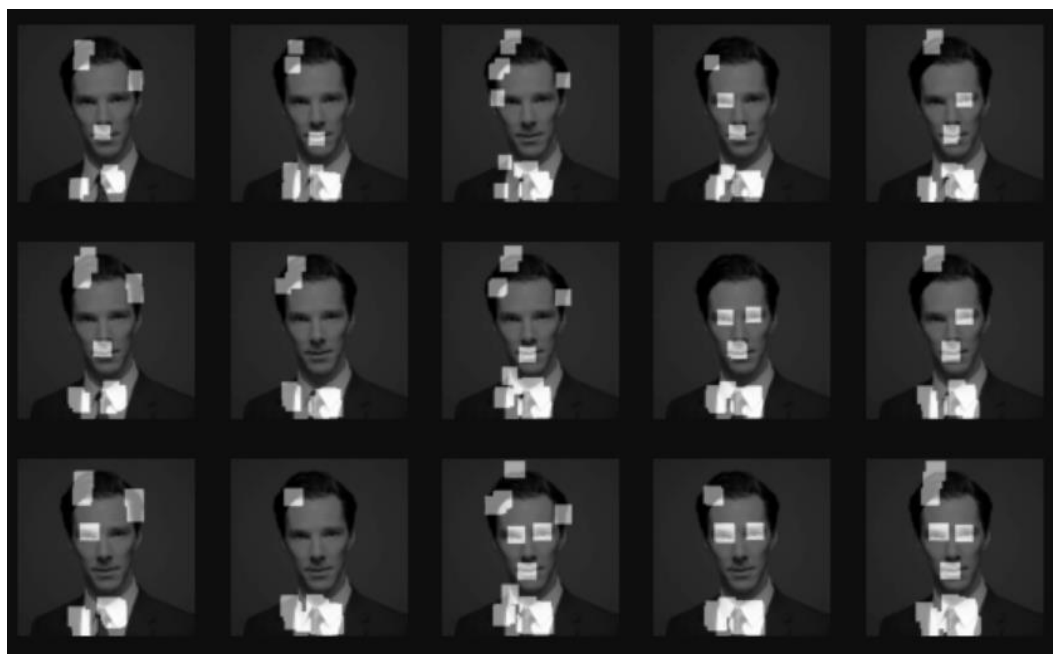


Рисунок 1.4 - Приклад візуалізації каскадів Хаара

Загальна інформація про бібліотеку OpenCV

OpenCV – це комп'ютерний зір з відкритим вихідним кодом і програмна бібліотека машинного навчання. OpenCV створено для забезпечення загальної інфраструктури додатків комп'ютерного зору й прискорення використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом, OpenCV спрощує використання і модифікацію коду.

Основні відомості про характеристики бібліотеки розміщено на сайті <http://opencv.org>., де можна завантажити бібліотеку для різних платформ, знайти повну документацію стосовно бібліотеки, короткі інструкції щодо встановлення (Quick Start), уроки з її використання (Tutorials), довідкову карту (Cheat Sheet), а також посилання на корисні сайти й т. ін. [7]

На сайті <http://docs.opencv.org> зібрано всю документацію, зокрема, детальні описи всіх функцій з повнотекстовим пошуком. На сайті <http://answers.opencv.org> можна ставити запитання, що стосуються бібліотеки, та отримувати на них

відповіді від розробників або інших користувачів. У бібліотеці зібрано більше 2500 оптимізованих алгоритмів, що являють собою повний набір класичних і сучасних алгоритмів комп'ютерного зору й машинного навчання. Ці алгоритми можна використовувати для виявлення й розпізнавання осіб, ідентифікації об'єктів, класифікації дій людини у відео, відслідковування руху камери й рухомих об'єктів, створення 3D-моделей сцени при роботі зі стереокамерами, зшивання зображень разом для одержання високої роздільності.

Бібліотека OpenCV має інтерфейси C ++, C, Python, Java і Matlab, підтримує ОС Windows, Linux, Android і Mac OS та орієнтується в основному на додатки в режимі реального часу. Сьогодні активно розвиваються повнофункціональні інтерфейси CUDA і OpenCL. Існує більше 500 алгоритмів і 10-кратна кількість функцій, які складають або підтримують ці алгоритми. Бібліотека OpenCV написана на мові C ++, має шаблонний інтерфейс.

На першому етапі бібліотека розроблялася як набір великих універсальних модулів (схcore, Cvaux, Highgui, Cvaux).

Проте у версії 2.2 структуру бібліотеки було реорганізовано – тепер великі модулі бібліотеки OpenCV замінено на менші за функціональним принципом:

opencv_core – ядро: базові структури, обчислення (математичні функції, генерація псевдовипадкових чисел, DFT, DCT, введення/виведення в XML і т. ін.);

opencv_imgproc – оброблення зображень (фільтри, перетворення й т. ін.);

opencv_highgui – простий UI (завантаження/зберігання зображень і відео);

opencv_ml – методи й моделі машинного навчання (SVM, для прийняття рішень і т. д.);

opencv_features2d – різні дескриптори (SURF);

opencv_video – аналіз руху й відслідковування об'єктів (оптичний потік, шаблони руху, видалення фону);

opencv_objdetect – детектування об'єктів на зображенні (вейвлети Хаару, HOG і т. д.);

opencv_calib3d – калібрування камери, пошук стереовідповідності й елементів оброблення тривимірних даних;

opencv_flann – бібліотека швидкого пошуку найближчих сусідів (FLANN);

opencv_contrib – супутній код, ще не готовий для застосування;

opencv_legacy – попередній код, збережений заради зворотної сумісності;

opencv_gpu – прискорення деяких функцій OpenCV з допомогою CUDA (NVidia).

Для початку роботи з бібліотекою OpenCV зазвичай рекомендують використовувати її в'язку з мовою Python (<http://python.org>). Це створює багато зручностей при реалізації проєктів. Програму на цій мові можна писати в будь-якому текстовому редакторі. Далі без попередньої компіляції вона виконується інтерпретатором Python. Мова Python з бібліотекою Numpy перетворюється на безкоштовний аналог системи Matlab, при цьому сама мова є розвиненішою, ніж Matlab. Ще одна перевага мови Python полягає в її хорошій документованості. В інтерактивній сесії Python (у командному вікні) можна швидко отримати список доступних функцій і методів конкретного класу, довідку про кожну функцію OpenCV, наприклад:

```
>>> import cv2 as cv
>>> dir(cv) # Список функцій і класів у cv
>>> cv.resize.__doc__ # Опис функції resize
>>> cap = cv.VideoCapture(0)
>>> dir(cap) # Список методів VideoCapture
>>> cap.read.__doc__ # Опис методу read
```

Застосування OpenCV та мови Python. Для початку роботи у віртуальному середовищі PyCharm і Python 3, слід установити необхідні бібліотеки для роботи зі всіма модулями, що цікавлять нас, і бібліотеками для вирішення конкретних завдань.

Модулі, які необхідно підключити під час установлення бібліотек:

1. NumPy – фундаментальний пакет для наукових обчислень з Python;
2. Pillow (Python Imaging Library) – основна бібліотека Python для роботи із зображеннями;
3. OpenCV – алгоритми комп'ютерного зору з відкритим вихідним кодом і програмна бібліотека машинного навчання.
4. Cv2 – набір додаткових функцій які розширюють потенціал OpenCV

Деякі особливості використання бібліотеки OpenCV у взаємодії з мовою програмування Python на базі віртуального середовища з операційною системою Windows, що рекомендується виробником. Сьогодні така композиція апаратно-програмних засобів дає змогу оптимально вирішувати безліч завдань машинного зору. В'язка OpenCV + Python найбільш продуктивною є при вирішенні завдань оброблення відеоданих у реальному часі на рухомих носіях (роботах і безпілотних ЛА).

Висновки до розділу 1

В цьому розділі нами було досліджено сутність поняття комп'ютерного зору, особливості алгоритмів для детектування облич та проведено їх аналіз. Під час дослідження було визначено етапи розпізнавання облич у відеопотоці (ініціалізація відеопотоку, захоплення кадру, попередня обробка, детекція облич, вирізання та обробка облич, розпізнавання (ідентифікація), відображення результатів). Проаналізовано алгоритми розпізнавання образів (Віола-Джонса, MTCNN, YOLO, SSD).

На підставі аналізу було встановлено особливості кожного алгоритму та вирішено застосовувати алгоритм Віоли-Джонса, що базується на застосуванні каскадів Хаара. Розглянуті інструменти програмування застосувань комп'ютерного зору. Вирішено застосовувати мову програмування Python з бібліотеками OpenCV, а середовищем розробки вибрали PyCharm.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗПІЗНАВАННЯ ОБЛИЧ У ВІДЕОПОТОЦІ

2.1. Характеристика програмного додатку, постановка завдання та визначення вимог

Розроблений програмний додаток є системою детекції облич у відеопотоці з використанням бібліотеки OpenCV. Основним завданням додатку є автоматизоване виявлення облич в режимі реального часу, порівняння їх з наявною базою даних та збереження нових облич для подальшого аналізу.

Програмний продукт реалізовано на мові Python з використанням бібліотек OpenCV, Pillow, NumPy та бібліотеки для створення інтерфейсу Tkinter. Архітектура програми побудована за моделлю Model-View-Controller (MVC), що забезпечує модульність, зручність у тестуванні та розширенні функціональності. Система має простий графічний інтерфейс, який дозволяє керувати основними діями — запуском камери, збереженням нових облич, а також роботою з базою даних.

Застосування даного додатку можливе у сфері освіти (для демонстрації роботи комп'ютерного зору), в системах контролю доступу, відеоспостереження та для реалізації простих біометричних систем.

До основних вимог до додатку належать:

- сумісність із Windows 10/11;
- можливість обробки відео з вебкамери в реальному часі;
- забезпечення збереження нових виявлених облич для

подальшого використання.

Більш детальна інформація щодо технічних характеристик, функціональних та нефункціональних вимог міститься в документі «Додаток В. Технічне завдання».

2.2 Архітектура та моделювання проєкта

Програмний додаток розробляється як система для детекції облич у відеопотоці в реальному часі. Основна мета — створити зручну та наочну систему, яка дозволяє виявляти обличчя з відео, ідентифікувати їх або фіксувати як нові. Структура проєкту додатку подано на рис.2.1

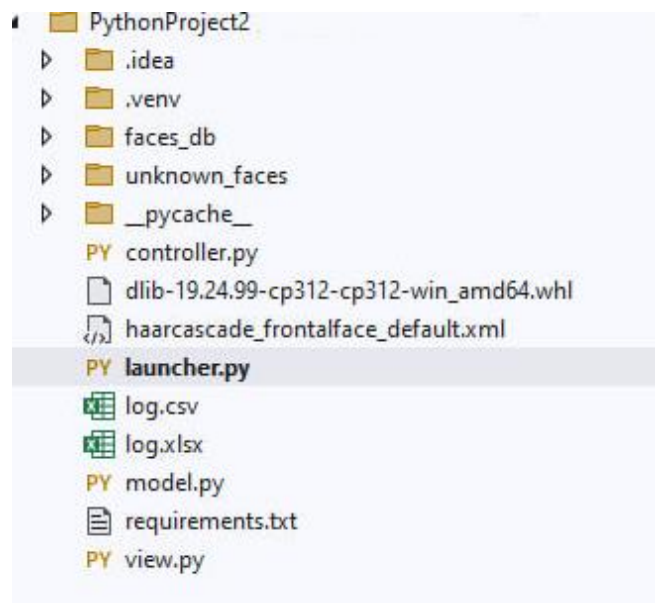


Рисунок 2.1 – Структура проєкту додатку

Для побудови системи використовується архітектурна модель **Model-View-Controller (MVC)**, що дозволяє чітко розділити функціональну логіку на:

- **модель** (обробка даних, порівняння, збереження),
- **представлення** (інтерфейс користувача),

- **контролер** (взаємодія між компонентами).

В даному проєкті блок View реалізовано у файлі view.py, що реалізує інтерфейс проєкту за допомогою бібліотеки Tkinter. У блоці Model (model.py) відбувається обробка даних, що застосовуються в проєкті: завантаження, порівняння та логування інформації в системі розпізнавання облич. Контролер controller.py реалізує логіку розпізнавання, керування камерою, зв'язок між моделлю та інтерфейсом.

Програма повинна працювати так:

1. Користувач запускає застосунок.
2. Відкривається вікно з графічним інтерфейсом (GUI).
3. Програма вмикає вебкамеру та виводить відео у реальному часі.
4. У кожному кадрі система шукає обличчя.
5. Якщо обличчя знайдено — система порівнює його з тими, що вже збережені у базі.
6. Якщо обличчя нове — зберігається у базу для подальшої ідентифікації.
7. Вся взаємодія виконується через прості кнопки та повідомлення інтерфейсу.

Загальна структура системи може бути описана такими основними блоками (рис.2.2):

- **Камера / джерело відео** — надає кадри.
- **Модуль обробки зображень** — знаходить обличчя.
- **Модуль розпізнавання / класифікації** — порівнює з базою.
- **База даних облич** — набір збережених зображень/векторів.
- **Інтерфейс користувача** — показує відео, результати, керування.



Рисунок 2.2 – Загальна UML-діаграма структури системи, побудована за логікою основних блоків

Для кращого розуміння структури програмного додатку на рисунку нижче наведено узагальнену UML-діаграму компонентів системи, яка ілюструє основні елементи та зв'язки між ними (Рис.2.3).

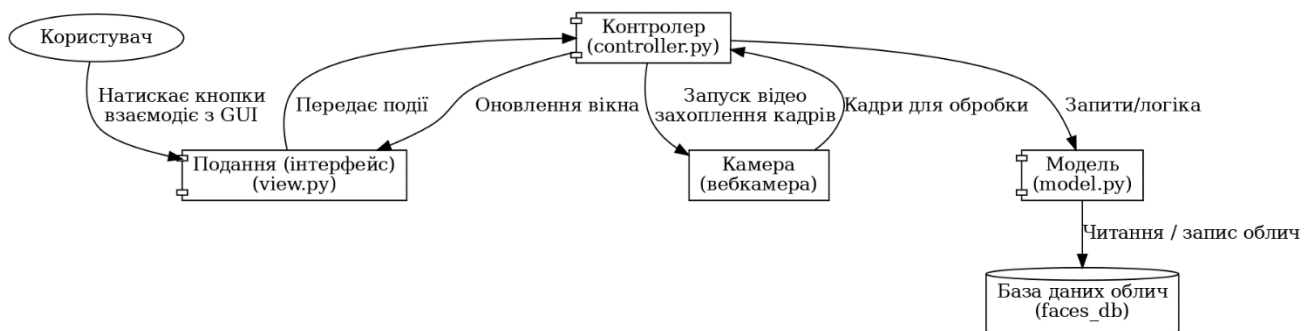


Рисунок 2.3 – UML-діаграма компонентів системи

Ця схема демонструє взаємодію між користувачем, інтерфейсом програми, модулем керування (контролером), логічною моделлю та зовнішніми пристроями (вебкамерою і базою даних облич).

Користувач керує додатком через графічний інтерфейс (View), який передає події контролеру. Контролер обробляє запити, активує камеру для отримання відео, виконує логіку розпізнавання через модель (Model) і взаємодіє з базою даних для збереження чи порівняння облич. Отримані результати повертаються через інтерфейс користувачу.

2.3 Програмна реалізація додатку.

Програмний додаток реалізовано мовою Python із застосуванням бібліотек OpenCV, Tkinter, Pillow та NumPy. Архітектура побудована за шаблоном **Model-View-Controller (MVC)**, що дозволяє розділити логіку програми на окремі компоненти.

До складу програмного забезпечення входять основні модулі:

- **controller.py** — головний модуль, що керує усіма процесами: отримує кадри з камери, виконує детекцію облич, ініціює взаємодію з моделлю та оновлює інтерфейс.
- **model.py** — відповідає за обробку даних: обчислення гістограм, порівняння облич, логування подій, роботу з базою зображень.
- **view.py** — реалізує графічний інтерфейс користувача, побудований на Tkinter, дозволяє керувати додатком за допомогою кнопок і виводити зображення з відеопотоку.
- **launcher.py** — містить клас `class PythonLauncher`, за допомогою якого реалізується запуск програми на виконання.

Користувач взаємодіє з інтерфейсом (view.py), звідки команди передаються у контролер (controller.py). Контролер захоплює відео з камери, здійснює обробку кожного кадру, виконує детекцію облич та порівнює їх з наявними даними за допомогою моделі (model.py). У разі виявлення нового обличчя програма пропонує зберегти його до бази.

Архітектура та моделювання проекту

На рисунку 2.4 представлено UML-діаграму класів розробленої системи

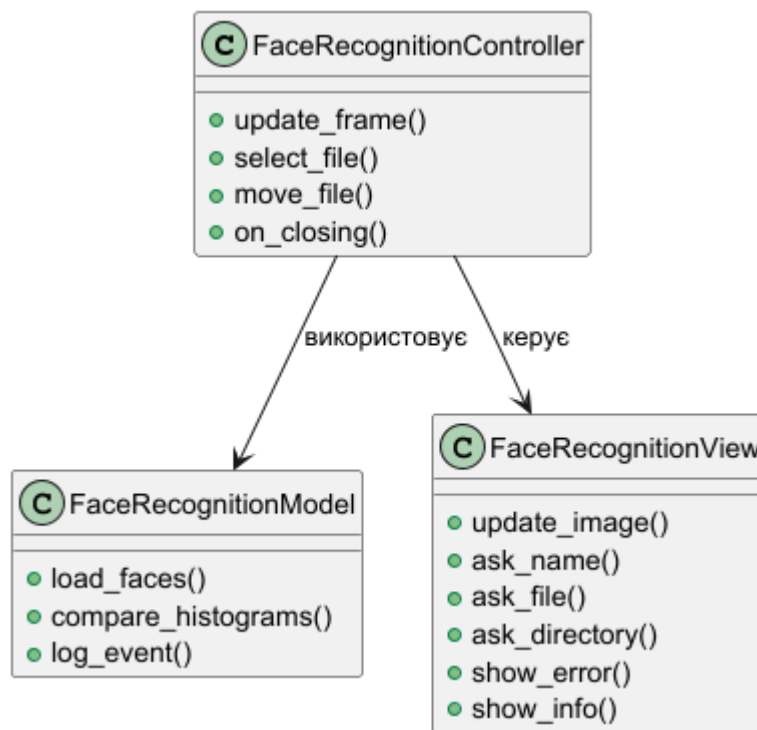


Рисунок 2.4 — UML-діаграма класів системи

Опис класів і методів

Клас **FaceRecognitionController** (controller.py)

Цей клас є основним контролером програми. Він відповідає за взаємодію між інтерфейсом користувача та логікою обробки даних.

Основні методи:

- `update_frame()` — здійснює обробку відеопотоку, детекцію облич та оновлення GUI.
- `select_file()` — дозволяє вибрати файл для переміщення.
- `move_file()` — переміщує вибране зображення до бази облич.
- `on_closing()` — коректно завершує роботу програми.

Клас `FaceRecognitionModel` (`model.py`)

Цей клас відповідає за збереження та обробку даних — бази облич, обчислення гістограм та логування подій.

Основні методи:

- `load_faces()` — завантажує відомі обличчя з бази.
- `compare_histograms()` — порівнює гістограми облич.
- `log_event()` — фіксує події у лог-файл.

Клас `FaceRecognitionView` (`view.py`)

Цей клас реалізує графічний інтерфейс користувача за допомогою бібліотеки Tkinter.

Основні методи:

- `update_image()` — оновлює зображення у вікні.
- `ask_name()` — запитує ім'я для нового обличчя.
- `ask_file()` — дозволяє вибрати файл.
- `ask_directory()` — дозволяє вибрати директорію.
- `show_error()` — показ повідомлення про помилку.
- `show_info()` — показ інформаційного повідомлення.

У програмі також реалізовано можливість переміщення файлів та логування подій у текстовий файл або таблицю Excel.

Детекція облич у відеопотоці

У файлі controller.py реалізовано захоплення кадрів з камери та виявлення облич за допомогою каскадів Хаара (Haar Cascades):

```
gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = cv2.CascadeClassifier(cv2.data.harcascades +
    "haarcascade_frontalface_default.xml") \
    .detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5)
```

Спочатку кадр перетворюється в відтінки сірого (для підвищення швидкодії), після чого виконується детекція облич. [12]

Порівняння з наявною базою

Після виявлення обличчя, створюється його гістограма, яка порівнюється з уже збереженими в базі:

```
hist = cv2.calcHist([face_crop], [0], None, [256], [0, 256])
hist = cv2.normalize(hist, hist).flatten()

for i, known_hist in enumerate(self.known_histograms):
    corr = self.model.compare_histograms(hist, known_hist)
    if corr > max_corr:
        max_corr = corr
        name = self.known_names[i]
```

Порівняння виконується за коефіцієнтом кореляції. Якщо обличчя не знайдено — викликається механізм додавання.

Збереження нового обличчя

Якщо обличчя нове, система запитує ім'я користувача і зберігає його у відповідну папку з датою:

```
save_path = os.path.join(date_folder, f"{name}_{timestamp}.jpg")
cv2.imwrite(save_path, frame[y:y + h, x:x + w])
```

Зображення обрізаного обличчя зберігається як JPEG-файл з іменем та міткою часу.

Оновлення інтерфейсу (GUI)

Інтерфейс користувача оновлюється в реальному часі — вікно виводить поточне зображення з розпізнаним іменем:

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
image = Image.fromarray(frame_rgb)
self.view.update_image(image)
self.view.root.after(50, self.update_frame)
```

Кадр перетворюється для правильного кольору і відображається в графічному вікні (через PIL.Image)

Логування подій

Кожна дія користувача або зміна в базі облич фіксується у лог-файл (CSV або Excel)[19]. Це реалізовано у модулі **model.py**, де зберігається ім'я, дата та тип події:

```
def log_event(self, name, status):
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(self.log_file, mode='a', newline="", encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow([now, name, status])
```

Це дозволяє вести журнал змін: коли було додано нове обличчя, ким, і з яким ім'ям.

Робота з базою зображень (faces_db)

База даних обличч — це локальна директорія з підпапками, що відповідають іменам користувачів. Кожна підпапка містить зображення обличч цієї особи. Під час запуску програма зчитує їх для побудови гістограм:

```
def load_faces(self):
    known_faces, face_names = [], []
    for filename in os.listdir(self.faces_dir):
        if filename.lower().endswith(('.jpg', '.png')):
            path = os.path.join(self.faces_dir, filename)
            image = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
            hist = cv2.calcHist([image], [0], None, [256], [0, 256])
            hist = cv2.normalize(hist, hist).flatten()
            known_faces.append(hist)
            face_names.append(os.path.splitext(filename)[0])
    return known_faces, face_names
```

Так система «запам'ятовує» користувачів для подальшого розпізнавання.

Переміщення обраного файлу

Через графічний інтерфейс користувач може обрати файл (наприклад, фото незнайомця) і перемістити його у певну директорію:

```
def move_file(self):
    if not self.selected_file:
        self.view.show_error("Спочатку виберіть файл!")
        return
    destination_folder = self.view.ask_directory()
    if not destination_folder:
        return
    try:
        destination_path = os.path.join(destination_folder,
os.path.basename(self.selected_file))
        shutil.move(self.selected_file, destination_path)
        self.view.show_info(f"Файл переміщено в {destination_path}")
        self.known_histograms, self.known_names = self.model.load_faces()
        self.view.label_selected.config(text="Файл ще не вибраний")
    except Exception as e:
        self.view.show_error(f"Помилка: {e}")
```

Після переміщення база облич автоматично оновлюється.

Запуск програми через лаунчер

Щоб зробити запуск зручнішим, реалізовано окремий скрипт *launcher.py*, який запускає головну програму:

```
def start_program(self):  
    if self.process is None or self.process.poll() is not None:  
        try:  
            self.process = subprocess.Popen([sys.executable, 'controller.py'])  
        except Exception as e:  
            messagebox.showerror("Помилка запуску", str(e))  
    else:  
        messagebox.showinfo("Інфо", "Програма вже запущена.")
```

Так можна відкрити додаток простим натисканням кнопки, не запускаючи консоль.

Ці додаткові елементи підсилюють функціональність системи та роблять її зручною у використанні, навіть для недосвідченого користувача.

2.4 Тестування і демонстрація роботи алгоритму.

Тестування програмного додатку має на меті перевірку коректності роботи алгоритму детекції облич у відеопотоці, стабільності системи та її відповідності поставленим вимогам. У процесі тестування було змодельовано різні сценарії користувацької взаємодії з програмою для виявлення потенційних недоліків і оцінки точності.

Тестування проводилось у таких умовах:

- Операційна система: Windows 11 64-bit
- Процесор: Intel Core i5
- Оперативна пам'ять: 8 ГБ
- Середовище розробки: PyCharm 2023
- Python: версія 3.12

- Вебкамера: стандартна HD-камера ноутбука
- Необхідні бібліотеки: OpenCV, Pillow, NumPy, Tkinter

Таблиця 2.1.

Результати тестування наведено в таблиці:

Сценарій	Очікуваний результат	Фактичний результат	Статус (Виконано /Не виконано/Частково виконано)
Детекція одного обличчя	Обличчя розпізнається, відображається ім'я	Обличчя розпізнається, відображається ім'я	Виконано
Виявлення нового обличчя	Програма пропонує ввести ім'я та зберігає обличчя	Програма пропонує ввести ім'я та зберігає обличчя	Виконано
Кілька облич у кадрі	Розпізнається хоча б одне обличчя	Розпізнаються всі обличчя	Виконано
Обличчя під нахилом	Обличчя частково або повністю розпізнається	Під невеликим кутом обличчя добре розпізнається	Виконано
Відсутність обличчя	Нічого не відображається, помилок немає	Нічого не відображається, помилок немає	Виконано

У ході тестування було зроблено скріншоти, які підтверджують працездатність системи. Вони демонструють розпізнавання облич, збереження нових користувачів та взаємодію з графічним інтерфейсом.

Демонстрація роботи алгоритму

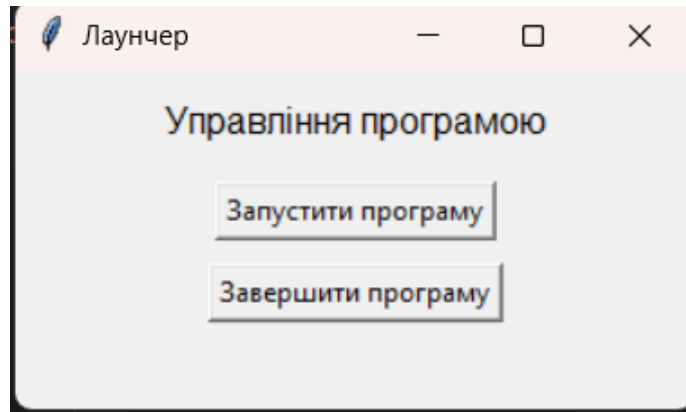


Рисунок 2.5 – Вікно запуску програми

Вікно запуску програми з лаунчером, що дозволяє запустити або завершити роботу.

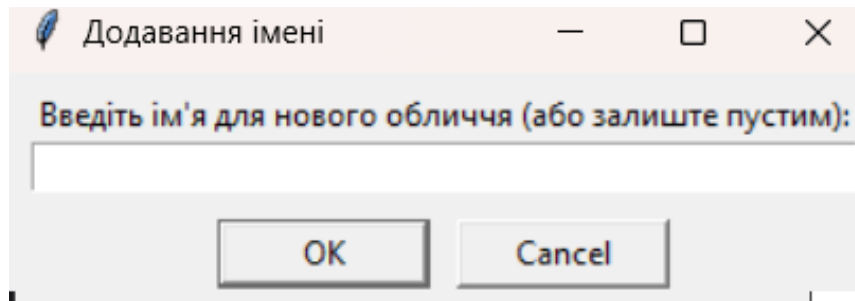


Рисунок 2.6 – Діалог при додаванні нового обличчя

Діалогове вікно, яке з'являється при виявленні нового обличчя. Програма запитує ім'я для збереження.

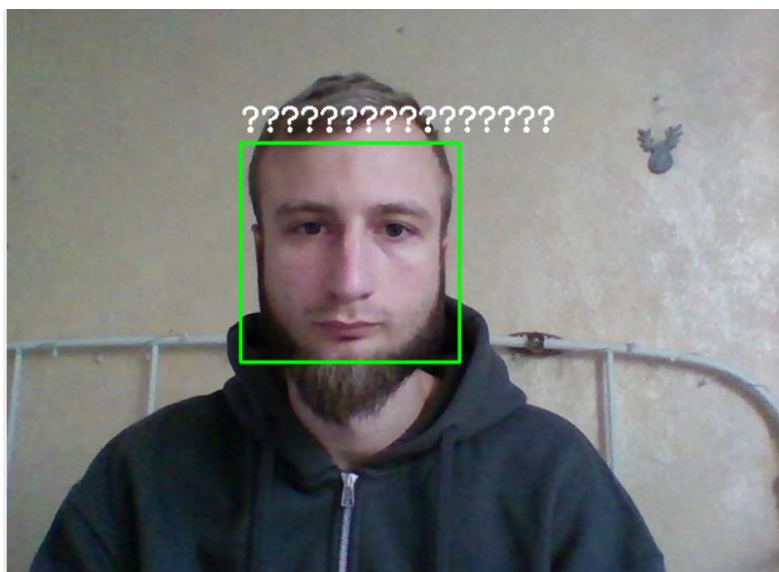


Рисунок 2.7 – Початок роботи програми: нове обличчя ще не додано
Кадр з відеопотоку, де обличчя ще не розпізнане. Його буде збережено як нове.

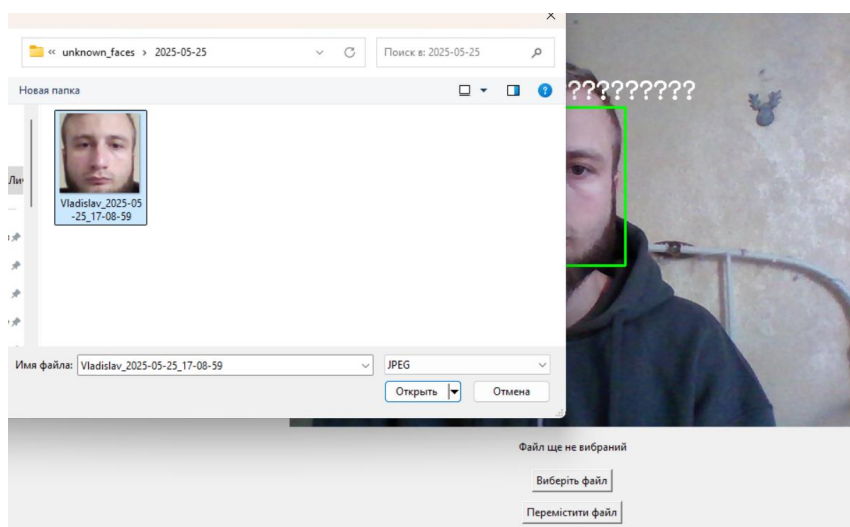


Рисунок 2.8 – Вибір обличчя для додавання з папки
unknown_faces

Користувач обирає нове обличчя з директорії unknown_faces для подальшого переміщення.

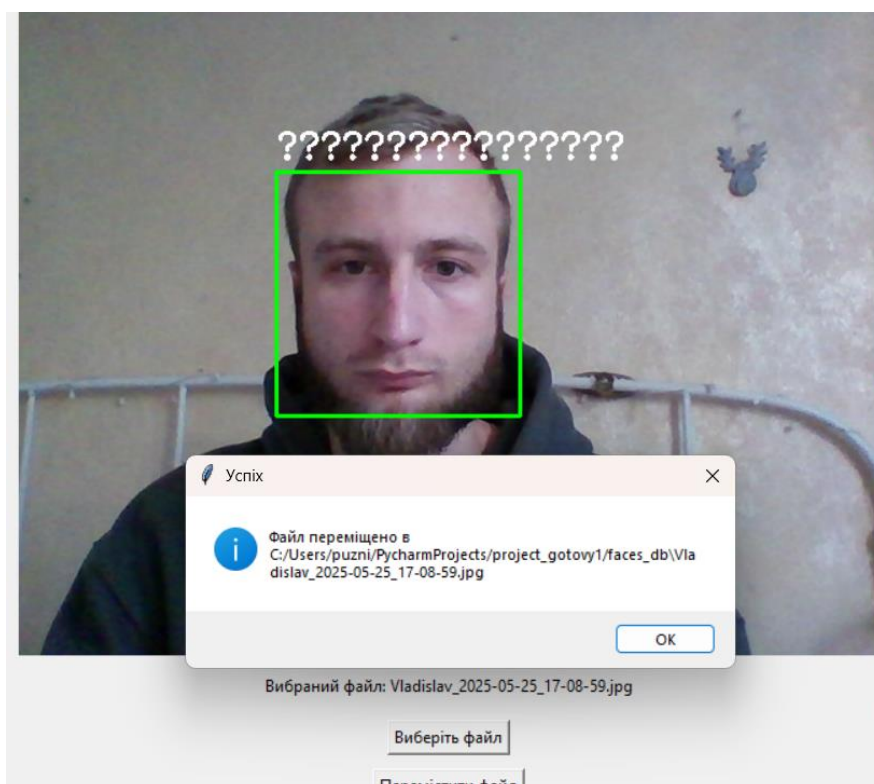


Рисунок 2.9 – Переміщення обличчя в базу faces_db

Скріншот після переміщення обличчя в базу faces_db. З'являється підтвердження успіху.

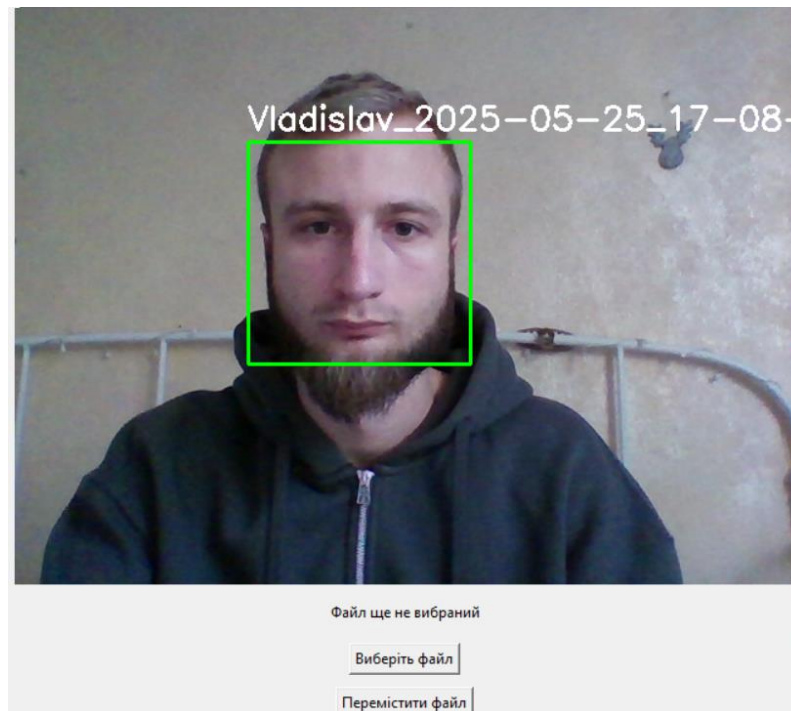


Рисунок 2.10 – Успішне розпізнавання обличчя

Після додавання нового обличчя програма успішно його ідентифікує у потоці.

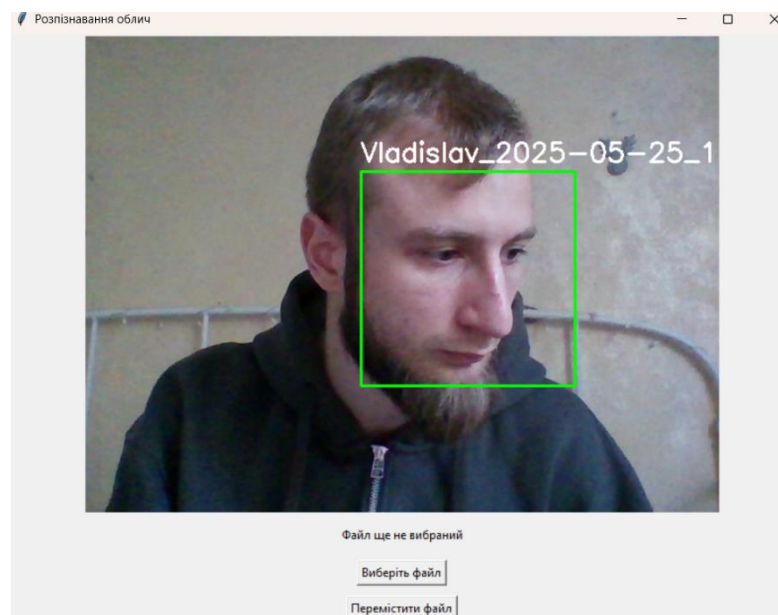


Рисунок 2.11 – Незначний нахил голови: обличчя розпізнається

Невеликий нахил голови не заважає роботі алгоритму — обличчя успішно розпізнається.

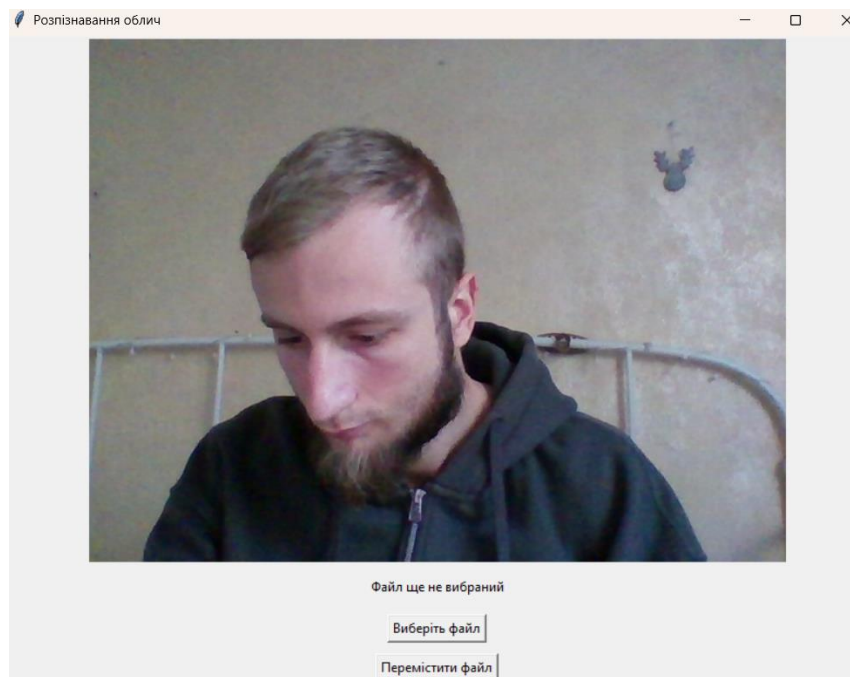


Рисунок 2.12 – Сильний нахил голови: обличчя не розпізнається

Сценарій, коли обличчя сильно нахилене — алгоритм не завжди розпізнає його коректно.

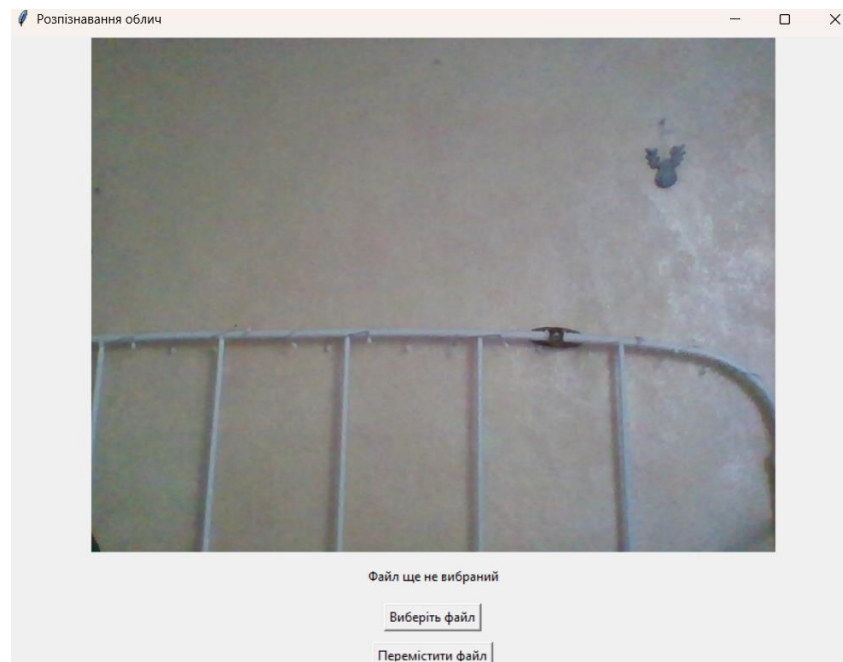


Рисунок 2.13 – Відсутність обличчя у кадрі: програма працює стабільно

Коли обличчя у кадрі відсутнє, програма працює стабільно й не видає помилок.

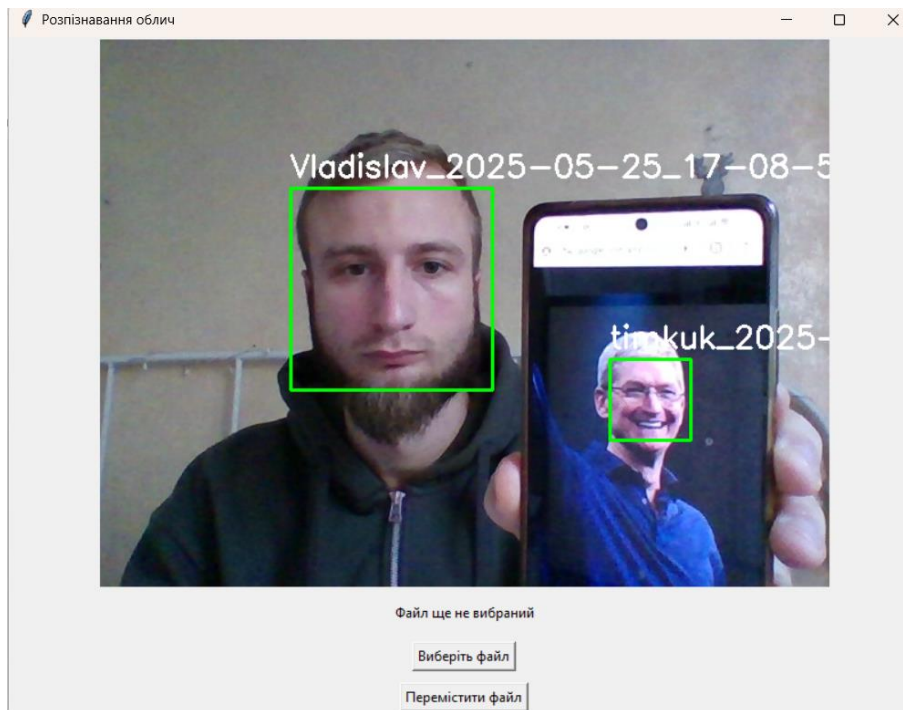


Рисунок 2.14 – Розпізнавання двох обличч (приклад із доданим обличчям)

Два обличчя одночасно розпізнані: додане обличчя користувача та обличчя з екрана телефона.

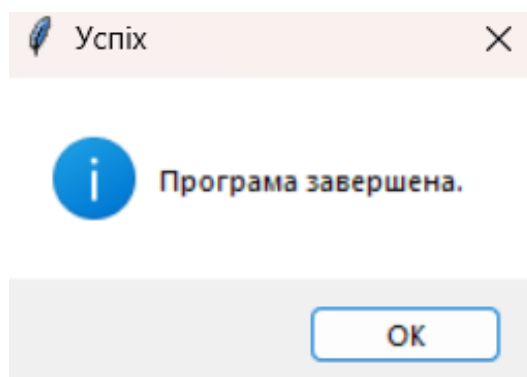


Рисунок 2.15 – Завершення роботи програми

Повідомлення про завершення роботи програми.

В результаті тестування встановлено, що система коректно виконує детекцію облич у режимі реального часу, успішно порівнює з базою даних і додає нові зображення при потребі. Інтерфейс зручний у використанні, основна функціональність працює стабільно. Виявлені незначні обмеження при частковому перекритті обличчя або його сильному нахилі, що не критично для поставлених цілей.

Висновки до розділу 2

У даному розділі було розроблено та реалізовано програмний додаток для детекції облич у відеопотоці з використанням бібліотеки OpenCV. Архітектура додатку побудована за шаблоном Model-View-Controller (MVC), що забезпечило чітке розділення логіки обробки, відображення та керування даними. У процесі реалізації було створено зручний графічний інтерфейс на основі Tkinter, модуль розпізнавання облич із використанням гістограм яскравості та модуль обробки подій і логування.

Результати тестування підтвердили працездатність системи в режимі реального часу. Програма стабільно розпізнає відомі обличчя, пропонує зберегти нові, не дає збоїв при відсутності об'єктів у кадрі, а також демонструє стійкість до незначних змін положення голови. Зафіксовано окремі обмеження — наприклад, погіршення точності при сильному нахилі або частковому перекритті обличчя, що є типовим для алгоритмів на основі каскадів Хаара.

Загалом, поставлене завдання виконано в повному обсязі: створено функціональний прототип системи детекції облич, перевірено його роботу, задокументовано результати та підготовлено матеріали для подальшого аналізу.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено програмний додаток для детекції облич у відеопотоці з використанням бібліотеки OpenCV та мови програмування Python.

Під час реалізації проєкту були виконані такі завдання:

- Проаналізовано методи та алгоритми розпізнавання облич.
- Обґрунтовано вибір архітектури програмного забезпечення на основі шаблону Model-View-Controller (MVC).
- Реалізовано графічний інтерфейс користувача з використанням бібліотеки Tkinter.
- Розроблено програмний модуль для обробки відеопотоку, детекції та розпізнавання облич.
- Забезпечено збереження нових облич у базі даних.
- Реалізовано функцію логування подій.
- Проведено тестування системи за різними сценаріями роботи.

Результати тестування підтвердили, що програмний додаток працює стабільно, виконує детекцію та розпізнавання облич у реальному часі, коректно обробляє нові дані та забезпечує взаємодію з користувачем через зручний інтерфейс.

Під час тестування було виявлено, що алгоритм коректно працює при стандартному розташуванні обличчя у кадрі, демонструє достатню стійкість до незначних змін положення голови та освітлення. Водночас зниження точності спостерігається при значному нахилі голови або її частковому закритті.

У цілому, поставлені у роботі цілі та завдання виконані повністю, а результати можуть бути використані як основа для подальшого розвитку систем розпізнавання облич та впровадження їх у практичні додатки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Software Foundation. Python Language Reference, version 3.12 [Електронний ресурс]. – Режим доступу: <https://docs.python.org/3/>
2. OpenCV. Open Source Computer Vision Library [Електронний ресурс]. – Режим доступу: <https://opencv.org/>
3. Tkinter – Python interface to Tcl/Tk [Електронний ресурс]. – Режим доступу: <https://docs.python.org/3/library/tkinter.html>
4. Pillow (PIL Fork) Documentation [Електронний ресурс]. – Режим доступу: <https://pillow.readthedocs.io/en/stable/>
5. NumPy Documentation [Електронний ресурс]. – Режим доступу: <https://numpy.org/doc/>
6. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features. // Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. – 2001.
7. Bradski G., Kaehler A. Learning OpenCV 4: Computer Vision with Python. – O'Reilly Media, 2019.
8. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016.
9. Zhang K., Zhang Z., Li Z., Qiao Y. Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks. // IEEE Signal Processing Letters, 2016.
10. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
11. Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector. // Proceedings of the European Conference on Computer Vision (ECCV), 2016.

12. OpenCV-Python Tutorials [Електронний ресурс]. – Режим доступу: <https://docs.opencv.org/>
13. Рудий С.Ю. Комп'ютерний зір та розпізнавання образів: навчальний посібник. – Київ: КНЕУ, 2020. – 128 с.
14. Слюсар В.І. Штучний інтелект: машинне навчання та комп'ютерний зір. – Київ: Кондор, 2021. – 276 с.
15. Gonzalez R.C., Woods R.E. Digital Image Processing. – Pearson Education, 2018.
16. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition (ResNet). // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
17. TensorFlow. An end-to-end open source platform for machine learning [Електронний ресурс]. – Режим доступу: <https://www.tensorflow.org/>
18. PyTorch. Open source machine learning library [Електронний ресурс]. – Режим доступу: <https://pytorch.org/>
19. Scikit-learn: Machine Learning in Python [Електронний ресурс]. – Режим доступу: <https://scikit-learn.org/stable/>
20. OpenVINO Toolkit [Електронний ресурс]. – Режим доступу: <https://www.intel.com/content/www/us/en/developer/tools/opencvino-toolkit/overview.html>

ДОДАТОК А

Код додатку з розпізнавання облич

Launcher.py

```
import tkinter as tk
from tkinter import messagebox
import subprocess
import os
import signal
import sys

class PythonLauncher:
    def __init__(self, master):
        self.master = master
        self.master.title("Лаунчер")
        self.master.geometry("300x150")
        self.process = None

        self.label = tk.Label(master, text="Управління програмою", font=("Arial", 12))
        self.label.pack(pady=10)

        self.start_button = tk.Button(master, text="Запустити програму",
command=self.start_program)
        self.start_button.pack(pady=5)

        self.stop_button = tk.Button(master, text="Завершити програму",
command=self.stop_program)
        self.stop_button.pack(pady=5)

    def start_program(self):
        if self.process is None or self.process.poll() is not None:
```

```

try:
    self.process = subprocess.Popen([sys.executable, 'controller.py'])
except Exception as e:
    messagebox.showerror("Помилка запуску", str(e))
else:
    messagebox.showinfo("Інфо", "Програма вже запущена.")

def stop_program(self):
    if self.process and self.process.poll() is None:
        try:
            # Завершення процесу
            self.process.terminate()
            self.process.wait()
            messagebox.showinfo("Успіх", "Програма завершена.")
        except Exception as e:
            messagebox.showerror("Помилка завершення", str(e))
    else:
        messagebox.showinfo("Інфо", "Програма не запущена.")

if __name__ == "__main__":
    root = tk.Tk()
    app = PythonLauncher(root)
    root.mainloop()

```

Controller.py

```

import cv2
import os
import shutil
from datetime import datetime
from PIL import Image
from model import FaceRecognitionModel

```

```

from view import FaceRecognitionView
import tkinter as tk

class FaceRecognitionController:
    def __init__(self, root):
        self.model = FaceRecognitionModel()
        self.view = FaceRecognitionView(root)

        self.faces_dir = "faces_db"
        self.unknown_dir = "unknown_faces"
        os.makedirs(self.unknown_dir, exist_ok=True)

        self.known_histograms, self.known_names = self.model.load_faces()
        self.video_capture = cv2.VideoCapture(0)
        if not self.video_capture.isOpened():
            raise Exception("Не удалось запустить камеру")

        self.selected_file = None
        self.name_requested = False
        self.name_input = ""

        self.view.btn_select.config(command=self.select_file)
        self.view.btn_move.config(command=self.move_file)

        self.update_frame()

    def update_frame(self):
        ret, frame = self.video_capture.read()
        if not ret:
            return

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = cv2.CascadeClassifier(cv2.data.harcascades + "haarcascade_frontalface_default.xml") \

```

```
.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5)
```

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

```
    face_crop = gray_frame[y:y + h, x:x + w]
```

```
    hist = cv2.calcHist([face_crop], [0], None, [256], [0, 256])
```

```
    hist = cv2.normalize(hist, hist).flatten()
```

```
    name = "Невідомо"
```

```
    max_corr = 0
```

```
for i, known_hist in enumerate(self.known_histograms):
```

```
    corr = self.model.compare_histograms(hist, known_hist)
```

```
    if corr > max_corr:
```

```
        max_corr = corr
```

```
        name = self.known_names[i]
```

```
if name == "Невідомо" and not self.name_requested:
```

```
    name = self.view.ask_name()
```

```
    self.name_requested = True
```

```
    timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
```

```
    date_folder = os.path.join(self.unknown_dir, datetime.now().strftime("%Y-%m-%d"))
```

```
    os.makedirs(date_folder, exist_ok=True)
```

```
    save_path = os.path.join(date_folder, f"{name}_{timestamp}.jpg")
```

```
    cv2.imwrite(save_path, frame[y:y + h, x:x + w])
```

```
    self.model.log_event(name, "Додано")
```

```
    self.known_histograms, self.known_names = self.model.load_faces()
```

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

```
cv2.putText(frame, name, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 255),
```

2)


```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
image = Image.fromarray(frame_rgb)
self.view.update_image(image)
self.view.root.after(50, self.update_frame)
```

```
def select_file(self):
    self.selected_file = self.view.ask_file()
    if self.selected_file:
        self.view.label_selected.config(text=f"Вибраний файл:
{os.path.basename(self.selected_file)}")
```

```
def move_file(self):
    if not self.selected_file:
        self.view.show_error("Спочатку виберіть файл!")
        return
    destination_folder = self.view.ask_directory()
    if not destination_folder:
        return
    try:
        destination_path = os.path.join(destination_folder, os.path.basename(self.selected_file))
        shutil.move(self.selected_file, destination_path)
        self.view.show_info(f"Файл переміщено в {destination_path}")
        self.known_histograms, self.known_names = self.model.load_faces()
        self.view.label_selected.config(text="Файл ще не вибраний")
    except Exception as e:
        self.view.show_error(f"Помилка: {e}")
```

```
def on_closing(self):
    self.video_capture.release()
    self.view.root.quit()
```

```
if __name__ == "__main__":
    root = tk.Tk()
```

```
app = FaceRecognitionController(root)
root.protocol("WM_DELETE_WINDOW", app.on_closing)
root.mainloop()
```

Model.py

```
import cv2
import os
import csv
from datetime import datetime
from openpyxl import Workbook, load_workbook

class FaceRecognitionModel:
    def __init__(self, faces_dir="faces_db", log_file="log.csv", excel_file="log.xlsx"):
        self.faces_dir = faces_dir
        self.log_file = log_file
        self.excel_file = excel_file
        os.makedirs(self.faces_dir, exist_ok=True)

    def load_faces(self):
        known_faces, face_names = [], []
        for filename in os.listdir(self.faces_dir):
            if filename.lower().endswith((' .jpg', ' .png')):
                path = os.path.join(self.faces_dir, filename)
                image = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
                hist = cv2.calcHist([image], [0], None, [256], [0, 256])
                hist = cv2.normalize(hist, hist).flatten()
                known_faces.append(hist)
                face_names.append(os.path.splitext(filename)[0])
        return known_faces, face_names

    def compare_histograms(self, hist1, hist2):
        return cv2.compareHist(hist1, hist2, cv2.HISTCMP_CORREL)
```

```

def log_event(self, name, status):
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(self.log_file, mode='a', newline="", encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow([now, name, status])

    if not os.path.exists(self.excel_file):
        wb = Workbook()
        ws = wb.active
        ws.title = "Log"
        ws.append(["Час", "Ім'я", "Статус"])
        wb.save(self.excel_file)

    wb = load_workbook(self.excel_file)
    ws = wb["Log"]
    ws.append([now, name, status])
    wb.save(self.excel_file)

```

View.py

```

import tkinter as tk
from tkinter import filedialog, messagebox, simpledialog
from PIL import Image, ImageTk

class FaceRecognitionView:
    def __init__(self, root):
        self.root = root
        self.root.title("Розпізнавання обличч")
        self.root.geometry("800x600")

        self.label_video = tk.Label(root)

```

```
self.label_video.pack()
```

```
self.label_selected = tk.Label(root, text="Файл ще не вибраний")
```

```
self.label_selected.pack(pady=10)
```

```
self.btn_select = tk.Button(root, text="Виберіть файл")
```

```
self.btn_select.pack(pady=5)
```

```
self.btn_move = tk.Button(root, text="Перемістити файл")
```

```
self.btn_move.pack(pady=5)
```

```
def update_image(self, image):
```

```
    photo = ImageTk.PhotoImage(image=image)
```

```
    self.label_video.config(image=photo)
```

```
    self.label_video.image = photo
```

```
def ask_name(self):
```

```
    while True:
```

```
        name = simpdialog.askstring("Додавання імені", "Введіть ім'я для нового обличчя (або  
залиште пустим):")
```

```
        if name is None:
```

```
            continue
```

```
        if not name.strip():
```

```
            messagebox.showwarning("Помилка", "Введіть ім'я!")
```

```
            continue
```

```
        return name
```

```
def show_error(self, message):
```

```
    messagebox.showerror("Помилка", message)
```

```
def show_info(self, message):
```

```
    messagebox.showinfo("Успіх", message)
```

```
def ask_file(self):  
    return filedialog.askopenfilename(title="Виберіть файл", filetypes=[("JPEG", "*.jpg"),  
("PNG", "*.png")])
```

```
def ask_directory(self):  
    return filedialog.askdirectory(title="Виберіть папку для переміщення")
```