

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ
РОЗПІЗНАВАННЯ ОБЛИЧ НА ВІДЕО

RESEARCH AND SOFTWARE IMPLEMENTATION OF THE APPLICATION
FOR FACE RECOGNITION ON VIDEO

Виконав: студент(ка) 2 курсу, групи КНТ-111м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Марков О.С.

(прізвище та ініціали)

Керівник Каплієнко Т.І.

(прізвище та ініціали)

Рецензент Шитікова О.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Маркова Олександра Сергійовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація застосунку розпізнавання облич на відео. Research and Software Implementation of the Application for Face Recognition on Video

керівник проєкту (роботи) Каплієнко Тетяна Ігорівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “04” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 01 грудня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Додатки

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Каплієнко Т.І., доцент		
Нормоконтролер	Белова А.В., асистент		

7. Дата видачі завдання “ 01 ” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-13 тижні	Додатки
8	Нормоконтроль та рецензування.	14–15 тижні	
9	Захист роботи.	16 тиждень	

Студент(ка)

(підпис)

Марков О.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Каплієнко Т.І.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
94 с., 12 табл., 33 рис., 4 дод., 39 джерел.

КОМП'ЮТЕРНА ПРОГРАМА, ЛОКАЛЬНІ БІНАРНІ ШАБЛони,
НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ ОБЛИЧ, OPENCV, PYTHON.

Об'єкт дослідження – технології нейромереж для обробки зображень та відео.

Предмет дослідження – програмні засоби для розпізнавання облич на відео.

Мета роботи – створення програмної системи для розпізнавання облич визначених відвідувачів на відео та надсилання сповіщення за результатами в месенджер Telegram.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування Python, графічна бібліотека інтерфейсів Tkinter, інтегроване середовище розробки Wing, персональний комп'ютер з процесором AMD Athlon під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний застосунок, який виконує розпізнавання облич на відео та сповіщення в месенджер Telegram про ідентифікацію заздалегідь визначених відвідувачів.

Висновки. Розроблено програмну систему для розпізнавання облич на відео та сповіщення за результатами в месенджер Telegram.

Галузь використання – розпізнавання облич на відео в системах індивідуальної безпеки.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 94 pages, 33 figures, 12 tables, 4 appendixes, 39 sources.

COMPUTER PROGRAM, LOCAL BINARY PATTERNS, NEURAL NETWORKS, FACE RECOGNITION, OPENCV, PYTHON.

Object of study are neural network technologies for image and video processing.

Subject of study are software tools for face recognition on video.

The purpose of the work is to create a software system to recognize the faces of certain visitors on video and send an alert on the results to Telegram messenger.

Materials, methods, and tools: structured and object-oriented programming, programming language Python, GUI library Tkinter, integrated development environment Wing, personal computer with AMD Athlon processor under the control of the Microsoft Windows 10 operating system.

Results. The software application is created which performed face recognition on video and notifications in the Telegram messenger about the identification of predefined visitors.

Conclusions. A software system that recognize faces and send an alert on the results to Telegram messenger has been developed.

The field of use is a face recognition on video in personal security systems.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Різниця між виявленням та розпізнаванням облич	11
1.2 Виявлення облич	11
1.3 Розпізнавання облич	12
1.4 Існуючі методи розпізнавання	13
1.5 Порівняння методів розпізнавання	18
1.6 Методи ідентифікації облич	18
1.7 Огляд існуючих аналогів та їх порівняння	21
1.8 Постановка завдання до роботи	23
2 Матеріали і методи	24
2.1 Обробка колекції зображень	24
2.2 Розпізнавання обличчя на відео та співставлення зі зразками	26
2.3 Сповіщення за результатами розпізнавання	27
2.4 Додаткові методи для роботи з відео	27
2.5 Висновки з розділу	28
3 Розробка архітектури програми	29
3.1 Структура та функціонування програми	29
3.2 Вибір інструментів розробки програми	32
3.3 Вимоги до складу технічних та програмних засобів	35
3.4 Висновки з розділу	36
4 Розробка програми	37
4.1 Основні рішення щодо реалізації класів	37
4.2 Основні рішення щодо використаних сервісів	43
4.3 Основні рішення щодо даних	45
4.4 Основні рішення щодо користувацького інтерфейсу	47

4.5 Висновки з розділу	50
5 Експлуатація, тестування, експериментальне дослідження програми	51
5.1 Експлуатація програми	51
5.2 Методика та результати тестування	51
5.3 Висновки з розділу	56
Висновки	57
Перелік джерел посилання	58
Додаток А Технічне завдання	62
Додаток Б Керівництво програміста	67
Додаток В Текст програми	72
Додаток Г Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AdaBoost	– Adaptive Boosting;
API	– Application Programming Interface;
COVID	– Coronavirus Disease;
GPS	– Global Positioning System;
GUI	– Graphic User Interface;
HTTP	– HyperText Transfer Protocol;
IDE	– Integrated Development Environment,
LAG	– Large Age-Gap;
ML	– Machine Learning;
OpenCV	– Open Source Computer Vision Library;
PIL	– Python Image Library;
pip	– Package Installer for Python;
RGB	– Red Green Blue;
SIFT	– Scale-invariant feature transform;
SMS	– Short Message Service;
XML	– Extensible Markup Language;
БД	– база даних;
ЕОМ	– електронно-обчислювальна машина;
КЗ	– комп’ютерний зір;
ЛБШ	– локальні бінарні шаблони;
МГК	– метод головних компонент;
ПЗ	– програмне забезпечення;
ПК	– персональний комп’ютер;
ПММ	– приховані марковські моделі;
РО	– розпізнавання облич;
СКВ	– система контролю версій;
ФГ	– фільтр Габора.

ВСТУП

Розпізнавання облич (РО) є прикладом практичного застосування теорії розпізнавання образів, що полягає в автоматичній локалізації обличчя на зображенні або відео, і, за необхідності, ідентифікації особи. Більшість його методів ґрунтуються на порівнянні рис обличчя заданого зображення з обличчями, що зберігаються в базах даних (БД).

Попри те, що системи, засновані на біометричних технологіях на кшталт розпізнавання відбитків пальців або райдужної оболонки є вищими за точністю, задача РО все одно стає більш актуальною. Найбільш поширеними галузями її застосування є комп'ютерні ігри та віртуальна реальність, персоналізація пристроїв, соціальні сервіси, шифрування даних тощо. Як правило, потреба ідентифікувати обличчя людини на відео виникає в реальному часі. Наприклад, для виявлення правопорушників серед перехожих вуличною камерою відеоспостереження на основі існуючої БД, підрахунку облич відвідувачів з метою запобігання скупчень в закладах у розпал Coronavirus Disease (COVID) тощо.

Ще одним актуальним прикладом є РО домофонними системами та подальше оповіщення про те, хто прийшов, засобами соціальних мереж та месенджерів. Такий механізм може бути корисним в час тривалої відсутності господарів вдома, адже облік гостей (бажаних, і, особливо, небажаних) здійснюється у форматі 24/7. Вирішення даного завдання може бути доволі складним, адже постають проблеми щодо способів зберігання даних, вибору методів РО на відео, а також організації необтяжливого користувацького інтерфейсу.

На сьогодні створена та запроваджена велика кількість рішень, що розглядають описану проблему комплексно, тобто, кінцевим користувачам пропонують системи, згруповані за принципом «Відеоспостереження + сигналізація». Як правило, це занадто дороговартісно та перевищує пересічні потреби в безпеці об'єктів.

Слід зазначити, що попит на технології, що пов'язані з РО на відео, зростає, і, в той же час, зростають вимоги щодо точності ідентифікації. Наслідком цього є ускладнення як розробки, так і супроводження відповідного програмного забезпечення (ПЗ). Таким чином, важливим завданням є знаходження балансу між простотою, ефективністю та складністю процесу розпізнавання.

Розроблене ПЗ має здійснювати процес РО на відео та виконувати сповіщення засобами месенджера Telegram. В подальшому воно може бути використаного в якості компоненту іншого ПЗ, що орієнтоване на більш детальний аналіз облич, зафіксованих відеозасобами відвідувачів.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Різниця між виявленням та розпізнаванням облич

Насамперед потрібно означити різницю між виявленням (англ. *detecion*) та розпізнаванням (англ. *recognition*) облич. Обидва ці терміни доволі часто використовують при аналізі результатів відеоспостереження як синоніми, але це є неточністю.

Виявлення обличчя полягає в зборі знімків і відеозаписів для подальшого перегляду вручну або з використанням спеціального ПЗ.

Розпізнавання – це аналітична процедура, в ході якої зіставляються характерні особливості об'єкта, що спостерігається [1].

1.2 Виявлення облич

Системи РО умовно відносять до складних систем розпізнавання образів [2], оскільки процес роботи не обмежується тільки класифікацією пікселів на зображеннях або відеокадрах.

Загальний процес роботи системи РО має кілька етапів.

Першим етапом в процесі РО є виявлення облич: на вхід програма отримує зображення з відеозапису (відбувається перебір кадрів відео у циклі) або з БД, та починає виявляти межі обличчя на ньому. Метою цього етапу є знаходження локації (розташування) обличчя на зображенні та його розмірів.

Щоб локалізувати обличчя на зображенні виділяють його основні компоненти, такі як ніс, ніздрі, брови, рот, губи, вуха тощо [3]. Таким чином, визначаються точні або ж приблизні координати обличчя за його наявності.

В результаті потрібний фрагмент зображення витягається (зазвичай вихідне зображення обрізають з урахуванням координат вищеописаних локацій) та переходить на етап розпізнавання.

1.3 Розпізнавання облич

На етапі розпізнавання виявлений фрагмент зображення з обличчям людини спершу трансформується: відбувається вирівнювання, коригування за яскравістю, контрастністю, інверсія, порогова фільтрація тощо. Метою цього є підвищення ефективності та якості РО, адже зазвичай ознаки вхідних зображення є викривлені або недостатньо ідентифіковані.

Далі здійснюється обчислення ознак і їхнє порівняння з еталонними значеннями. В результаті, залежно від логіки ПЗ, здійснюється верифікація («Чи є особа на відео тією, з якою ми порівнюємо (на зображенні)?») чи ідентифікація («Якому зі зразків облич в БД відповідає ця особа? Хто це?»).

Загальну схему виявлення та РО, що реалізується в ПЗ, представлено на рис. 1.1.



Рисунок 1.1 – Загальна схема виявлення та РО

1.4 Існуючі методи розпізнавання

Існуючі методи розпізнавання у загальному випадку мають наступний алгоритм:

- перетворення вихідного зображення за певними стандартами розмірів, напівтонів, яскравості тощо;
- виділення ключових ознак;
- механізм моделювання.

Найбільш популярні методи, які застосовуються в сучасних системах РО:

- приховані марковські моделі (ПММ);
- метод Віюлі-Джонса;
- фільтр Габора (ФГ);
- метод головних компонент (МГК).

1.4.1 Приховані марковські моделі

ПММ є статичним методом, що використовується для РО.

Елементами в цих моделях є:

- множина прихованих станів;
- множина станів, що спостерігаються;
- матриця перехідних ймовірностей;
- початкова ймовірність станів.

Кожному стану ставиться у відповідність своя марковська модель («один до одного»). При РО здійснюється перевірка проініціалізованих об'єктів БД марковської моделі. Метою цього є знаходження максимальної вірогідності того, що послідовність спостережень (в контексті РО – потоку відеокادрів) генерується відповідною ПММ.

Оскільки цей метод використовує статистичні параметри просторових характеристик сигналів, то швидкість РО залишає бажати кращого.

Попри це ПММ активно застосовуються в учбових цілях при РО, зокрема, у випадках, якщо особа на відео здійснює певну активність. Таким чином, цей метод може використовуватися для вивчення динаміки між послідовними появами обличчя на відео [4].

Ще однією областю застосування ПММ є онлайн-співставлення карт дорожніх мереж [5] для збільшення точності при відображенні траєкторій Global Positioning System (GPS).

1.4.2 Метод Віюли-Джонса

Для РО в режимі реального часу доволі часто застосовують алгоритм КЗ Віюли-Джонса. Навчання класифікаторів, як правило, йде занадто повільно, але результати пошуку образів отримуються доволі швидко. Підбиваючи підсумки в питанні швидкості, можна стверджувати, що метод Віюла-Джонса є одним з кращих за співвідношенням показників ефективності розпізнавання та швидкодії.

Цей детектор має низьку ймовірність помилкового виявлення за умови розташування образів під кутом до 30 градусів. Окрім того, для застосунків, де висота кадрів не є необхідною, метод Віюли-Джонса дозволяє здійснювати постобробку та аналіз [6].

Загалом алгоритм Віюли-Джонса має 4 етапи:

- вибір примітиву Хаара;
- створення інтегрального зображення;
- застосування Machine Learning (ML) алгоритму AdaBoost (adaptive boosting);
- каскадні класифікатори.

Основу методу Віюли-Джонса складають примітиви Хаара, що є розбивкою прямокутної області на набори різнотипних під областей прямокутної форми. В процесі розвитку методу примітиви Хаара

видозмінено шляхом їх нахилення на кут в 45 градусів та застосування несиметричних конфігурацій як показано на рис. 1.2 [7].

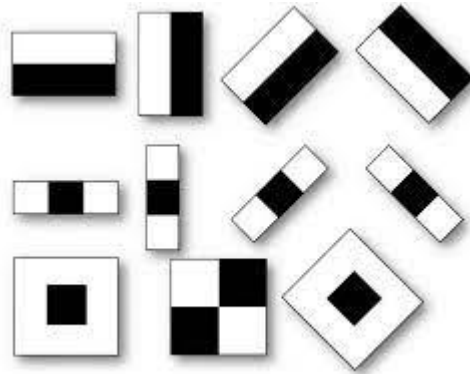


Рисунок 1.2 – Примітиви Хаара

Метою цих змін, запропонованих Лінхартом та Майдом [8], було збільшення розмірності простору ознак.

Вибір саме прямокутних примітивів Хаара замість більш зручних математичних моделей на кшталт синусоїд пояснюється тим, що ці примітиви значно швидше обчислюються через інтегральне представлення (другий крок алгоритму Віюлі-Джонса). Використання інших згорток потребує кількість дій, що пропорційна квадрату розміра примітиву.

Дані ознаки позиціонуються на вихідному зображенні, підсумовується інтенсивність пікселів в областях, після чого вираховується різниця між сумами. Таким чином, отримуються значення ознак.

Вибір ML-алгоритму AdaBoost як складової для алгоритму Віюлі-Джонса пояснюється його значними перевагами:

- задовільна узагальнююча здатність, тобто, для більшості реальних задач РО вдається будувати композиції, що перевершують за ефективністю базові алгоритми [9];
- відносна простота реалізації;
- покращена швидкодія, оскільки час побудови композиції визначається часом навчання базових алгоритмів;
- можливість аналізу об'єктів, які за своєю сутністю є шумом.

В результаті еволюції методу вдалося відійти в бік від піксельного представлення зображення, зберігши швидкість обчислення ознак навіть для облич з поворотом на кут в межах тридцяти градусів.

1.4.3 Фільтр Габора

Особливістю ФГ є його подібність сприйняття зображень до фільтрів зорової системи людини. Важливим відкриттям в нейрофізіології стала подібність рецептивних полів нейронів первинної зорової кори головного мозку до різноманітних значень кута орієнтації ФГ [10].

Обличчя в якості вхідного значення представляється графом на основі антропометричних точок (вище описано основні компоненти обличчя). Значення ознак в вершинах графу обчислюються саме з використанням ФГ.

За своєю сутністю, цей метод представляє лінійний фільтр, що використовується для аналізу текстури зображення – ФГ аналізує, чи є присутнім конкретний частотний вміст в зображенні на певній локалізованій ділянці. Таким чином, даний метод особливо добре підходить для розрізнення текстур.

ФГ є стійким в застосуванні до зображень з різними масштабами, положенням на площині та освітленням. З іншого боку, цей метод має занадто високу обчислювальну складність, що обумовлює його низьку технологічність.

Відомо, що для порівняння вхідного зображення з 87 еталонами за методом ФГ витрачалося близько 25 секунд. Дане дослідження проводилося 1993 року на електронно-обчислювальній машині (ЕОМ) з 23 трансп'ютерами [11], для аналогічних експериментів повідомлялося, що час роботи просто завеликий.

1.4.4 Метод головних компонент

МГК є самостійним статистико-математичним методом багатовимірної аналізу [12]. Його загальне призначення – зменшувати розмірність даних із мінімальною втратою інформації.

Ідеї МГК в застосуванні до відео та до зображень дещо різняться. В контексті обробки зображень, МГК дозволяє представити людське обличчя вектором малої розмірності, що порівнюється з еталонними векторами із БД. Це досягається видаленням шуму із блоку пікселів, представлення околу обробленого блоку набором точок багатовимірної зображення [13]. Далі МГК прибирає компоненти зображення з шумом. Що стосується обробки відео, МГК розглядає кожен відеокадр декількома значеннями, що в подальшому використовується для побудови БД. Така заміна є суттєвим зменшенням розмірності, більш того, зростає стійкість до ряду спотворень і дефектів на відеокадрах.

Спочатку весь навчальний набір зображень обличч перетворюється в одну загальну матрицю даних, де кожен ряд відповідає екземплярові зображення. Очевидно, що усі зображення мають бути стандартизовані за розміром та бути приведені за яскравістю. Таким чином, швидкість обробки зображень із застосуванням МГК може бути доволі високою за ряду умов:

- зображення приведені за розміром до певного стандарту;
- обличчя на зображеннях не мають значних «перекосів» за освітленістю;
- обличчя не мають значних змін за виразом (зображення гримас псуватимуть продуктивність).

На жаль, дані умови на практиці доволі складно втілити при зборі датасетів.

1.5 Порівняння методів розпізнавання

Щоб вибрати оптимальний метод РО для вирішення завдання, усі вищерозглянуті методи порівняно за точністю та швидкістю. Результати порівняння представлено в табл. 1.1.

Таблиця 1.1 – Порівняння методів РО

Метод РО	Точність	Швидкість
ПММ	Незадовільна, оскільки максимізується відгук кожного зображення на свою модель без мінімізації інших моделей	Незадовільна через використовувані параметри
Метод Віоли-Джонса	Низька ймовірність помилкового виявлення обличчя	Задовільна для більшості об'єктів різного розміру та складності
ФГ	Задовільна, зокрема для розрізнення текстур	Незадовільна через обчислювальну складність
МГК	Задовільна	Задовільна за ряду виконання умов

Враховуючи результати порівняння з табл. 1.1, було прийнято рішення щодо використання реалізації методу Віоли-Джонса.

1.6 Методи ідентифікації облич

Важливе значення має і група алгоритмів, які відповідають за ідентифікацію особи по обличчю. В загальному вигляді відбувається порівняння вхідного зображення обличчя з набором даних з метою знаходження особи, що відповідає даному обличчю [14] (порівняння «один до багатьох»). Усі вони оперують кольором та яскравістю пікселів

заздалегідь оброблених зображень (фрагмент обличчя обрізано, вирівняно у фронтальному положенні тощо), як описано в попередніх підрозділах.

Основні алгоритми даної групи:

- Eigenfaces;
- локальні бінарні шаблони (ЛБШ);
- scale-invariant feature transform (SIFT).

1.6.1 Eigenfaces

Алгоритм Eigenfaces (дослівно «власне обличчя»), запропонований Терком і Пентландом [15], був одним із перших успішних методів РО.

Eigenfaces розглядає відмінні характеристики облич за допомогою двовимірних зображень в градаціях сірого.

Ідея Eigenfaces полягає в представленні зображення точкою багатовимірного простору зображень, на якому здійснюється співставлення подання з так званого маловимірного простору. Ефективність цього підходу досягається мінімізацією дисперсії всередині класу та одночасною максимізацією дисперсії між класами [16].

Ступінь схожості обличчя з кадром та шаблонів в БД визначається за допомогою коефіцієнтів відмінності. Визначення цих коефіцієнтів і дозволяє виконати ідентифікацію особи.

Алгоритм Eigenfaces демонструє високі результати, якщо вхідні зображення отримано в добре освітлених приміщеннях. Це пов'язано з тим, алгоритм опрацьовує як ознаки осіб, так і освітленість зображень. В інших умовах його ефективність різко знижується.

1.6.2 Локальні бінарні шаблони

Основною перевагою ЛБШ (англ. Local Binary Patterns) є його ефективність при розпізнаванні як локальних, так і глобальних особливостей зображення, а також простота в реалізації.

Для знаходження 256-розмірного вектора ознак ЛБШ вікно пошуку ділиться на чарунки пікселів, а далі для кожного пікселя в чарунці здійснюється його порівняння з сусідами. Якщо інтенсивність пікселя-сусіда більше або дорівнює інтенсивності центрального пікселя, то сусід позначається 1, а інакше 0. За отриманими частотами появи «чисел» (комбінацій пікселів більше та менше центрального) будуються гістограми. Нормалізовані та конкатеновані гістограми усіх чарунок і дають шуканий вектор.

Перевагою такого підходу є урахування дрібних деталей зображення, адже фіксований розмір вікна пошуку не дозволяє кодувати деталі зображення різної розмірності.

У порівнянні з Eigenfaces, алгоритм ЛБШ є стійким до змін освітлення, що зумовлює його кращу пристосованість до обробки в реальному часі.

1.6.3 Scale-invariant feature transform

Алгоритм SIFT дозволяє отримувати ключові «точки інтересу» для будь-якого об'єкту на зображенні. Очевидно, що для обличчя такими точками є межі його основних компонент, наприклад, високо контрастні ділянки очей або рота. SIFT демонструє ефективність ідентифікації навіть серед шуму або часткового перекриття [17], але алгоритм не виявить ознак при порівнянні зображень, якщо відбуватимуться зміни у їхній внутрішній геометрії. Простіше кажучи, застосування SIFT на рухомих чи гнучких об'єктах є малоефективним.

1.7 Огляд існуючих аналогів та їх порівняння

1.7.1 Програма «ZoneMinder»

Офіційний сайт програми: <https://zoneminder.com/>

Дане ПЗ, вікно «Налаштування фільтру» якого представлено на рис. 1.3 [18], є потужним інструментом, що дає змогу досліджувати відео як з комп'ютера, так і зі смартфона. Платформи – Android, Linux, iOS.

Рисунок 1.3 – Налаштування фільтру в ZoneMinder [18]

Комерційна версія даного продукту дозволяє здійснювати глибокий аналіз створених скріншотів з відео. Також користувачі мають можливості керувати зонами розпізнавання на відео та регулювати чутливість камер. Сповіщення формуються у вигляді Short Message Service (SMS) та електронних листів.

1.7.2 Програма «Netcam Studio»

Офіційний сайт програми: <https://www.netcamstudio.com/>

Дане ПЗ, вікно «Налаштування відеокамери» якого представлено на рис. 1.4 [19], призначено передусім для організації відеоспостереження. Платформи – Android, iOS, Windows.

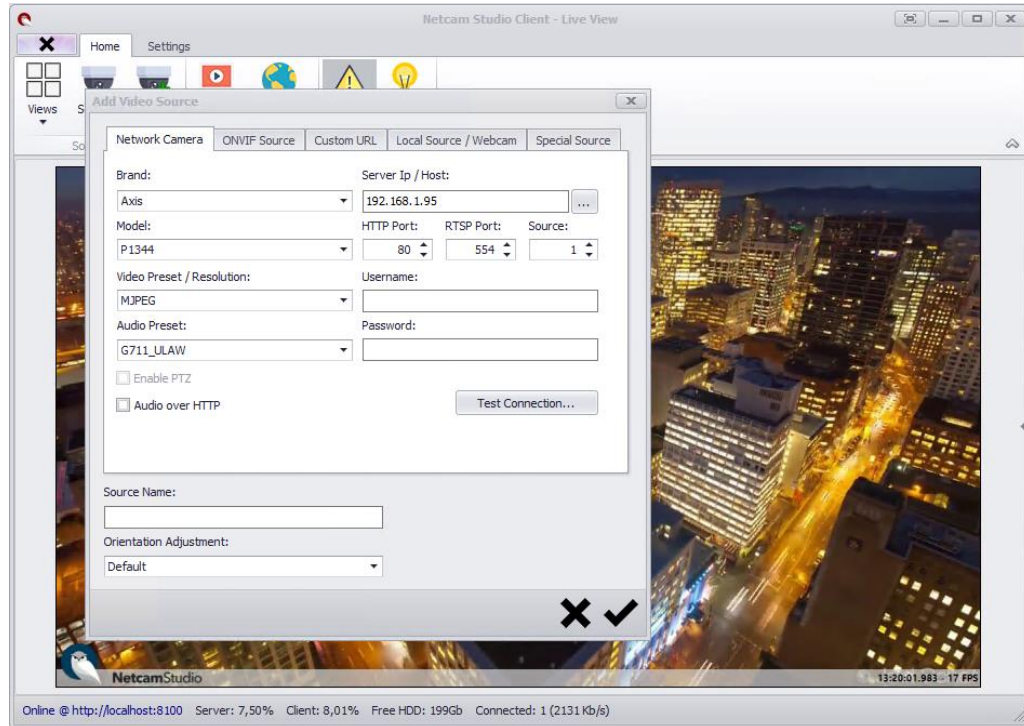


Рисунок 1.4 – Налаштування відеокамери в Netcam Studio [19]

Базовий функціонал Netcam Studio містить в собі роботу з аналоговими, а також IP- та USB- камерами, дозволяє синхронно переглядати та записувати відео з декількох джерел. Сповіщення щодо виявлених подій/осіб формуються у вигляді push-повідомлень, SMS, електронних листів.

Безкоштовна версія даного ПЗ має низку обмежень: обмеження кількості використовуваних камер (не більше двох), сповіщень (не більше п'яти на добу), водяний знак на відео.

1.7.3 Порівняння аналогів

Існуючі рішення, описані в підрозділах 1.7.1 та 1.7.2 було проаналізовано та порівняно з точки зору поставленої задачі. Результати порівняння (табл. 1.2) засвідчують, що обидва рішення мають як переваги, так і недоліки, що спонукає до розробки власного ПЗ.

Таблиця 1.2 – Порівняння існуючих аналогів

Критерій порівняння	ZoneMinder	Netcam Studio
Багатозадачність	+	+
Кросплатформність	+	+
РО «в один клік»	–	–
Збереження скріншотів	+	–
Сповіщення в соцмережах	–	+
Обмеження безкоштовної версії	–	–

1.8 Постановка завдання до роботи

За результатами дослідження проблеми РО на відео, було формалізовано завдання до роботи – розробка застосунку, що задовольняв би критерії наведені в підрозділі 1.7.

Також за результатами аналізу було сформовано технічне завдання, яке представлено в додатку А.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Обробка колекції зображень

Першочерговим завданням програми, що виконує РО на відео, є обробка колекції зображень, що містять обличчя потенційних відвідувачів. Очевидно, що цей крок визначає ефективність при співставленні розпізнаного з відео обличчя та облич з колекції зображень.

2.1.1 Обрізування зображень

Обробка зображень розпочинається з обрізування зображень, тобто необхідно на вихідному зображенні [20] (прикладом є рис. 2.1) виділити обличчя та зберегти його як окреме зображення (рис. 2.2).



Рисунок 2.1 – Приклад вихідного зображення [20]



Рисунок 2.2 – Приклад зображення з виокремленим обличчям

Як зазначалося в попередньому розділі, виявлення обличчя за характерними рисами на вихідному зображенні є доволі складним завданням. Приклад демонструє типові проблеми цього етапу: неодномірність освітлення, складність ракурсу (гострий кут нахилу обличчя до камери), фігура іншої особи як «зайвий» статичний елемент сцени.

2.1.2 Переведення в градації сірого

Людське око краще сприймає кольорові зображення, але алгоритми комп'ютерного зору (КЗ) краще опрацьовують зображення в градаціях сірого кольору. Слід зазначити, що поняття чорно-білого зображення та зображення в градаціях сірого не є тотожними поняттями. Поняття «в градаціях сірого» в контексті досліджуваної предметної області є більш точним.

Тому логічним є здійснити перетворення виокремленого обличчя в просторі Red Green Blue (RGB) за формулою [21] (2.1):

$$RGB \text{ to Gray : } Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B, \quad (2.1)$$

де R – канал Red;

G – канал Green;

B – канал Blue.

Загальною причиною для такого перетворення є можливість усувати зайві елементи, які «відволікають» алгоритми КЗ та негативно впливають на швидкодію РО. Ключовим «зайвим» елементом є колір. Його усунення надає наступні переваги:

- підкреслення текстури обличчя;
- згладжування відносної яскравості окремих елементів обличчя;
- зменшення ефекту від тіней.

Приклад виділеного на зображенні обличчя в градаціях сірого представлено на рис. 2.3.



Рисунок 2.3 – Приклад обличчя в градаціях сірого

Результатом даного етапу є фрагмент вихідного зображення, що містить обличчя, переведений в градації сірого. Його можна зберегти в директорію для подальшої передачі алгоритмам КЗ та можливості надсилання користувачам як результату початкового виявлення обличчя в зображенні.

2.1.3 Вектор рис обличчя

Останній крок обробки зображень перед їхнім використанням для співставлення – переведення рис обличчя у векторний формат. Взагалі кожне обличчя на зображенні може бути представлено у вигляді вектора розмірності 128 [22]: цього цілком досить для кодування усіх рис. Проте для подальшого співставлення достатньо першого елементу вектору з індексом 0.

Дуже важливо зберігати ці елементи у вигляді масиву, адже не виключено, що запропоновані зразки зображень є груповими фото.

2.2 Розпізнавання обличчя на відео та співставлення зі зразками

Процес співставлення обличчя на відео з обробленими зразками відбувається при циклічному отриманні кадрів з відео. Оскільки кожний кадр

розглядається не інакше як зображення, то кроки всередині циклу подібні до вищеописаних: обличчя на кадрі локалізуються та отримуються їхні кодування та відбувається, власне, співставлення кодувань зображень між собою. Співпадіння в кодуваннях вказує, що обличчя особи на відео вже є в колекції вихідних зображеннях.

2.3 Сповіщення за результатами розпізнавання

Згідно до результатів роботи загальної функції РО необхідно сформулювати сповіщення для користувача в месенджері Telegram, яке б давало відповідь на питання «Чи з'явився на визначений на відеозаписі?»

У випадку позитивної відповіді, текстове повідомлення доцільно підкріпити обробленим зображенням зі сховища, прикладом якого є рис. 2.3. Тобто, користувач застосунку замість вихідного зображення із своєї колекції побачить фрагмент-обличчя, «спійманий» детектором.

Також, якщо користувач в процесі роботи створював скріншоти, їх також можна надіслати, незалежно від результатів РО. Якщо ж скріншоти не створювалися, про це також можна оповістити користувача.

Тексти повідомлень представлено в «Керівництві програміста» (додаток Б).

Додатковим повідомленням може бути час здійснення РО у форматі «DD.MM.YY hh:mm:ss». Слід зазначити, що при обробці відео у реальному часі, процеси відеозйомки та РО є паралельними. Оскільки в розроблюваній програмі йде РО на вже відзнятих відеофайлах, вказуватимуться дата і час саме для етапу РО.

2.4 Додаткові методи для роботи з відео

Окрім інформування користувачів щодо приходу визначених відвідувачів, важливим є надання додаткових методів для роботи з

досліджуваним відео. Очевидно, що користувачу важливо дати змогу обрати та продивитись відеозапис з можливістю призупиняти та відновлювати перегляд, а також робити скріншоти кадрів, що є значущими до нього.

Вибір відеозапису за замовчуванням здійснюється з директорії датасету, проте немає сенсу встановлювати жорсткі обмеження. Тобто, користувач може запускати відеозаписи як із накопичувача, так із інших директорій на диску.

2.5 Висновки з розділу

На даному етапі роботи було проаналізовано матеріали та методи, а також викладено ідеї щодо вдосконалення існуючих підходів до розробки аналогічного ПЗ.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Структура та функціонування програми

3.1.1 Алгоритм роботи програми

З точки зору кінцевих користувачів ПЗ – власників приватних домофонних систем або інших специфічних систем відеоспостереження, більш доцільним є наступний алгоритм роботи (блок-схему представлено на рис. 3.1):

- первинне знайомство з довідковою інформацією по роботі з програмою;
- можливість оновлення існуючої колекції зображень (мається на увазі копіювання зображень з флешки до сховища програми в «один клік» без ручного «перетягування», що може спричиняти масу помилок та плутанини);
- вибір відеозапису – результату відеоспостереження за певний період часу;
- перегляд відеозапису з можливостями ставити його на паузу, відновлювати програвання та робити скріншоти;
- розпізнавання облич на відео «в один клік»;
- надсилання сповіщення в соцмережу Телеграм: чи виявлено фото визначених відвідувачів, їхню кількість, а також скріншоти, самостійно додані користувачем.

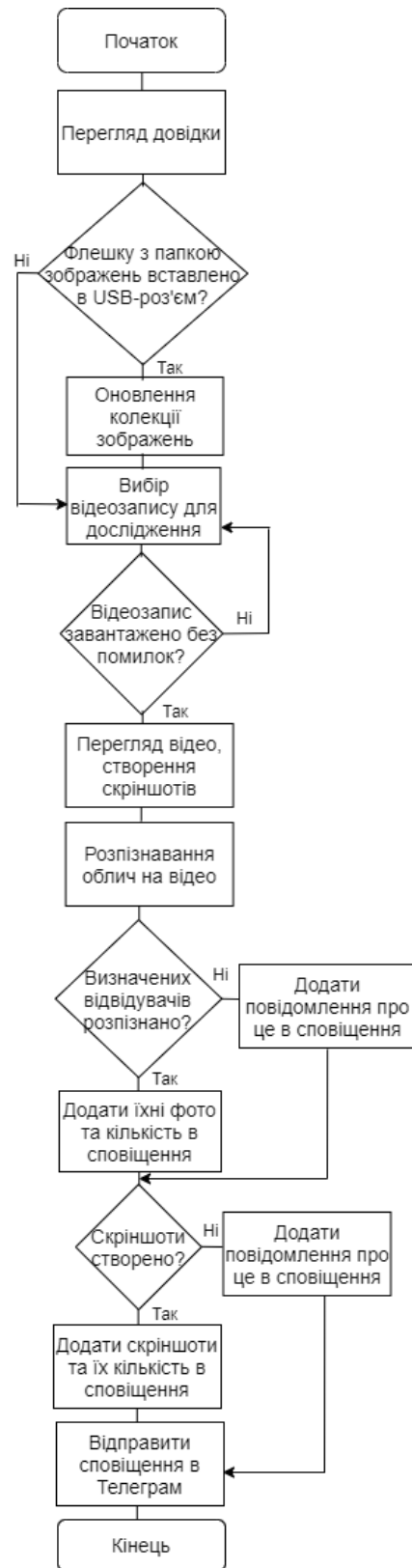


Рисунок 3.1 – Приклад вихідного зображення

3.1.2 Файлова структура програми

Файлову структуру розроблюваного десктопного застосунку складають:

- головний модуль, що виконує запуск проєкту;
- пакет з модулями проєкту (константи, окремі функції тощо);
- директорія, що представляє датасет (набори зображень та відеозаписів);
- директорія для зберігання скріншотів, котрі доречно відділити від датасету;
- директорія, що містить модель для РО (.xml-файл);
- набір незалежних програмних компонент в окремій директорії, що надають Application Programming Interface (API) для роботи з ML-алгоритмами, обробки зображень тощо;
- текстовий файл, що містить перелік бібліотек, необхідних для виконання програми. Щоб не складати цей перелік вручну, можна використати спеціальну команду системи управління пакету Package Installer for Python (pip): `pip freeze > libraries.txt`. Дана команда, виконана через термінал в директорії проєкту, записує перелік бібліотек до текстового файлу `libraries.txt`;
- директорія, що містить файл-іконку застосунку.

Вищеописану структуру проєкту у вигляді дерева каталогів представлено на рис. 3.2:

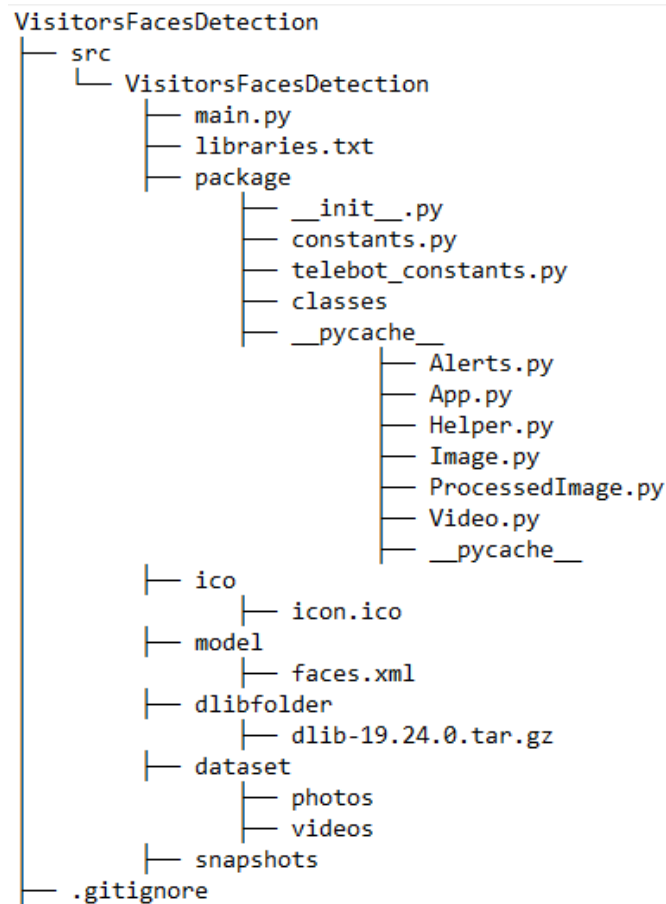


Рисунок 3.2 – Файлова структура проєкту

Для керування змінами під час розробки проєкту використано систему контролю версій (СКВ) Git. Частину файлів та директорій проєкту слід «приховати» від Git: тимчасових (`__pycache__`/, `libraries.txt`), конфігураційних (`dlibfolder`/, `ico`/, `model`/), секретних (`telebot_constants.py`), датасетів (`dataset`/). Для цього вони додані в файл `.gitignore`.

3.2 Вибір інструментів розробки програми

Важливим етапом після здійснення аналізу предметної області, матеріалів та методів є вибір необхідних інструментів та технологій, що будуть задіяні на етапі розробки: мови програмування, Integrated Environment Environment (IDE), графічної бібліотеки, додаткових бібліотек тощо.

3.2.1 Вибір мови програмування

Основним інструментом для розробки застосунку є мова програмування. Ключовими параметрами при порівнянні та виборі є підтримка ООП-парадигми, продуктивність, підтримка бібліотек ML, простота розробки Graphic User Interface (GUI), лаконічність синтаксису, крива навчальності, широке ком'юніті розробників. Для популярних мов програмування C++, Java, Python, що можуть вирішувати поставлену задачу, здійснено порівняння. Його результати занесено в таблицю 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій порівняння	C++	Java	Python
Підтримка ООП-парадигми	++	++	++
Продуктивність	++	++	+
Підтримка ML бібліотек	+	+	++
Простота розробки GUI	+	±	+
Лаконічність синтаксису	±	±	++
Крива навчальності	+	+	++
Широке ком'юніті розробників	++	++	++

За результатами порівняння мовою розробки було вибрано Python – ефективну за параметрами швидкості та обсягу програм, чистоти «коду», адаптовану до реалізації бізнес-завдань, застосуванню в різноманітних прикладних сферах, наукових й академічних дослідженнях [23].

3.2.2 Вибір графічної бібліотеки

Для розробки графічного інтерфейсу було вибрано кросплатформну графічну бібліотеку Tkinter на основі Tk. Вона містить велику кількість графічних компонентів та є складовою стандартної бібліотеки Python. Взагалі Tkinter є множиною обгортки, що впроваджують Tk-віджети, як Python-класи [24].

3.2.3 Вибір середовища для розробки

Середовище для розробки оцінюється за такими критеріями як наявність вбудованих плагінів, автодоповнення, зручність роботи з терміналом, підсвічування коду, загальний обсяг можливостей безкоштовної версії, вимоги до ресурсів персонального комп'ютеру (ПК), відсутність реклами. Результати порівняння представлено в табл. 3.2.

Таблиця 3.2 – Порівняння середовищ для розробки

Критерій порівняння	PyCharm IDE	VSC (Visual Studio Code)	Wing IDE	Sublime Text
1	2	3	4	5
Вбудовані плагіни	++	++	+	—
Автодоповнення	++	++	++	+
Зручність роботи з терміналом	++	+	++	+
Підсвічування коду	++	++	++	++

Продовження таблиці 3.2

1	2	3	4	5
Можливості безкоштовної версії	+	+	++	+
Вимоги до ресурсів ПК	—	—	++	++
Відсутність реклами	++	+	++	—

За результатами порівняння середовищем для розробки було вибрано Wing IDE, спираючись його можливості (потужні функції редагування, аналізу коду, рефакторингу, відлагодження тощо) [25] та невимогливість до апаратних ресурсів ПК.

3.2.4 Вибір додаткових бібліотек та сервісів

При виборі додаткових бібліотек та сервісів враховувалася їхня підтримка обраної мови програмування, а також їхні потужності для вирішення поставленого завдання.

Перелік додаткових бібліотек, сервісів та їхній опис, обраних для впровадження окремих функцій, наведено в наступному розділі.

3.3 Вимоги до складу технічних та програмних засобів

Вимоги до складу технічних та програмних засобів наведено в технічному завданні (додаток А).

3.4 Висновки з розділу

На даному етапі роботи було виконано реалізацію попередньо визначених компонентів програми. Також здійснено обґрунтування у виборі структури програми та необхідних засобів розробки, зокрема, мови програмування, середовища, додаткових бібліотек та сервісів.

4 РОЗРОБКА ПРОГРАМИ

4.1 Основні рішення щодо реалізації класів

В результаті аналізу предметної області, описаного в попередніх розділах було виділено ряд самостійних сутностей: «Зображення» («Image»), «Оброблене зображення» («ProcessedImage»), «Відео» («Video»), «Робоча область» («App»), «Повідомлення про помилку» («Error»), «Інформація для користувача» («Info»), «Сповідження» («Report»), «Клас-помічник» («Helper»).

Кожну з перелічених сутностей реалізовано у вигляді окремого класу. Таким чином, розроблена програма реалізована згідно до ідеології об'єктно-орієнтованого програмування (ООП) – усі елементи є сукупністю взаємодіючих об'єктів, кожний з яких є екземпляром певного класу.

Структуру класів програми, їхні методи та атрибути, представлено на рис. 4.1.

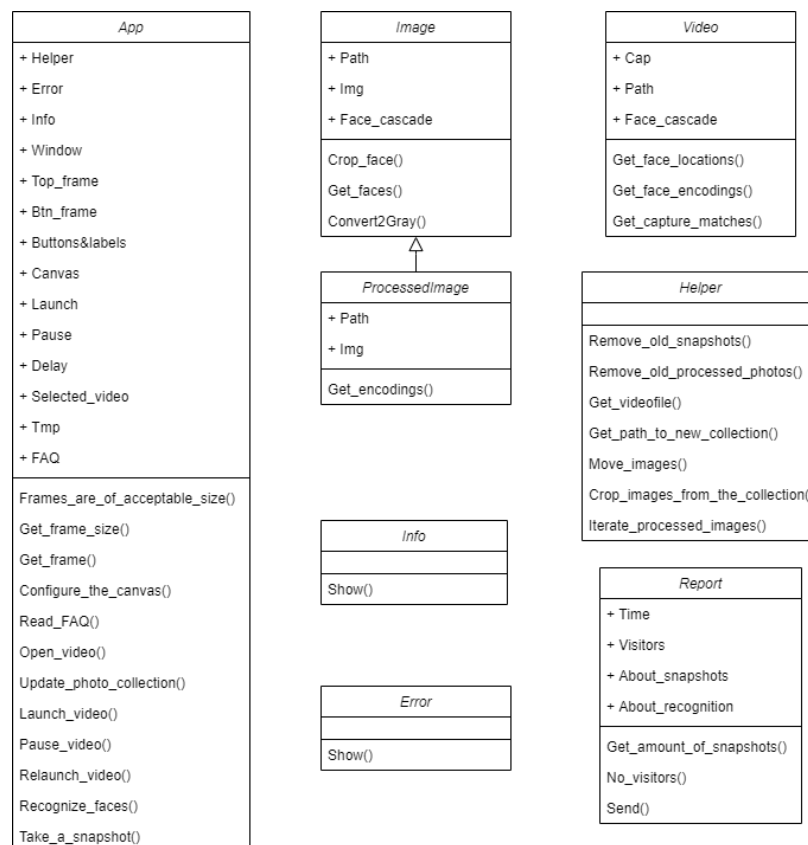


Рисунок 4.1 – Структура класів програми

В підрозділах 4.1.1 – 4.1.7 кожен з наведених на схемі класів програми описаний більш детально.

4.1.1 Класи Info, Error

Дані класи не мають жодних атрибутів, вони призначені для виведення інформаційного повідомлення або сповіщення про помилку у вигляді спливаючого вікна. Їхні функції, що здійснюють виклик методів `showerror` та `showinfo` з Tkinter-класу `messagebox`, виділено в окремий файл, щоб не захащувати код інших класів, наприклад, `App`, `Video` тощо.

4.1.2 Клас Image

Конструктор класу `Image`, що втілює сутність «Зображення», отримує в якості параметру повний шлях до файлу із зображенням, далі виділяє ім'я файлу, та за допомогою OpenCV-методу `imread` отримує графічне представлення зображення. Методи цього класу реалізують отримання фрагментів з обличчями із вихідного зображення, їхнє обрізування та переведення в градації сірого.

Як зазначалося раніше, спочатку здійснюється перетворення зображення в градації сірого. В OpenCV воно реалізується за допомогою функції `cvtColor`, яка в якості аргументів отримує вихідне зображення та параметр `COLOR_BGR2GRAY`, що вказує на характер трансформації.

Виявлення облич на перетвореному зображенні виконує метод `detectMultiScale`. Його аргументи:

- вихідне зображення `gray`;
- масштабний коефіцієнт для зображення `scaleFactor`, що визначає, на скільки збільшується вікно пошуку при послідовних скануваннях;
- мінімальна кількість суміжних прямокутних фрагментів `minNeighbours` (за замовчуванням дорівнює 3 [26]), що є метою виявлення.

Емпіричним шляхом було встановлено, що найбільш оптимальними є значення аргументів `scaleFactor = 1.1`, `minNeighbours = 4`. Інші параметри методу `detectMultiScale` не вказувалися явно через неактуальність або застарілість.

Таким чином, результатом виклику `detectMultiScale` є виявлені обличчя, що представляються прямокутними ділянками зі збереженням координат та розмірів. Обрізування фрагментів облич з цих ділянок можна здійснити, перевизначивши значення висоти та ширини за формулою (4.1):

$$\text{img} = \text{img} [y : y + h, x : x + w], \quad (4.1)$$

де *img* – зображення;

x, y – координати ділянки;

w, h – значення ширини та висоти відповідно.

В Python така операція реалізована з допомогою механізму зрізів.

Остаточним результатом обробки вихідного зображення методами класу `Image` є фрагмент-обличчя в градаціях сірого.

4.1.3 Клас `ProcessedImage`

Як показано на рис. 4.1, клас `ProcessedImage` є класом-нащадком від класу `Image`, бо його функціонал спрямований на роботу зі заздалегідь обробленим зображенням.

Єдиний метод, який він додає, називається `get_encodings`, параметром для якого є успадкований атрибут-зображення, а результатом – масив кодувань для обличчя у вигляді багатомірного вектора. Дані кодування отримуються завдяки методу `face_encodings` далі описаної бібліотеки `face_recognition`.

4.1.4 Клас Video

Клас Video, хоча і опрацьовує відео, а не зображення, але його функціонал є дещо схожим до вищерозглянутих класів Image та ProcessedImage. Це пояснюється тим, що відео розглядається алгоритмами КЗ як потік кадрів.

Конструктор класу в якості параметру отримує шлях до відеофайлу та за допомогою OpenCV-методу VideoCapture дозволяє захоплювати кадри відео.

На відміну від попередніх класів, в Video для отримання кодувань облич використано додатковий параметр locations, що є масивом областей облич на кадрі. Масив locations для кожного кадру отримується за допомогою методу face_locations класу face_recognition.

Одним з найважливіших методів класу і програми взагалі є get_capture_matches, який повертає масив визначених відвідувачів, яких розпізнано на даному відео. Його логіка наступна:

- для кожного вдало зчитаного відеокадру отримуються його локації обличчя;
- якщо такі локації існують, отримуються кодування цих облич;
- далі для кожного виявленого обличчя з колекції зображень відбувається порівняння його кодування з кодуваннями облич відеокадру. Співпадіння засвідчують, що визначеного користувача розпізнано, отже, інформацію про це можна додати у фінальне Telegram-сповіщення.

Останній крок описаної логіки реалізовано з використанням методу compare_faces класу face_recognition.

4.1.5 Клас Report

Клас Report, що відповідає сутності сповіщення в Телеграм має схоже призначення до класів Info та Error, але його структура є більш складною, бо

вміст сповіщення формується в залежності від результатів РО та того, чи були створені скріншоти.

Для реалізації механізму відправлень повідомлень в месенджер Telegram створено користувацький чат за допомогою сервісу BotFather. Цей же сервіс надає токен доступу до HyperText Transfer Protocol (HTTP) API. Код до акаунта користувача, для якого перевіряється надходження сповіщень програми, отримано іншим Telegram-сервісом IDBot.

Оскільки і токен, і код до акаунта не рекомендується зберігати з використанням СКВ, їх винесено як константи в окремий файл. Цей файл додано в .gitignore як «секретний».

4.1.6 Клас App

Клас App, що відповідає сутності «Робоча область» містить усі необхідні методи та атрибути для візуального представлення вікна програми: відображення фреймів кнопок та полотна для відео, їхні налаштування, виклики основних функцій програми, запуск/зупинка/відновлення обраного відео тощо.

Усі методи цього класу наведено в таблиці 4.1.

Таблиця 4.1 – Методи класу App

Метод	Опис
frames_are_of_acceptable_size	Перевірка роздільної здатності відеофайлу
get_frame_size	Отримання роздільної здатності відеофайлу
configure_the_canvas	Налаштування розмірів полотна під роздільну здатність відеофайлу
get_frame	Зчитування відеокадру
open_video	Вибір відеофайлу

Продовження таблиці 4.1

Метод	Опис
update_photo_collection	Оновлення колекції зображень в один клік
launch_video	Запуск відео
pause_video	Зупинка відео
relaunch_video	Відновлення програвання відео
take_a_snapshot	Створення знімку відеокадру
recognize_faces	РО на обраному відео
read_FAQ	Відкриття вікна-довідки

4.1.7 Клас Helper

Клас Helper, що також не має атрибутів, містить допоміжні методи, які налагоджують взаємодію між іншими об'єктами: отримання шляху до файлу, циклічний перебір зображень, видалення неактуальних скріншотів тощо.

Усі методи цього класу наведено в табл. 4.2.

Таблиця 4.2 – Методи класу Helper

Метод	Опис
Remove_old_snapshots	Видалення неактуальних знімків відеокадрів
remove_old_processed_photos	Видалення неактуальних оброблених зображень
Get_videofile	Отримання відеофайлу через меню файлового діалогу
get_path_to_new_collection	Отримання шляху до директорії з колекцією зображень на накопичувачі
move_images	Переміщення зображень між папками

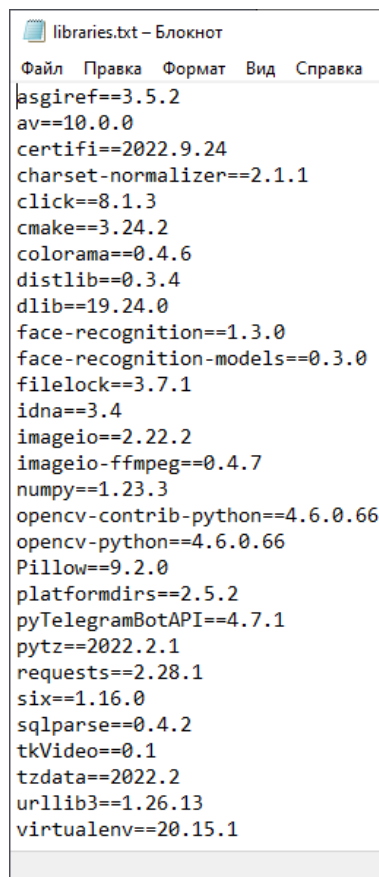
Продовження таблиці 4.2

Метод	Опис
crop_images_from_the_collection	Перебір вихідних зображень для попередньої обробки
iterate_processed_images	Перебір оброблених зображень для РО на відео

4.2 Основні рішення щодо використаних сервісів

В попередньому розділі було сформульовано ознаки вибору додаткових сервісів та бібліотек – підтримка мови розробки та потужності для вирішення завдання.

В попередньому розділі зазначалося, що усі ці бібліотеки записано в текстовий файл libraries.txt. Їх повний перелік представлено на рис. 4.2.



```

libraries.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
asgiref==3.5.2
av==10.0.0
certifi==2022.9.24
charset-normalizer==2.1.1
click==8.1.3
cmake==3.24.2
colorama==0.4.6
distlib==0.3.4
dlib==19.24.0
face-recognition==1.3.0
face-recognition-models==0.3.0
filelock==3.7.1
idna==3.4
imageio==2.22.2
imageio-ffmpeg==0.4.7
numpy==1.23.3
opencv-contrib-python==4.6.0.66
opencv-python==4.6.0.66
Pillow==9.2.0
platformdirs==2.5.2
pyTelegramBotAPI==4.7.1
pytz==2022.2.1
requests==2.28.1
six==1.16.0
sqlparse==0.4.2
tkVideo==0.1
tzdata==2022.2
urllib3==1.26.13
virtualenv==20.15.1

```

Рисунок 4.2 – Скріншот текстового файлу libraries.txt

Частину бібліотек встановлено за допомогою рір неявно, як складові, дочірні пакети інших сервісів.

Для впровадження методів ML обрано Open Source Computer Vision Library (OpenCV), яка містить понад 2500 алгоритмів КЗ, серед них – виявлення та РО [27]. Застосування чисельних алгоритмів для обробки зображень було наведено в підрозділі 4.1.

При виборі системи обміну повідомлення (месенджеру), яка використовувалася б користувачами для прийому повідомлень, обрано Telegram, ключовими перевагами якої є безкоштовність, висока швидкість та безпека даних, а також унікальність функціоналу, що передусім передбачає можливість самостійного програмування ботів [28].

Повний перелік бібліотек та сервісів встановлених явно, їхній опис та деталі впровадження в ПЗ представлено в табл. 4.3.

Таблиця 4.3 – Додаткові бібліотеки та сервіси

Бібліотека	Версія	Опис	Використання в розробці застосунку
Python Imaging Library (PIL)	9.2.0	Робота з растровою графікою, зокрема, для швидкого доступу до даних, що зберігаються в основних піксельних форматах [29]	Представлення відеокадрів в графічному блоці Canvas
OpenCV	4.6.0.66	Можливості швидкого створення складних застосунків для КЗ [30]	Отримання кадрів з відеофайлу, переведення зображень в градації сірого, тренування нейромережі тощо
dlib	19.24.0	Програмні компоненти для обробки зображень	Підтримка компонент для обробки зображень, зокрема, бібліотеки face_recognition
face_recognition	1.3.0	Розпізнавання та маніпулювання обличчями з зображень [31]	Завантаження зображень в змінні, отримання кодувань зображень та їхнє порівняння між собою

Продовження таблиці 4.3

Бібліотека	Версія	Опис	Використання в розробці застосунку
haarcascade_ _fronta lface_default .xml	1.0	Розпізнавання фронтальних зображень облич [32]	Методи РО
os, shututil		Високорівнева робота з файлами та каталогами, маніпулювання шляхами [33]	Перебір файлів в колекції зображень, отримання переліків файлів з директорій тощо
pyTelegram BotAPI	4.7.1	Заснований на HTTP інтерфейс, призначений для розробки Telegram-ботів [34]	Відправка сповіщення користувачеві в месенджері Telegram
subprocess		Породження нових процесів, підключення до потоків введення/виведення, отримання кодів повернення [35]	Перевірка, чи вставлено флешку в ПК

4.3 Основні рішення щодо даних

В даній роботі було прийнято рішення зберігати зразки зображень в каталогах, тобто, аналогічно до систем зі статичними БД, в яких здійснюється розпізнавання заздалегідь визначених осіб.

Натомість в системах з динамічними БД, дані отримані з відеофайлів, постійно звіряються, класифікуються та додаються до сховища, що є більш актуальним для банківських установ, а не приватних будинків.

Для користувачів передбачена можливість оновлювати поточну колекцію зображень «в один клік» – зображення завантажуються з директорії на накопичувачі до сховища.

Відеофайли також зберігаються в окремій директорії, як показано на схемі в попередньому розділі. Таким чином, датасет програми представляє собою вміст двох директорій: videos і photos.

Типи даних відео та зображень представлено в табл. 4.4.

Таблиця 4.4 – Типи даних відео та зображень

Дані	Тип	Роль типу даних в програмі
Зображення	*.jpg, *.png, *.gif	Програма перевіряє тип файлів в директорії для зображень; для усіх відповідних файлів виконується обробка
Відео	*.mp4, *.avi, *.wmv	Відображається у меню файлового діалогу; користувач може обирати, відео якого типу він розглядає

Для РО на відео вибрано фотографії та відеофайли кількох злочинців та правопорушників – зазвичай саме такі особи стають заздалегідь визначеними відвідувачами в системах відеоспостереження. Для збільшення обсягів колекції було додано декілька зображень знаменитостей із Large Age-Gap (LAG) датасету [36].

Створені користувачем скріншоти, як зазначалося у розділі 2, немає сенсу зберігати сумісно з датасетами, тому для них створена окрема директорія.

Важливим питанням є організація роботи з тимчасовими даними типу оброблених зображень та скріншотів – оскільки ці дані генеруються при кожному запуску програми та майже одразу втрачають свою актуальність для майбутнього використання, є ризики переповнення простору диску. Щоб уникнути цього, реалізовано функції очищення директорій зі скріншотами та обробленими зображеннями після вибору користувачем нового відеофайлу для аналізу. Ці функції представлено в класі Helper у підрозділі 4.1.

4.4 Основні рішення щодо користувацького інтерфейсу

При розробці GUI враховувався досвід користувачів в предметній області та їхні очікування, що формувалися при використанні раніше розглянутих аналогів.

Окрім цього, важливою метою було досягнути простоти, щоб вікно застосунку не захащувалося зайвими елементами, такими як смуги прокрутки, написи, прапорці тощо, розсіюють увагу користувача.

Найбільш доцільним є розділення робочого вікна на два умовні фрейми (блоки): для кнопок та для відеозапису. При запуску вікна програми (рис. 4.3) другий блок порожній, оскільки користувач може відкрити застосунок тільки для читання довідки програми або оновлення колекції зображень.

Обробка подій при натисканні будь-якої з кнопок спроектована так, щоб потрібна користувачеві операція виконувалася в один клік. Для блоку кнопок вибрано світло-зелений колір, кольори та шрифти інших елементів GUI залишено за замовчуванням. Таким чином, вікно застосунку справляє консервативне враження.

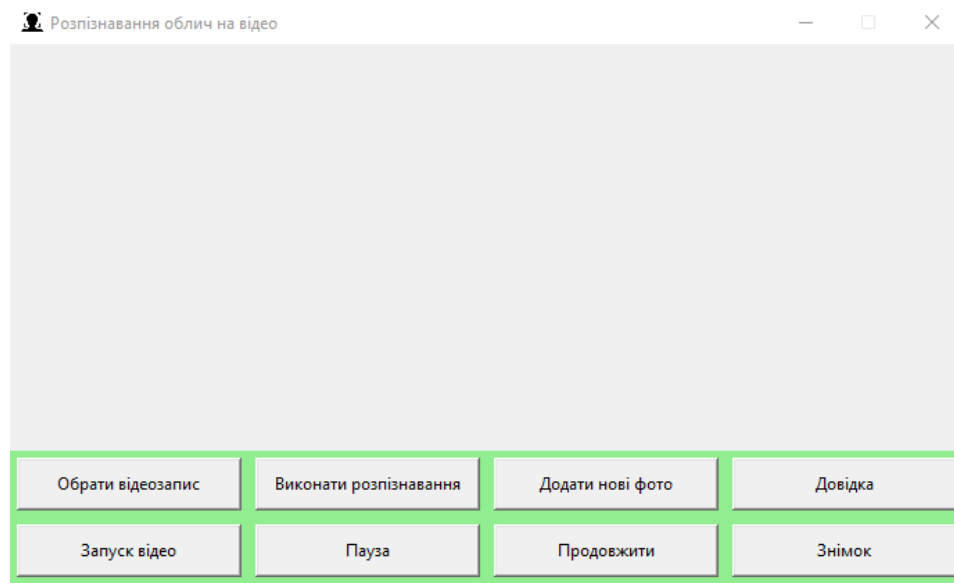


Рисунок 4.3 – Стартове вікно програми

Якщо ж користувач обирає відеофайл для відтворення та аналізу, розмір блоку для відео формується з урахуванням широти та висоти кадру, як представлено на рис. 4.4.

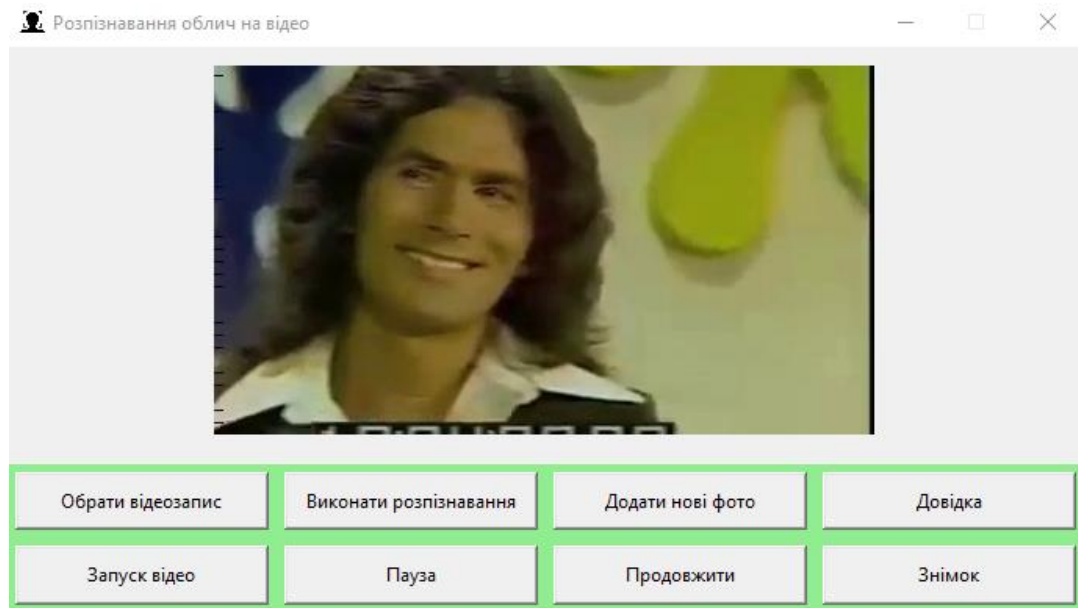


Рисунок 4.4 – Вікно програми після завантаження відеофайлу

Розділ «Довідка» виділений в окреме вікно (рис. 4.5), що містить декілька кроків, представлених різнокольоровими блоками з написами.

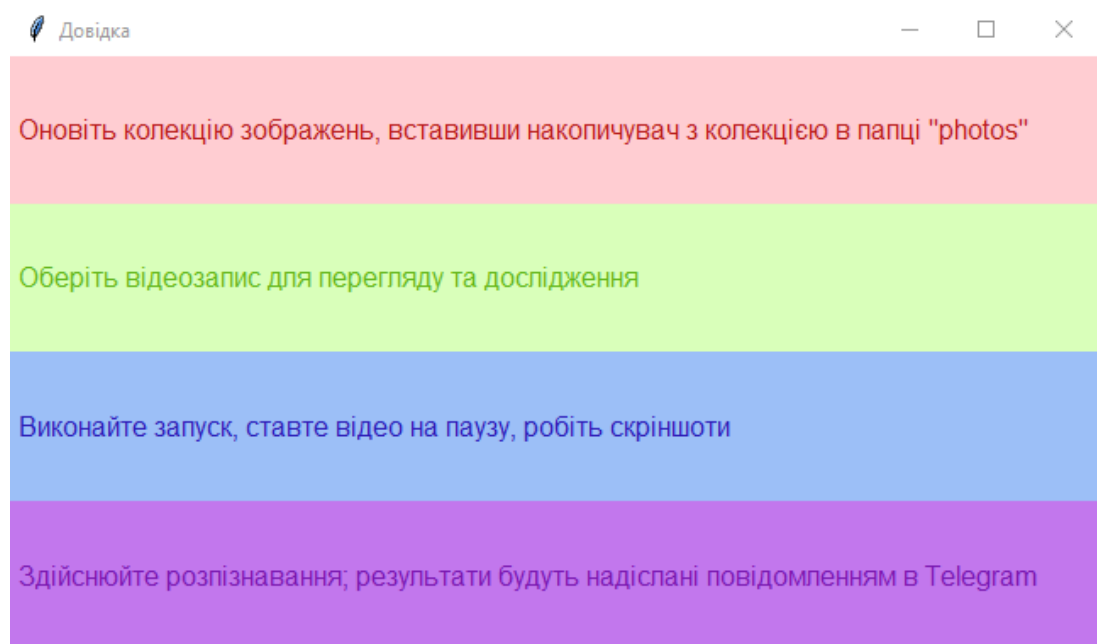


Рисунок 4.5 – Вікно «Довідка»

Згідно вимог до програми передбачена обробка виняткових ситуацій: користувач отримує діалогове вікно про помилку, наприклад, при натисканні кнопки «Запуск відео» перед вибором відеофайлу (рис. 4.6).

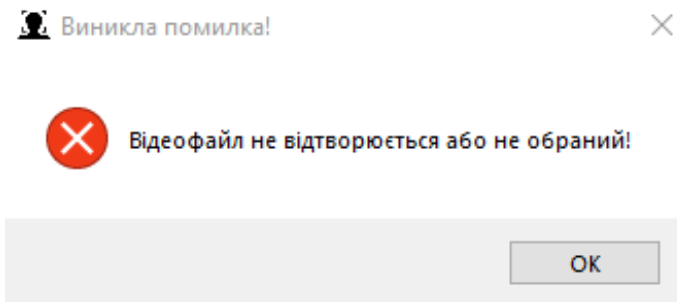


Рисунок 4.6 – Приклад діалогового вікна про помилку

Кінцевий результат роботи програми у вигляді сповіщень користувачеві в месенджері Telegram представлено на рис. 4.7.

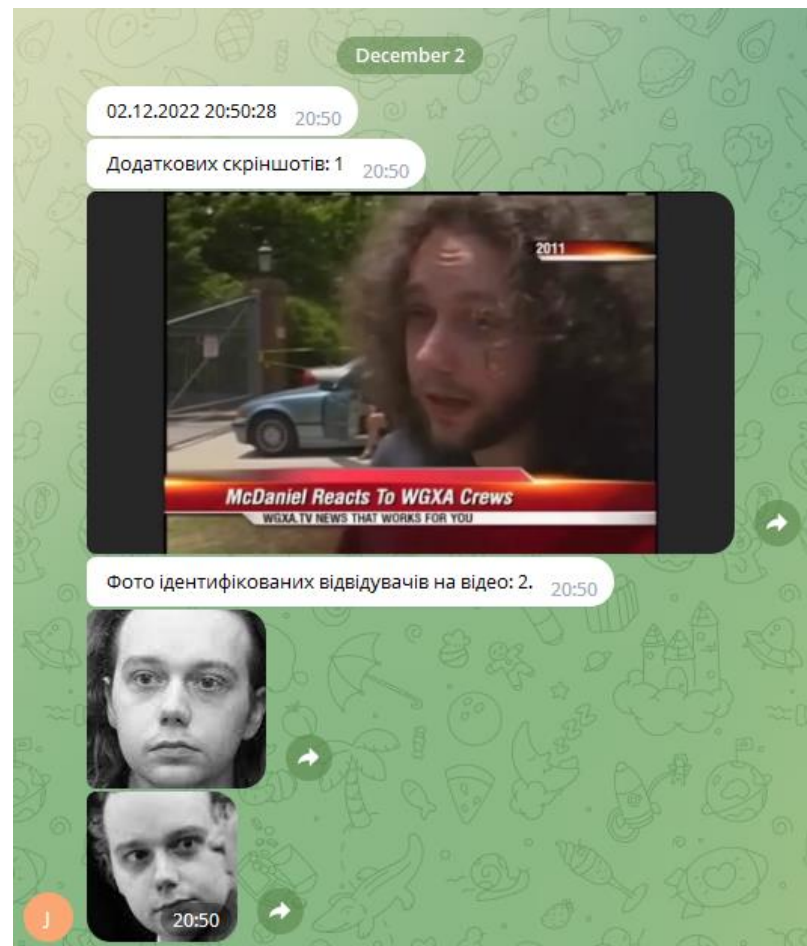


Рисунок 4.7 – Сповіщення про результати РО на відео

В представленому випадку користувач при перегляді відео самостійно зробив один знімок, а РО на відео виявило, що два фото ідентифікованої

особи містяться в колекції зображень. Також представлено час і дату виконання РО.

4.5 Висновки з розділу

На даному етапі роботи описано етапи розробки окремих компонентів програми, проаналізовано реалізацію кожного елементу архітектури програми: рішення щодо розробки класів, використаних сервісів, користувацького інтерфейсу, роботи з даними.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ, ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Експлуатація програми

Опис умов застосування та експлуатації програми, зокрема, призначення, характеристики, переліки вхідних та вихідних даних наведено в «Керівництві програміста» (додаток Б).

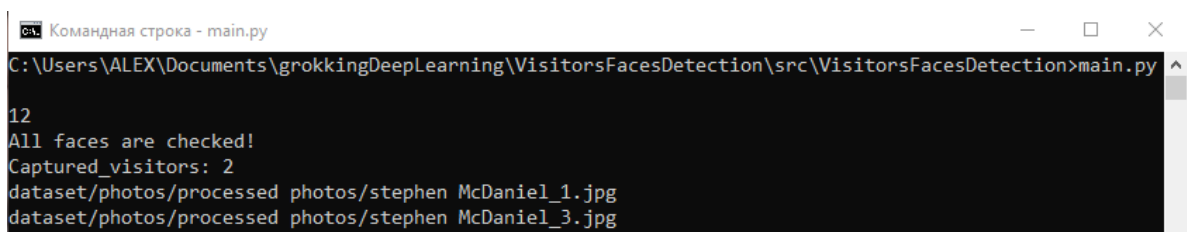
5.2 Методика та результати тестування

5.2.1 Загальний підхід до тестування

Тестування програми є невід’ємним етапом в процесі розробки, адже ПЗ, яке працює некоректно, може призвести до низки проблем.

З урахуванням того, що розроблюваний застосунок не призначено для використання в промислових масштабах, прийнято рішення зосередитись саме на мануальному (ручному) тестуванні.

За ознакою знання системи в ході розробки застосовувалося тестування білого ящика – гілка програмного коду, що реалізує окремий сценарій дій користувача, доповнювалася командами виведення повідомлень в консоль. Таким чином, перевіряється, наприклад, чи правильно розпізнано обличчя на відео (рис. 5.1).



```
Командная строка - main.py
C:\Users\ALEX\Documents\grokkingDeepLearning\VisitorsFacesDetection\src\VisitorsFacesDetection>main.py
12
All faces are checked!
Captured_visitors: 2
dataset/photos/processed photos/stephen McDaniel_1.jpg
dataset/photos/processed photos/stephen McDaniel_3.jpg
```

Рисунок 5.1 – Консольне повідомлення щодо проміжних результатів РО

За ознакою позитивності сценаріїв здійснювалося як позитивне, так і негативне тестування. Прикладом позитивного сценарію, коли перевіряється

нормальна поведінка системи [37], є оновлення колекції зображень, коли в роз'єм ПК вставлено накопичувач з папкою «photos», що містить нову версію колекції. Прикладом негативного сценарію є спроба відтворити або поставити на паузу відео при умові, що відеофайл не обрано. Перевірка даних сценаріїв дозволяє додавати обробку виняткових ситуацій та попереджати потенційні помилки.

5.2.2 Загальний чекліст

Аби процес тестування не був хаотичним, суто інтуїтивним та, як наслідок, неповним, усі перевірки заздалегідь внесено в загальний чекліст (рис. 5.2). Чекліст дозволяє пам'ятати все, що необхідно протестувати, фіксувати результати тестування та відслідковувати статистику про статус розробки ПЗ. Ще однією неочевидною перевагою використання чеклісту є можливість заощаджувати час на розробку високо деталізованих тест-кейсів [38].

Найпростішою формою чек-лісту є невпорядкований перелік ідей щодо тестування. Її розширенням є впорядкування цього переліку, розбиття функціоналу, що тестується, за пріоритетом:

- базові функції ПЗ, які є предметом дослідження для «димового тестування» [39]. Якщо їхнє тестування фахівцем з якості чи навіть програмістом виявляє наявність дефектів, то версія ПЗ повертається на переробку;
- функції, затребувані більшістю користувачів ПЗ – предмет для проведення тестування критичного шляху [39];
- малозначущі функції, дефекти в яких не є критичними для роботи користувача з ПЗ.

Вдосконаленням описаного підходу до складання чеклістів є створення багаторівневого списку. Його сутність полягає в тому, що ідеї чеклісту представляють ієрархічно, наприклад, для узагальненої ідеї

«Створення скріншоту» (рис. 5.2) доцільно виділити більш конкретизовані перевірки:

- чи відбувається програвання відеозапису (а інакше як робити знімок кадру);
- чи збережено створений знімок в директорію (в розділі 2 описано, що скріншоти є додатками до повідомлення для користувачів).

Розпізнавання облич на відео		
№	Перевірка	Статус
1	Вибір відеофайлу	Пройдено успішно
	Перевірка роздільної здатності відео	Пройдено успішно
	Перевірка чи відбувається програвання відеозапису	Пройдено успішно
2	Перегляд довідки програми	Пройдено успішно
3	Оновлення колекції зображень	Пройдено успішно
	Перевірка наявності накопичувача з папкою-колекцією	Пройдено успішно
	Копіювання колекції з накопичувача в директорію програми	Пройдено успішно
4	Запуск відеофайлу	Пройдено успішно
	Зчитування відеокадрів	Пройдено успішно
	Циклічне розміщення кадрів на полотні вікна	Пройдено успішно
5	Зупинка відеофайлу	Пройдено успішно
6	Відновлення програвання відеофайлу	Пройдено успішно
7	Створення скріншоту	Пройдено успішно
	Перевірка чи відбувається програвання відеозапису	Пройдено успішно
	Запис скріншоту в директорію	Пройдено успішно
8	Розпізнавання облич на відео	Пройдено успішно
	Видалення оброблених фото застарілої колекції	Пройдено успішно
	Виокремлення облич обрізуванням зображень колекції	Пройдено успішно
	Переведення обрізаних фото в градації сірого	Пройдено успішно
	Отримання кодувань облич для оброблених фото	Пройдено успішно
	Співставлення кодувань зображень та відеокадрів	Пройдено успішно
9	Сповіщення про результати	Пройдено успішно
	Підрахунок виявлених осіб	Пройдено успішно
	Формування повідомлення	Пройдено успішно
	Відправка повідомлення в Телеграм	Пройдено успішно

Рисунок 5.2 – Загальний чекліст застосунку

Таким чином, отримано логічний і структурований чекліст, на основі якого можна проводити тест-дизайн.

5.2.3 Тест-дизайн

Відповідно до визначених цілей та ідей щодо тестування, проведено тест-дизайн – проєктування тест-кейсів, кожен із яких представляє окремий тестовий випадок. Основними техніками тест-дизайну, що застосовувалися стали «Причина/наслідок» і «Передбачення помилки».

Техніка «Причина/наслідок» заснована на отриманні відповіді від системи на основі виконання ряду умов. Наприклад, зміст повідомлень, що отримує користувач в месенджері Telegram, повністю залежатиме від результатів РО на відео. Тобто, вхідною комбінацією умов є вибране відео, зміст колекції зображень та знімки відеокадрів, створені вручну.

Техніка «Передбачення помилки» полягає у «передбаченні» вхідних умов, які призводять до виникнення тих чи інших дефектів в роботі ПЗ. Основним джерелом для їхнього пошуку є вимоги до застосунку. Інтерпретуючи ці вимоги, можна отримувати ситуації, що потенційно спричинятимуть як дефекти, так і збої в роботі ПЗ. Приклади такої інтерпретації наведено в табл. 5.1.

Таблиця 5.1 – Приклади застосування техніки «Передбачення помилки»

Вимога	Потенційна небезпека
Максимально допустима роздільна здатність відео становить 860X480	Користувач обирає з відеоколекції файл з розширенням 1024X600
Нова колекція зображень копіюється з директорії накопичувача до директорії датасету на диску	Користувач видалив директорію-датасет на диску та здійснює команду «Додати нові фото»
Створені користувачем знімки записуються в окрему директорію	Користувач в ході роботи програми вручну видалив існуючі знімки разом з директорію

Приклади розроблених тест-кейсів представлено в табл. 5.2. Як раніше зазначалося, обраний тип тестування, а, отже, і тип тест-кейсів – мануальний. Передумовою для усіх тест-кейсів є виконання успішного запуску програми.

Таблиця 5.2 – Приклади розроблених тест-кейсів

Опис	Кроки	Очікувані результати	Пріоритет
Перевірка перегляду довідки	Натиснути кнопку «Довідка»	Відкривається вікно «Довідка»	Середній
Перевірка Telegram-повідомлення для РО на відео, що не містить кадрів з обличчями	1. Натиснути кнопку «Обрати відеозапис»	1. Відкривається меню файлового діалогу	Високий
	2. Обрати відео з колекції	2. Розміри вікна програми змінюються під розмір відеокadrів	
	3. Натиснути кнопку «Виконати розпізнавання»	3. Користувач отримує Telegram-повідомлення: дата і час РО; «Окремих скріншотів для відео не створено»; «Визначених відвідувачів з колекції зображень на відео не зафіксовано».	

5.2.4 Результати тестування

Результати тестування представлено в якості статусу проходження пунктів загального чеклісту (рис. 5.2). Отримані результати засвідчують, що розроблене ПЗ відповідає заявленим вимогам, зокрема, не містить очевидних помилок та задовольняє загальному визначенню якості для аналогічних десктопних застосунків.

5.3 Висновки з розділу

На даному етапі роботи було розроблено «Керівництво програміста», в якому, зокрема, описано експлуатацію програми. Також здійснено ручне тестування за спроектованими тест-кейсами згідно зі заздалегідь сформованим чеклістом.

ВИСНОВКИ

В ході виконання роботи було проаналізовано предметну область, описано існуючі матеріали та методи, спроектовано архітектуру системи, реалізовано її компоненти та проведено тестування розробленої програми. Також оформлено програмні документи «Технічне завдання» та «Керівництво програміста».

На першому етапі було виконано аналіз проблеми та постановку завдання до роботи, описано об'єкти і процеси предметної області, готові рішення та методики, їхні переваги і недоліки. На другому етапі було сформульовано та викладено власні ідеї щодо розробки та удосконалення існуючих моделей. Наступним кроком стала розробка архітектури програми, в ході якої була обґрунтована її структура, обрані мова програмування, середовище розробки, бібліотеки та сервіси тощо.

Далі було складено «Технічне завдання», що стало основою для реалізації компонентів програми. Розроблену програму було протестовано на основі заздалегідь складених тестових артефактів – чеклісту та тест-кейсів. Після цього було складено «Керівництво програміста».

Підсумком роботи є програмний продукт, що виконує РО на відео, ідентифікує визначених відвідувачів у колекції зображень та надсилає результати користувачеві застосунку в месенджер Telegram.

За потребою розроблена програма може бути модифікованою та слугувати основою для більш складних розробок систем аналізу відеоспостережень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Технологія виявлення та розпізнавання обличчя: в чому різниця? [Електрон. ресурс]. – Режим доступу: <https://worldvision.com.ua/tehnologiya-obnaruzheniya-i-raspoznavaniya-litsa-v-chem-raznitsa/>
2. Лекція_3_теорія_розпізнавання [Електрон. ресурс]. – Режим доступу: https://sites.google.com/site/navcalnijsajtdlastudentivvnz/teoreticni-osnovi-informatiki/lekcii-17-18/lekcia_teorija_rozpiznavanna
3. Ming-Hsuan Yang. Detecting Faces in Images: A Survey / Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja // Transactions on Pattern Analysis and Machine Intelligence. – California: IEEE, January 2002 : proceedings. – Vol. 24, NO. 1. – P. 34.
4. Stan Z. Li Handbook of Face Recognition / Z. Li Stan, K. Jain Anil. – New York : Springer, 2004. – 395 p.
5. Goh C. Y. Online map-matching based on Hidden Markov model for real-time traffic sensing applications / C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran and P. Jaillet. – Intelligent Transportation Systems: 15th International IEEE Conference, 2012. – pp. 776-781.
6. Viola, P. Robust Real-Time Face Detection / P. Viola, M.J. Jones // Technical Report Series. – Cambridge: IEEE, February 2001 : proceedings. – p. 137–154.
7. Примітиви Хаара [Електрон. ресурс]. – Режим доступу: <https://kostyakulakov.ru/wp-content/uploads/2015/07/5-300x225.jpg>
8. Lienhart R. An extended set of Haar-like features for rapid object detection / R. Lienhart, J. Maydt // International Conference on Image Processing. – New York: IEEE, February 2002 : proceedings. – Vol. 1. – p. 900–903.
9. Мурыгин К.В. Особенности реализации алгоритма AdaBoost для обнаружения объектов на изображениях / К.В. Мурыгин // Институт проблем искусственного интеллекта МОН Украины и НАН Украины. Штучний інтелект. – Донецьк, 2009. – №3. – с. 573-581.

10. Jones J.P. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. / J.P. Jones, L.A. Palmer. – Journal of Neurophysiology, 1987. – pp. 1233-1258.
11. Lades M. Distortion invariant object recognition in the dynamic link architecture / [M. Lades et al.] // IEEE Transactions on Computers, March 1993. – Vol. 42, No. 3. – pp. 300-311. – doi: 10.1109/12.210173.
12. Опря, А.Т. Статистика (модульний варіант з програмованою формою контролю знань): навч. посіб. / А.Т. Опря. – К.: Центр учбової літератури, 2012. – 448 с.
13. Muresan D. D. Adaptive Principal Components and Image Denoising / D. D. Muresan, T. W. Parks // IEEE International Conference on Image Processing. – Arlington: IEEE, September 2003 : proceedings. – Vol. 1. – p. 101-104.
14. Salton do Prado K. Face Recognition: Understanding LBPH Algorithm. – Towards Data Science, November 2017 [Електрон. ресурс]. – Режим доступу: <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>
15. Turk, Matthew A; Pentland, Alex P. Face recognition using eigenfaces / Matthew A. Turk, Alex P. Pentland // IEEE Conference on Computer Vision and Pattern Recognition. – Massachusetts: IEEE, 1991: proceedings. – pp. 586–591.
16. Brunelli R. Face recognition through geometrical features / R. Brunelli, T. Poggio. – Berlin/Heidelberg: Springer, 1992. – pp. 792-800.
17. Lowe, David G. Object recognition from local scale-invariant features. // IEEE Conference on Computer Vision 2. – Vancouver: IEEE, 1999: proceedings. – pp. 1150–1157. – doi:10.1109/ICCV.1999.790410.
18. Форма «Use filter» [Електрон. ресурс]. – Режим доступу: https://zoneminder.readthedocs.io/en/latest/_images/filter-filterview.png
19. Форма «Add Video Source» [Електрон. ресурс]. – Режим доступу: <https://www.netcamstudio.com/assets/images/Connection.png>
20. Lauren Giddings' killer Stephen McDaniel to appear in court | Macon

Telegraph [Електрон. ресурс]. – Режим доступу: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSHLBeY4PK_Z5uZP2W0u7xpH5qmXDAM1dzzLw&usqp=CAU

21. Color conversions [Електрон. ресурс]. – Режим доступу: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray

22. face_recognition package [Електрон. ресурс]. – Режим доступу: https://face-recognition.readthedocs.io/en/latest/face_recognition.html

23. Ерік Маттес Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування [Текст] / Ерік Маттес; перекл. з англ. Ольги Белової. – Львів : Видавництво Старого Лева, 2021. – 600 с.

24. Graphical User Interfaces with Tk [Електрон. ресурс]. – Режим доступу: <https://docs.python.org/3/library/tk.html>

25. Wing Pro – Version 9.0.0 – Released 2022-11-04 [Електрон. ресурс]. – Режим доступу: <https://wingware.com/downloads/wing-pro>

26. cv::CascadeClassifier Class Reference [Електрон. ресурс]. – Режим доступу: https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html

27. Open CV – About [Електрон. ресурс]. – Режим доступу: <https://opencv.org/about/>

28. Soffar H. What are the advantages and disadvantages of Telegram? / H. Soffar // Online Sciences, May 2015 [Електрон. ресурс]. – Режим доступу: <https://www.online-sciences.com/technology/what-are-the-advantages-and-disadvantages-of-telegram/>

29. Pillow (PIL Fork) 9.3.0 documentation. Overview [Електрон. ресурс]. – Режим доступу: <https://pillow.readthedocs.io/en/stable/handbook/overview.html>

30. Bradski G. Learning OpenCV / G. Bradski, A. Kaehler. – М.: O'Reilly Media, 2008. – 556 с.

31. face-recognition [Електрон. ресурс]. – Режим доступу: <https://pypi.org/project/face-recognition/>

32. haarcascade_frontalface_default.xml [Електрон. ресурс]. – Режим доступу: https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
33. os – Miscellaneous operating system interfaces [Електрон. ресурс]. – Режим доступу: <https://docs.python.org/3.7/library/os.html>
34. Telegram Bot API [Електрон. ресурс]. – Режим доступу: <https://core.telegram.org/bots/api>
35. subprocess – Subprocess management [Електрон. ресурс]. – Режим доступу: <https://docs.python.org/3/library/subprocess.html>
36. S. Bianco Large Age-Gap Face Verification by Feature Injection in Deep Networks In Pattern Recognition Letters. – volume 90. – pp. 36-42, 2017. – [Електрон. ресурс]. – Режим доступу: <http://www.lvl.disco.unimib.it/activities/large-age-gap-face-verification/>
37. Позитивне та негативне тестування [Електрон. ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/positive-negative-testing/>
38. Типи тестування: пам'ятка початківця II [Електрон. ресурс]. – Режим доступу: <https://training.epam.ua/News/Items/502?lang=ua>
39. Что такое чек-лист? [Електрон. ресурс]. – Режим доступу: <https://sitechco.ru/2011/08/chto-takoe-chek-list/>

ДОДАТОК А
Технічне завдання

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломний проєкт на тему «Дослідження та програмна реалізація застосунку розпізнавання облич на відео», затверджене наказом по НУ «Запорізька політехніка» № 363 від 4 листопада 2022 року.

A.2 Призначення розробки

Дане ПЗ призначено для використання в системах індивідуальної безпеки.

Системи для РО на відео та сповіщення користувачів про його результати засобами соцмереж зазвичай є комерційними та громіздкими за функціоналом.

Використовуючи розроблене ПЗ, користувачі зможуть переглядати і швидко аналізувати відеозаписи, здійснені камерами відеоспостереження, фіксувати окремі знімки та отримувати сповіщення про результати аналізу в месенджері Telegram.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- реалізація графічного інтерфейса користувача;
- реалізація можливості вибору відеофайлу для роботи;
- реалізація можливості перегляду, зупинки для створення скріншотів та відновлення перегляду відео;
- реалізація можливості оновлення колекції зображень в один клік;
- реалізація сторінки з довідковою інформацією;

- реалізація РО на відео та надсилання сповіщення користувачеві в месенджер Telegram згідно його результатів.

А.3.2 Вимоги надійності

В програмі обов'язкова реалізація наступних вимог по надійності:

- захист від спроби вибору відеофайлів некоректних форматів (допустимими є формати *.mp4, *.wmv, *.avi) та з некоректною роздільною здатністю (максимально допустима складає 860x480);
- захист від спроб роботи з відеофайлом ще до його вибору через меню файлового діалогу;
- обробка виняткових ситуацій (наприклад, помилка зчитування відеокадру, отримування кодувань обличчя тощо);
- стабільне функціонування при безперервній роботі ПК.

А.3.3 Умови експлуатації

Умови експлуатації програми відповідають вимогам до експлуатації технічних засобів (комп'ютера, сервера тощо). Для використання програми не потребується додаткова кваліфікація користувача.

А.3.4 Вимоги до складу і параметрів технічних засобів

Вимоги до складу і параметрів технічних засобів:

- ОС Microsoft Windows 7 x32/ 10 x64;
- процесор із тактовою частотою 2.5 ГГц і вище;
- жорсткий диск місткістю не менше, ніж 100 МБ;
- мінімум 2 ГБ обсягу оперативної пам'яті (рекомендується 4 ГБ і більше);

- накопичувач з директорією photos для копіювання колекції зображень на диск;
- додаткове ПЗ: встановлені бібліотеки та сервіси dlib версії 19.24 і вище, face_recogniton, OpenCV версії 4.6 і вище, haarcascade_frontalface_default.xml, PIL версії 9.2 і вище, pyTelegramBotAPI версії 4.7 і вище.

A.3.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових операцій щодо маркування та упакування.

A.3.6 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання програми відповідають аналогічним вимогам носія, на якому вона записана.

A.3.7 Вимоги до програмної документації

Склад програмної документації:

- технічне завдання;
- керівництво програміста.

A.3.8 Стадії і етапи розробки

Стадії і етапи розробки програми:

- аналіз проблеми та постановки завдань;
- аналіз існуючих матеріалів та методів;
- реалізація;
- тестування.

А.3.9 Порядок контролю та приймання

Порядок контролю та приймання відповідає затвердженому порядку контролю та приймання дипломних кваліфікаційних робіт студентів денної та заочної форми навчання освітнього ступеня «магістр» НУ «Запорізька політехніка».

ДОДАТОК Б
Керівництво програміста

Б.1 Вимоги до програми

Дане ПЗ призначене для РО визначених відвідувачів в колекції зображень на відео та сповіщення користувачів про результати роботи в месенджері Telegram.

Б.2 Характеристики програми

Програма може працювати в декількох режимах:

- перегляд відео та створення скріншотів;
- оновлення колекції зображень;
- вивчення довідки;
- РО на відео.

Програма написана з використанням мови програмування Python в Wing IDE, для розробки графічного інтерфейсу використано бібліотеку Tkinter, для роботи з зображеннями та відеофайлами – dlib, face_recogniton, OpenCV, haarcascade_frontalface_default.xml, PIL, pyTelegramBotAPI (версії бібліотек вказані в технічному завданні).

Каталог рішення займає 975 кілобайт, розмір розархівованої бібліотеки dlib – 198 мегабайт.

Перелік файлів, з яких складається програма:

а) файли, що зберігають змінні-константи:

1) constants.py – константи для значень кольорів, розмірів, файлових шляхів, написів, повідомлень, заголовків, файлових типів тощо;

2) telebot_constants.py – токен боту та код акаунта;

б) файл main.py, що ініціалізує та запускає вікно програми;

в) файли, що реалізують сутності програми:

1) Alerts.py – файл класів Error (вікно про помилку), Info (вікно з повідомленням), Report (повідомлення користувачеві в Телеграм);

2) App.py – файл з класом Tkinter-вікна програми;

- 3) Image.py – файл класу для сутності необробленого зображення;
- 4) ProcessedImage.py – файл класу для сутності обробленого зображення (з виокремленим обличчям відвідувача);
- 5) Video.py – файл класу для сутності обраного відеофайлу;
- 6) Helper.py – файл класу з допоміжними функціями перебору зображень, отримання шляху до файлу тощо;
- г) файл __init__.py, що об'єднує файли класів попереднього пункту в єдиний пакет;
- д) faces.xml – імпортована модель, що здійснює РО на оброблених зображеннях та відеокадрах.

Б.3 Звернення до програми

Звернення до програми здійснюється за допомогою запуску файлу main.py через командний рядок.

Б.4 Вхідні і вихідні дані

Вхідними даними програми є датасет, що складається з директорії відеофайлів videos, отриманих камерами спостереження та директорії зображень photos, що містять обличчя визначених відвідувачів. Окрім цього, до вхідних даних можна віднести колекцію зображень з директорії накопичувача photos (якщо користувач має намір оновити існуючу колекцію зображень).

Вихідними даними є скріншоти, зроблені користувачем (за бажанням) під час роботи з обраним відеофайлом, результати РО на відео та їхній текстовий опис, що надсилаються користувачеві повідомленнями в месенджер Telegram з вказанням дати та часу аналізу.

Б.5 Повідомлення

В ході роботи ПЗ може генерувати повідомлення та сповіщення, опис яких представлено в таблиці Б.1.

Таблиця Б.1 – Повідомлення та їхні описи

Повідомлення	Опис
Виникла помилка!	Заголовок спливаючого вікна про помилку
Відеофайл обрано, можна починати розпізнавання	Повідомлення після вдалого вибору та завантаження відео через меню файлового діалогу
Відеофайл не відтворюється або не обраний!	Помилка при спробі роботи з відео до вибору відеофайлу
Неможливо відтворити відеофайл!	Помилка при невдалій спробі зчитати та опрацювати відеокадр
Спочатку вставте накопичувач з папкою «photos»	Для оновлення колекції зображень необхідно, щоб в ПК було вставлено накопичувач з директорією «photos»
Роздільна здатність відео більша ніж 860X480!	Помилка при спробі опрацювати відеофайл з більшою роздільною здатністю
Визначених відвідувачів з колекції зображень на відео не зафіксовано.	Сповіщення в месенджер Telegram, якщо відвідувачів не розпізнано
Окремих скріншотів для відео не створено.	Сповіщення в месенджер Telegram, якщо скріншоти не додавалися

Сторінка «Довідка» реалізована у вигляді окремого вікна, яке містить кроки з інструкціями, представленими в таблиці Б.2.

Таблиця Б.2 – Записи в довідці

№ кроку	Текст
1	Оновіть колекцію зображень, вставивши накопичувач з колекцією в папці "photos"
2	Оберіть відеозапис для перегляду та дослідження
3	Виконайте запуск, ставте відео на паузу, робіть скріншоти
4	Здійснюйте розпізнавання; результати будуть надіслані повідомленням в Telegram

Як зазначалося раніше, усі ці текстові дані винесено в окремий файл як константи, щоб не захаращувати код програми.

ДОДАТОК В
Текст програми

В.1 Файл main.py

```
from package.classes.App import *
if __name__ == "__main__":
    App(Tk())
```

В.2 Файл constants.py

```
DISKS_CAPTIONS = "wmic logicaldisk get DriveType, caption"
PC_DISKS = ('C', 'D')

VIDEOS_FOLDER = "dataset/videos"
PHOTOS_FOLDER = "dataset/photos"
IMAGES_FOLDER_ON_USB_FLASH_DRIVE = 'photos'
FULL_PATH_TO_PHOTOS_FOLDER =
"C:/Users/ALEX/Documents/grokkingDeepLearning/VisitorsFacesDetection/src/VisitorsFacesDetection/dataset/ph
otos"

SNAPSHOTS_FOLDER = "snapshots"
PROCESSED_PHOTOS_FOLDER = "dataset/photos/processed photos"

VIDEOTYPES = (("MP4 files", "*.mp4"),
               ("WMV files", "*.wmv"),
               ("AVI files", "*.avi"))

IMAGE_ENDS_WITH = ('jpg', 'png', 'gif')

MAX_FRAME_WIDTH = 860
MAX_FRAME_HEIGHT = 480
WINDOW_BACKGROUND = "#f2dbff"
BTN_FRAME_COLOR = 'lightgreen'
ICON = "ico/icon.ico"
BTN_W = 20

WINDOW_TITLE = "Розпізнавання облич на відео"
ERROR_TITLE = "Виникла помилка!"

MAY_LAUNCH_MSG = "Відеофайл обрано, можна починати розпізнавання"
VIDEO_NOT_SELECTED_MSG = "Відеофайл не відтворюється або не обраний!"
CANT_LAUNCH_VIDEO_MSG = "Неможливо відтворити відеофайл!"
CANT_ADD_IMAGES_MSG = "Спочатку вставте накопичувач з папкою \"photos\";"
```

```

+ "\nТакож перевірте розташування цільової директорії для колекції"
CANT_DISPLAY_VIDEO_MSG = "Розмір відео занадто великий!"
NO_CAPTURED_VISITORS_MSG = "Визначених відвідувачів з колекції зображень на відео не
зафіксовано."
NO_SNAPSHOTS_MSG = "Окремих скріншотів для відео не створено."

FAQ_STEP1 = "Оновіть колекцію зображень, вставивши накопичувач з колекцією в папку
\"photos\""
FAQ_STEP2 = "Оберіть відеозапис для перегляду та дослідження"
FAQ_STEP3 = "Виконайте запуск, ставте відео на паузу, робіть скріншоти"
FAQ_STEP4 = "Здійснюйте розпізнавання; результати будуть надіслані повідомленням в
Telegram"

SELECT_VIDEO_LBL = "Обрати відеозапис"
RECOGNIZE_FACES_MSG = "Виконати розпізнавання"
ADD_NEW_IMAGES_LBL = "Додати нові фото"
FAQ_LBL = "Довідка"
LAUNCH_VIDEO_LBL = "Запуск відео"
PAUSE_VIDEO_LBL = "Пауза"
RELAUNCH_LBL = "Продовжити"
TAKE_A_SNAPSHOT_LBL = "Знімок"

FACE_RECOG_MODEL = 'model/faces.xml'

```

В.3 Файл telebot_constants.py

```

TOKEN = '5712011509:AAF8Yj6dKzvSu2dmN7KzCGyNU1xTTTjJsa4'
TG_USER = '521127116'

```

В.4 Файл __init__.py

```

NAME = 'app_package'

```

В.5 Файл Alerts.py

```

from tkinter import messagebox
from package.constants import *
from package.telebot_constants import *

```

```

import telebot, time, os

class Error:
    def show(self, message):
        messagebox.showerror(title = ERROR_TITLE, message = message)

class Info:
    def show(self, message):
        messagebox.showinfo(message = message)

class Report():
    def __init__(self, captured_visitors):
        self.time = time.strftime("%d.%m.%Y %H:%M:%S")
        self.visitors = captured_visitors

    def get_amount_of_snapshots(self):
        return len([entry for entry in os.listdir(SNAPSHOTS_FOLDER)\
                    if os.path.isfile(os.path.join(SNAPSHOTS_FOLDER, entry))])

    def no_visitors(self):
        if len(self.visitors) == 0:
            return True
        return False

    def send(self):
        bot = telebot.TeleBot(TOKEN)
        bot.send_message(TG_USER, self.time)
        if self.get_amount_of_snapshots() == 0:
            self.about_snapshots = NO_SNAPSHOTS_MSG
            bot.send_message(TG_USER, self.about_snapshots)
        else:
            self.about_snapshots = f"Додаткових скріншотів: {self.get_amount_of_snapshots()}"
            bot.send_message(TG_USER, self.about_snapshots)
            for (root, dirs, snapshots) in os.walk(SNAPSHOTS_FOLDER):
                for snapshot_name in snapshots:
                    path = os.path.join(root, snapshot_name).replace("\\", "/")
                    photo = open(path, 'rb')
                    bot.send_photo(TG_USER, photo)

        if self.no_visitors():
            self.about_recognition = NO_CAPTURED_VISITORS_MSG

```

```

        bot.send_message(TG_USER, self.about_recognition)
    else:
        self.about_recognition = f'Фото ідентифікованих відвідувачів на відео:
{len(self.visitors)}.'

        bot.send_message(TG_USER, self.about_recognition)
        for visitor in self.visitors:
            photo = open(visitor.path, 'rb')
            bot.send_photo(TG_USER, photo)

```

В.6 Файл App.py

```

from tkinter import *
from tkinter import ttk
from package.classes.Helper import *
from package.classes.Alerts import *
from package.classes.Video import *
import PIL.Image, PIL.ImageTk
import cv2, os, time

class App():
    def __init__(self, window):

        self.helper = Helper()
        self.error = Error()
        self.info = Info()

        self.window = window
        self.window.title(WINDOW_TITLE)
        self.window.iconbitmap(ICON)
        self.window.resizable(False, False)

        self.launch = False
        self.selected_video, self.tmp = None, None

        top_frame = Frame(self.window)

        top_frame.pack(side = TOP, pady = 10)

        btn_frame = Frame(self.window, bg = BTN_FRAME_COLOR)
        btn_frame.pack(side = BOTTOM, pady = 5)

```

```

self.pause = False

self.canvas = Canvas(top_frame)
self.canvas.pack()

self.btn_select_video = Button(btn_frame, text = SELECT_VIDEO_LBL, width =
BTN_W, command = self.open_video)
self.btn_select_video.grid(row = 0, column = 0, ipadx = 6, ipady = 6, padx = 5, pady = 5)

self.btn_recognize_faces = Button(btn_frame, text = RECOGNIZE_FACES_MSG, width
= BTN_W, command = self.recognize_faces)
self.btn_recognize_faces.grid(row = 0, column = 1, ipadx = 6, ipady = 6, padx = 5, pady =
5)

self.btn_update_photo_collection = Button(btn_frame, text =
ADD_NEW_IMAGES_LBL, width = BTN_W, command = self.update_photo_collection)
self.btn_update_photo_collection.grid(row = 0, column = 2, ipadx = 6, ipady = 6, padx =
5, pady = 5)

self.btn_open_FAQ = Button(btn_frame, text = FAQ_LBL, width = BTN_W, command
= self.read_FAQ)
self.btn_open_FAQ.grid(row = 0, column = 3, ipadx = 6, ipady = 6, padx = 5, pady = 5)

self.btn_launch_video = Button(btn_frame, text = LAUNCH_VIDEO_LBL, width =
BTN_W, command = self.launch_video)
self.btn_launch_video.grid(row = 1, column = 0, ipadx = 6, ipady = 6, padx = 5, pady =
5)

self.btn_pause_video = Button(btn_frame, text = PAUSE_VIDEO_LBL, width =
BTN_W, command = self.pause_video)
self.btn_pause_video.grid(row = 1, column = 1, ipadx = 6, ipady = 6, padx = 5, pady = 5)

self.btn_relaunch_video = Button(btn_frame, text = RELAUNCH_LBL, width =
BTN_W, command = self.relaunch_video)
self.btn_relaunch_video.grid(row = 1, column = 2, ipadx = 6, ipady = 6, padx = 5, pady =
5)

self.btn_take_a_snapshot = Button(btn_frame, text = TAKE_A_SNAPSHOT_LBL, width
= BTN_W, command = self.take_a_snapshot)
self.btn_take_a_snapshot.grid(row = 1, column = 3, ipadx = 6, ipady = 6, padx = 5, pady
= 5)

```

```

self.delay = 15

self.window.mainloop()

def frames_are_of_acceptable_size(self, selected_video):
    self.cap = cv2.VideoCapture(selected_video)
    canvas_width, canvas_height = self.get_frame_size(selected_video)

    if MAX_FRAME_WIDTH >= canvas_width > 0 and MAX_FRAME_HEIGHT >=
canvas_height > 0:
        return True
    return False

def get_frame_size(self, selected_video):
    cap = cv2.VideoCapture(selected_video)
    canvas_width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
    canvas_height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
    return canvas_width, canvas_height

def configure_the_canvas(self, canvas, selected_video):
    return canvas.config(width = self.get_frame_size(selected_video)[0],
        height = self.get_frame_size(selected_video)[1])

def read_FAQ(self):
    faq = Toplevel()
    faq.geometry("645x350")
    faq.title("Довідка")

    faq_label1 = ttk.Label(faq, text=FAQ_STEP1, background="#FFCDD2",
foreground="#B71C1C", font=("Arial", 12), padding=3)
    faq_label2 = ttk.Label(faq, text=FAQ_STEP2, background="#D9FFBA",
foreground="#62B71C", font=("Arial", 12), padding=3)
    faq_label3 = ttk.Label(faq, text=FAQ_STEP3, background="#9CBFF7",
foreground="#2B1CB7", font=("Arial", 12), padding=3)
    faq_label4 = ttk.Label(faq, text=FAQ_STEP4, background="#C277ED",
foreground="#791AB0", font=("Arial", 12), padding=3)

    faq_label1.pack(fill=BOTH, expand=True)
    faq_label2.pack(fill=BOTH, expand=True)
    faq_label3.pack(fill=BOTH, expand=True)
    faq_label4.pack(fill=BOTH, expand=True)

```

```

def open_video(self):
    if self.selected_video is not None:
        self.pause = False
        self.tmp = self.selected_video

    self.selected_video = self.helper.get_videofile()

    if self.selected_video:
        self.helper.remove_old_snapshots()

        if self.frames_are_of_acceptable_size(self.selected_video):
            self.info.show(message = MAY_LAUNCH_MSG)
            self.configure_the_canvas(self.canvas, self.selected_video)
        else:
            self.error.show(message = CANT_DISPLAY_VIDEO_MSG)
    else:
        if self.tmp is not None:
            self.selected_video = self.tmp
        else:
            self.selected_video = None

def update_photo_collection(self):
    path_to_new_collection = self.helper.get_path_to_new_collection()

    if path_to_new_collection is not None:
        try:
            self.helper.move_images(path_to_new_collection)
        except:
            self.error.show(message = CANT_ADD_IMAGES_MSG)
    else:
        self.error.show(message = CANT_ADD_IMAGES_MSG)

def get_frame(self):
    try:
        if self.cap.isOpened() and self.launch:
            ret, frame = self.cap.read()
            return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    except:
        self.error.show(message = CANT_LAUNCH_VIDEO_MSG)

def launch_video(self):

```

```

try:
    if self.selected_video is None:
        self.error.show(message = VIDEO_NOT_SELECTED_MSG)
        return

    self.tmp = self.selected_video
    self.launch = True

    ret, frame = self.get_frame()

    if ret:
        self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(frame))
        self.canvas.create_image(0, 0, image = self.photo, anchor = NW)

    if not self.pause:
        self.window.after(self.delay, self.launch_video)
except:
    pass

def pause_video(self):
    if self.pause == True:
        pass
    if self.selected_video is None:
        self.error.show(message = VIDEO_NOT_SELECTED_MSG)

    self.launch = False
    self.pause = True

def relaunch_video(self):
    self.pause = False
    self.launch_video()

def take_a_snapshot(self):
    if self.selected_video is None or self.launch == False:
        self.error.show(message = VIDEO_NOT_SELECTED_MSG)
        return

    ret, frame = self.get_frame()
    if ret:
        cv2.imwrite(os.path.join(SNAPSHOTS_FOLDER, "frame-"
time.strftime("%d-%m-%Y-%H-%M-%S") + ".jpg"), cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

```

```

def recognize_faces(self):

    if self.selected_video is None:
        self.error.show(message = VIDEO_NOT_SELECTED_MSG)
    else:
        self.helper.remove_old_processed_photos()
        self.helper.crop_images_from_the_collection()

        video = Video(self.selected_video)

        visitors = self.helper.iterate_processed_images(video)
        report = Report(visitors)
        report.send()

def __del__(self):
    if self.cap.isOpened():
        self.cap.release()
        cv2.destroyAllWindows()

```

В.7 Файл Helper.py

```

import os, shutil
from shutil import copyfile, ignore_patterns
from package.constants import *
from tkinter import filedialog
from package.classes.Alerts import *
from package.classes.Image import *
from package.classes.ProcessedImage import *
import subprocess

class Helper:
    def remove_old_snapshots(self):
        for files in os.listdir(SNAPSHOTS_FOLDER):
            full_path = os.path.join(SNAPSHOTS_FOLDER, files)
            try:
                shutil.rmtree(full_path)
            except OSError:
                os.remove(full_path)

    def remove_old_processed_photos(self):

```

```

for files in os.listdir(PROCESSED_PHOTOS_FOLDER):
    full_path = os.path.join(PROCESSED_PHOTOS_FOLDER, files)
    try:
        shutil.rmtree(full_path)
    except OSError:
        os.remove(full_path)

def get_videofile(self):
    return filedialog.askopenfilename(title = SELECT_VIDEO_LBL,
        initialdir = VIDEOS_FOLDER,
        filetypes = VIDEOTYPES)

def get_path_to_new_collection(self):
    path = None
    out = subprocess.check_output(DISKS_CAPTIONS, shell = True)
    for drive in str(out).strip().split("\r\n"):
        if '2' in drive:
            drive_letter = drive.split(':')[0]
            if drive_letter not in PC_DISKS:
                path =
f"{drive_letter}:/IMAGES_FOLDER_ON_USB_FLASH_DRIVE}"
                break
    return path

def move_images(self, path):
    print(path)
    collection = os.listdir(path)
    for img in collection:
        if img.endswith(('jpg', 'png', 'gif')):
            copyfile(f"{path}/{img}",
f"{FULL_PATH_TO_PHOTOS_FOLDER}/{img}")

def crop_images_from_the_collection(self):
    for (root, dirs, images) in os.walk(PHOTOS_FOLDER):
        for image_name in images:
            if ("jpg" in image_name or ".png" in image_name or ".gif" in
image_name) and (PROCESSED_PHOTOS_FOLDER not in os.path.join(root, image_name).replace("\\", "/")):
                try:
                    path = os.path.join(root, image_name).replace("\\",
"/")

                    img = Image(path)
                    img.crop_face()

```

```

except Exception as e:
    pass

def iterate_processed_images(self, video):
    images_dict = {}
    for (root, dirs, images) in os.walk(PROCESSED_PHOTOS_FOLDER):
        for image_name in images:
            path = os.path.join(root, image_name).replace("\\", "/")
            image = ProcessedImage(path)
            image_encodings = image.get_encodings()
            images_dict[image] = image_encodings
    captured_visitors = video.get_capture_matches(images_dict)

    return captured_visitors

```

В.8 Файл Image.py

```

import cv2
import face_recognition
from package.constants import *

class Image:
    def __init__(self, path):
        self.path = path.split(f"{PHOTOS_FOLDER}")[1]
        self.img = cv2.imread(path)
        self.face_cascade = cv2.CascadeClassifier(FACE_RECOG_MODEL)

    def crop_face(self):
        faces = self.get_faces()
        for (x, y, w, h) in faces:
            self.img = self.img[y:y + h, x:x + w]
            path_to_cropped_img = f"{PROCESSED_PHOTOS_FOLDER}/{self.path}"
            cv2.imwrite(path_to_cropped_img, self.img)

    def get_faces(self):
        gray = self.convert2Gray()
        faces = self.face_cascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 4)
        return faces

    def convert2Gray(self):
        self.img = cv2.cvtColor(self.img, cv2.COLOR_BGR2GRAY)

```

```
return self.img
```

В.9 Файл ProcessedImage.py

```
import cv2
import face_recognition
from package.constants import *
from package.classes.Image import *

class ProcessedImage(Image):
    def __init__(self, path):
        self.path = path
        self.img = cv2.imread(path)

    def get_encodings(self):
        try:
            image_enc = face_recognition.face_encodings(self.img)[0]
            image_encs = [image_enc]
            return image_encs
        except Exception as e:
            return None
```

В.10 Файл Video.py

```
import cv2
import face_recognition
from package.constants import *

class Video:
    def __init__(self, path):
        self.cap = cv2.VideoCapture(path)
        self.path = path[path.find(VIDEOS_FOLDER + "/"):]
        self.face_cascade = cv2.CascadeClassifier(FACE_RECOG_MODEL)

    def get_face_locations(self, frame):
        return face_recognition.face_locations(frame)

    def get_face_encodings(self, frame, locations):
        return face_recognition.face_encodings(frame, locations)
```

```

def get_capture_matches(self, images_dict):
    captured_visitors = []
    while self.cap.read():
        try:
            success, img = self.cap.read()
            if not success:
                break
            frame = img[:, :, :-1]
            face_locations = self.get_face_locations(frame)
            face_encodings = self.get_face_encodings(frame, face_locations)
            for (top, right, bottom, left), face_encoding in zip(face_locations,
face_encodings):

                for image in images_dict.keys():
                    matches = face_recognition.compare_faces
(images_dict[image], face_encoding)

                    if True in matches:
                        if image not in captured_visitors:
                            captured_visitors.append(image)

        except Exception as e:
            pass
    return captured_visitors

```

ДОДАТОК Г
Слайди презентації

Національний університет "Запорізька політехніка"
Кафедра програмних засобів
Дипломна кваліфікаційна робота магістра

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ОБЛИЧ НА ВІДЕО

Виконав:
студент групи КНТ-111м
Керівник роботи:
к.т.н., доцент

О.С. Марков

Т.І. Каплієнко

2022

Рисунок Г.1 – Слайд 1

Об'єкт, предмет і мета дослідження

Об'єкт дослідження – технології нейромереж для обробки зображень та відео.

Предмет дослідження – програмні засоби для розпізнавання облич на відео.

Мета – створення програмної системи для розпізнавання облич визначених відвідувачів на відео та надсилання сповіщення за результатами в месенджер Telegram.

2

Рисунок Г.2 – Слайд 2

Існуючі методи розпізнавання облич

Метод	Точність	Швидкодія
Приховані марковські моделі	Незадовільна, оскільки максимізується відгук кожного зображення на свою модель без мінімізації інших моделей	Незадовільна через використовувані параметри
Метод Віоли-Джонса	Низька ймовірність помилкового виявлення обличчя	Задовільна для більшості об'єктів різного розміру та складності
Фільтр Габора	Задовільна, зокрема для розрізнення текстур	Незадовільна через обчислювальну складність
Метод головних компонент	Задовільна	Задовільна за ряду виконання умов ³

Рисунок Г.3 – Слайд 3

Існуючі методи ідентифікації облич

- ✓ Локальні бінарні шаблони;
- ✓ Eigenfaces;
- ✓ Scale-invariant feature transform.

Рисунок Г.4 – Слайд 4

Порівняння існуючих аналогів

Критерій порівняння	ZoneMinder	NetcamStudio
Багатозадачність	+	+
Кросплатформність	+	+
Розпізнавання в "один клік"	–	–
Збереження знімків	+	–
Сповіщення в соцмережах	–	+
Обмеження безкоштовної версії	–	–

5

Рисунок Г.5 – Слайд 5

Постановка завдання до роботи

За результатами дослідження проблеми розпізнавання облич на відео, було формалізовано завдання до роботи, яке полягає в розробці застосунку за наступними вимогами:

- реалізація графічного інтерфейса користувача;
- реалізація можливості вибору відеофайлу для роботи;
- реалізація можливості перегляду, зупинки для створення скріншотів та відновлення перегляду відео;
- реалізація можливості оновлення колекції зображень в один клік;
- реалізація сторінки з довідковою інформацією;
- реалізація розпізнавання облич на відео та надсилання сповіщення користувачеві в месенджер Telegram згідно його результатів.

6

Рисунок Г.6 – Слайд 6

Вибір мови програмування та середовища для розробки

Критерій порівняння	C++	Java	Python
Підтримка ООП	++	++	++
Продуктивність	++	++	+
Підтримка Machine Learning бібліотек	+	+	++
Простота розробки GUI	+	±	+
Лаконічність синтаксису	±	±	++
Крива навчальності	+	+	++
Широке ком'юніті розробників	++	++	++

Критерій порівняння	PyCharm	Visual Studio Code	Wing IDE	Sublime Text
Вбудовані плагіни	++	++	+	–
Автодоповнення	++	++	++	+
Зручність роботи з терміналом	++	+	++	+
Підсвічування коду	++	++	++	++
Можливість безкоштовної версії	+	+	++	+
Вимоги до ресурсів персонального комп'ютера	–	–	++	++
Відсутність реклами	++	+	++	–

7

Рисунок Г.7 – Слайд 7

Вибір додаткових сервісів та бібліотек

- ✓ Графічна бібліотека інтерфейсів Tkinter;
- ✓ Python Imaging Library для роботи з растровою графікою;
- ✓ OpenCV, dlib, face_recognition – програмні компоненти для роботи з відео та зображеннями;
- ✓ haarcascade_frontalface_default.xml – модель для розпізнавання фронтальних зображень облич;
- ✓ os, shutil, subprocess – робота з файлами та каталогами, керування потоками введення/виведення;
- ✓ pyTelegramBotAPI – інтерфейс для розробки Telegram-ботів.

8

Рисунок Г.8 – Слайд 8

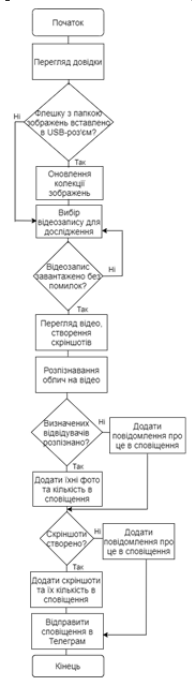
Загальна схема виявлення та ідентифікації облич



9

Рисунок Г.9 – Слайд 9

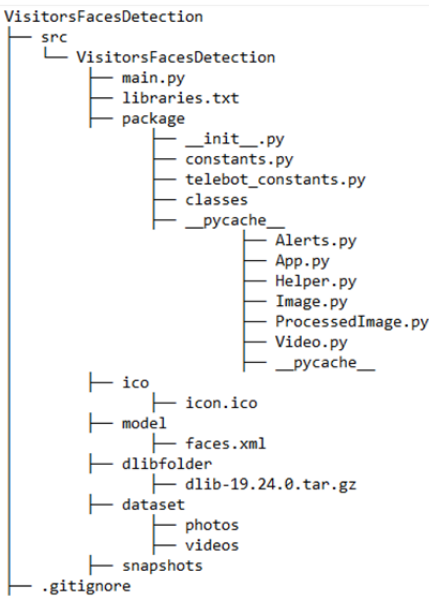
Алгоритм роботи програми



10

Рисунок Г.10 – Слайд 10

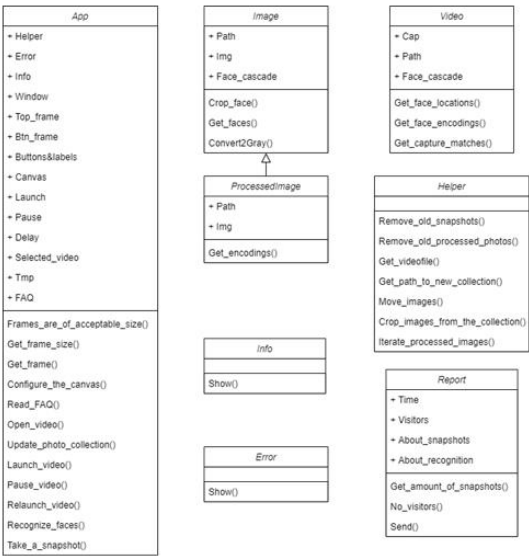
Файлова структура проекту



11

Рисунок Г.11 – Слайд 11

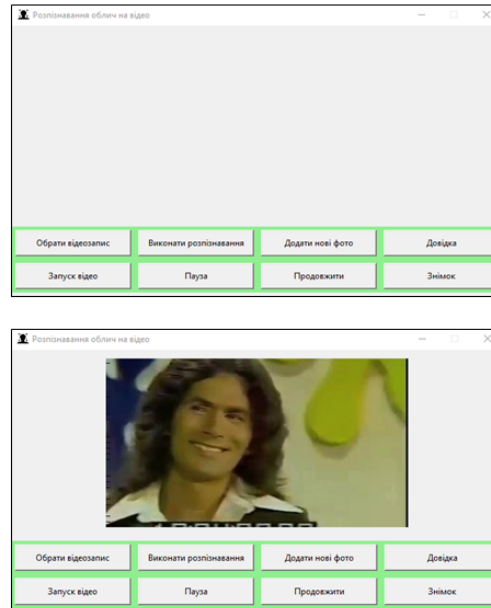
Структура класів



12

Рисунок Г.12 – Слайд 12

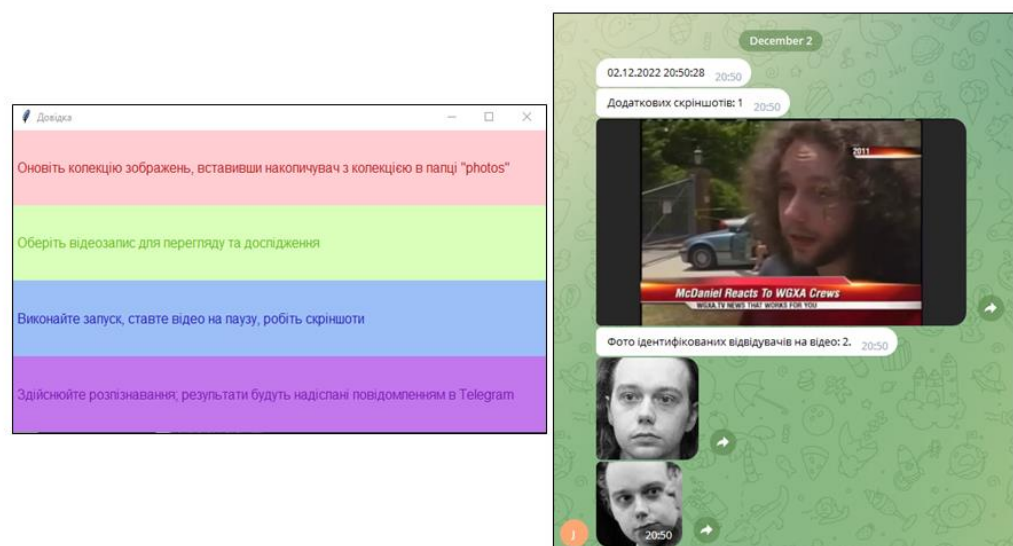
Екранні форми вікна програми до та після вибору відеофайлу



13

Рисунок Г.13 – Слайд 13

Екранні форми вікна “Довідка” та сповіщень користувача в Telegram



14

Рисунок Г.14 – Слайд 14

Загальний чекліст застосунку

Розпізнавання облич на відео		
№	Перевірка	Статус
1	Вибір відеофайлу	Пройдено успішно
	Перевірка роздільної здатності відео	Пройдено успішно
	Перевірка чи відбувається програвання відеозапису	Пройдено успішно
2	Перегляд довідки програми	Пройдено успішно
3	Оновлення колекції зображень	Пройдено успішно
	Перевірка наявності накопичувача з папкою-колекцією	Пройдено успішно
	Копіювання колекції з накопичувача в директорію програми	Пройдено успішно
4	Запуск відеофайлу	Пройдено успішно
	Зчитування відеокادрів	Пройдено успішно
	Циклічне розміщення кадрів на полотні вікна	Пройдено успішно
5	Зупинка відеофайлу	Пройдено успішно
6	Відновлення програвання відеофайлу	Пройдено успішно
7	Створення скріншоту	Пройдено успішно
	Перевірка чи відбувається програвання відеозапису	Пройдено успішно
	Запис скріншоту в директорію	Пройдено успішно
8	Розпізнавання облич на відео	Пройдено успішно
	Видалення оброблених фото застарілої колекції	Пройдено успішно
	Виокремлення облич обрізуванням зображень колекції	Пройдено успішно
	Переведення обрізаних фото в градації сірого	Пройдено успішно
	Отримання кодувань облич для оброблених фото	Пройдено успішно
	Співставлення кодувань зображень та відеокادрів	Пройдено успішно
9	Сповіщення про результати	Пройдено успішно
	Підрахунок виявлених осіб	Пройдено успішно
	Формування повідомлення	Пройдено успішно
	Відправка повідомлення в Телеграм	Пройдено успішно

15

Рисунок Г.15 – Слайд 15

Висновки

В ході виконання роботи було проаналізовано предметну область, описано існуючі матеріали та методи, спроєктовано архітектуру системи, реалізовано її компоненти та проведено тестування розробленої програми. Також оформлено програмні документи «Технічне завдання» та «Керівництво програміста».

Підсумком роботи є програмний продукт, що виконує розпізнавання облич на відео, ідентифікує визначених відвідувачів у колекції зображень та надсилає результати користувачеві застосунку в месенджер Telegram.

За потребою розроблена програма може бути модифікованою та слугувати основою для більш складних розробок систем аналізу відеоспостережень.

16

Рисунок Г.16 – Слайд 16