МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

Аліна САВЧЕНКО.

«_____» _____ 2022р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНА РОБОТА, ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"МАГІСТР"**

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ

"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: "Біометрична ідентифікація та автентифікація за геометрією облич"

Виконавець: студент групи УС-212М Бабич Олексій Дмитрович

Керівник: к.т.н., доцент Холявкіна Тетяна Володимирівна

Нормоконтролер: Ігор РАЙЧЕВ

Київ – 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____Аліна САВЧЕНКО
«_____»_____ 2022 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Бабича Олексія Дмитровича
(прізвище, ім'я, по батькові)

1. Тема роботи: «Біометрична ідентифікація та автентифікація за геометрією обличчя» затверджена наказом ректора № 1774/ст від 28.09.2022р.

2. Термін виконання роботи: з 26 вересня 2022 року по 27 листопада 2022 року.

3. Вихідні дані до роботи: мова розробки додатка – C#, технології розробки додатка WPF, EmguCV, середовище розробки Visual Studio.

4. Зміст пояснювальної записки: вступ, аналіз предметної області та постановка задачі, методи та засоби біометричної ідентифікації облич, програмна реалізація, висновки.

5. Перелік обов'язкового ілюстративного матеріалу: традиційні алгоритми розпізнавання обличчя, виявлення ознак обличчя, блок-схема алгоритму власних поверхонь, функціональна схема додатку розпізнавання обличчя, мануальне тестування додатку біометричної.

6. Календарний план-графік

№ п/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Проаналізувати літературу та джерела за темою дипломної роботи	26.09.22 – 28.09.22р.	
2.	Розроблення та затвердження плану дипломної роботи	28.09.22 – 29.09.22р.	
3.	Привести консультації з науковим керівником щодо створення першого розділу	29.09.22 – 30.09.22р.	
4.	Розробка розділу 1	30.09.22 – 10.10.22р.	
5.	Розробка розділу 2	10.10.22 – 20.10.22р.	
6.	Розробка розділу 3	20.10.22 – 31.10.22р.	
7.	Висновки та оформлення пояснювальної записки дипломної роботи	31.10.22 – 02.11.22р.	
8.	Підписання необхідних документів у встановленому порядку	16.11.22 –18.11.22р.	
9.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломної роботи	19.11.22 – 21.11.22р.	

7. Дата видачі завдання 26 вересня 2022р.

Керівник дипломної роботи _____ Тетяна ХОЛЯВКІНА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Олексій БАБИЧ
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Біометрична ідентифікація та автентифікація за геометрією обличчя» складається зі вступу, трьох розділів, висновку та списку бібліографічних посилань і містить 70 сторінок, 1 таблицю та 44 рисунка. Список бібліографічних посилань складається з 34 найменувань.

Ключові слова: БІОМЕТРИЧНА ІДЕНТИФІКАЦІЯ, РОЗПІЗНАВАННЯ ОБЛИЧ, АВТЕНТИФІКАЦІЯ ЗА ГЕОМЕТРІЄЮ ОБЛИЧЧЯ.

Актуальність роботи. Біометрична ідентифікація вже зайняла ключову позицію в багатьох сферах діяльності. Технологія стрімко розвивається, модифікується та інтегрується у будь-які процеси життєдіяльності, а тому для конкурентоспроможності ПЗ необхідно завчасно закладати системи ідентифікації.

Мета дипломної роботи: аналіз методів та засобів реалізації, розробка та тестування додатка біометричної ідентифікації людей за геометрією їх облич.

Об'єкт дослідження: додаток для біометричної ідентифікації за геометрією обличчя.

Предмет дослідження: розробка додатка біометричної ідентифікації за геометрією обличчя.

Теоретична основа: відчизняні та зарубіжні дослідження біометричної ідентифікації за геометрією облич, збірки статей наукових конференцій, тематичні публікації на сайтах.

Теоретична і практична значимість: на основі отриманих знань можна безкоштовно побудувати свою програмну реалізацію додатка біометричної ідентифікації з відкритим кодом та без ліцензійних обмежень за досить короткий проміжок часу, що дозволяє вільно модифікувати та оптимізовувати алгоритми аналізу.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

C# – об'єктно-орієнтована мова програмування.

OpenCV – бібліотека алгоритмів комп'ютерного зору.

.NET – платформа від Microsoft, яка дозволяє створювати програмні продукти.

EmguCV – крос-платформне .Net доповнення бібліотеки OpenCV для обробки зображень.

API – програмний інтерфейс програми.

WPF – платформа інтерфейсу користувача для створення клієнтських додатків для настільних систем.

XAML – декларативна мова розмітки.

MWP – продукт, що має мінімальні, але достатні для задоволення перших споживачів функції.

PoC – демонстрація практичної здійсненності будь-якого методу, ідеї, технології.

Face ID – технологія захисту, призначена для сканування обличчя.

UI – інтерфейс користувача.

PCA – метод факторного аналізу в статистиці.

БІ – біометрична ідентифікація.

БХЛ – біологічні характеристики людини.

FRR – ймовірність відмови власнику у доступі при авторизації.

FAR – ймовірність надання доступу некоректному користувачеві при авторизації.

ВСТУП

Біометрична автентифікація вже давно і досить глибоко проникла в життя багатьох людей. Наразі ми вже не звертаємо особливу увагу і не дивуємося тому, що цифрові прилади вміють розпізнавати нашу особу. А чи взагалі важливо це зараз? Люди, які не працюють у сфері цифрових технологій, часто навіть і не знають як функціонує біометрична автентифікація чи надійні такі технології, на якому етапі розвитку вони є, і що було б якби ми не мали можливості використовувати те що маємо.

Слід почати з того, що люди використовують кожен день, а саме мобільний телефон. Наразі всі нові моделі смартфонів мають функцію розпізнавання особи за геометрією обличчя. Здавалося б, що це всього лише приємний функціонал і без нього легко можна обійтися за допомогою звичайних паролів, але ні. Паролі дуже часто і з відносною легкістю викрадаються зловмисниками, а потім як результат стають ключем для викрадення персональних даних. Однак кібербезпека це не єдиний мінус. Людина в середньому за день розблоковує телефон близько 150 разів. Якщо врахувати, що на пароль довжиною в 6 символів ми витрачаємо приблизно 2-і секунди, то кожного дня така процедура витрачає 5 хвилин часу. Досить неприємний результат, враховуючи те, що за допомогою біометричної автентифікації цей процес займає мілісекунди.

Приклад зі смартфоном, насправді, є лише маленькою долею користі від інтеграції біометричної ідентифікації в наше повсякденне життя. Наразі ця технологія широко використовується в правоохоронних органах для пошуку злочинців, у приватних компаніях заради безпеки важливих бізнес секретів, у домівка, у банківських установах, навіть вже в державному додатку “Дія” для цифрового підпису, і у багатьох інших областях (рисунок 1). Найголовніше те, що все це робить людям повсякденність зручнішою і виграє найцінніший ресурс – час.

Рис. 1. Області застосування біометричних технологій

Усі системи, що використовуються для підтвердження особи людини базуються на методах біометричної верифікації. Такі методи основані на унікальних біологічних характеристиках людини (БХЛ). Існує дві групи таких особливостей – статичні та динамічні (рисунок 2). До динамічних особливостей відносять: голос, серцевий ритм, підпис та почерк, міміка. До статичних особливостей: геометрія тіла, геометрія обличчя, венозна архітектура, відбиток пальців, ДНК, сітківка ока, геометрія долоні. Слід відразу підкреслити, що системи ідентифікації особи за динамічними особливостями наразі не є широко поширеним через високу похибку ідентифікації, складність реалізації та вразливості алгоритмів до зовнішніх факторів.

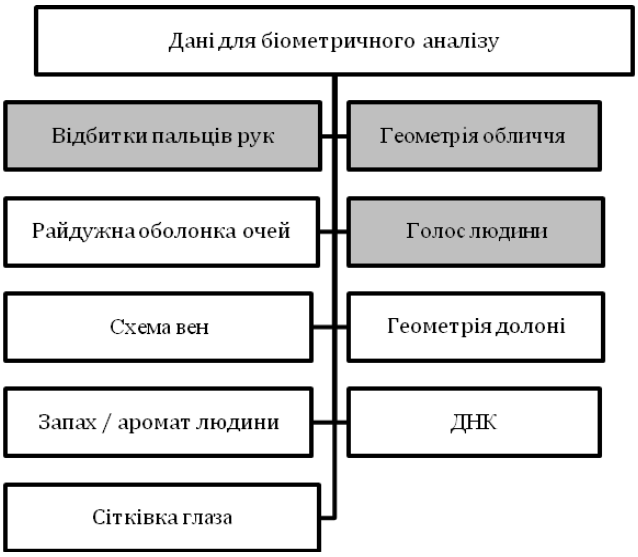

Рис. 2. Використовувані БХЛ

Основними вимогами до систем БІ є:

- швидкість (система повинна максимально швидко давати результат ідентифікації особи заради зручності використання і безпеки авторизації особи, що надала біометричні дані);
- надійність (системи БІ можуть бути використані для надання доступу для керування об'єктами критичної інфраструктури, до персональних даних, до секретної інформації, і т.д.);
- інтелектуальність (системи повинні ігнорувати дані, які отримані обманним або ж насильницьким методом).

Додатковими вимогами до систем БІ є:

- відносна дешевизна (в залежності від сфери використання, повинна бути варіація екземплярів для оптимізації ціни);
- простота використання (необхідно забезпечити легкий інтерфейс застосування для економії часу).

Щоб оцінити надійність систему біометричної ідентифікації використовується помилка першого (FRR - False Rejection Rate – ймовірність відмови власнику у доступі) та другого (FAR – False Acceptance Rate – ймовірність некоректного надання доступу іншому користувачеві) роду (таблиця 1).

Таблиця 1

Оцінки методів ідентифікації, заснованих на наведених БХЛ

Біометрична ознака	Точність		Простота використання
	FRR	FAR	
Геометрія обличчя	близько 6%	близько 0,1%	Висока
Відбиток пальця	близько 1%	близько 0,00002%	Середня
Голос	близько 3%	близько 0,1%	Висока
Райдужна оболонка ока	близько 0,2%	Близько 0,0001%	Середня
Схема вен	близько 0,01%	Близько 0,00008%	Середня

До суб'єктивних характеристик вибору факторів створення біометричної системи ідентифікації можна віднести наступне. Розглянемо їх більш детально.

Метод ідентифікації відбитків пальців. Переваги - відбитки пальців є унікальними ідентифікаторами, більшість людей знайомі з цим методом автентифікації, не потрібно запам’ятовувати паролі, сканери відносно дешеві. Недоліки - травми, тимчасові чи постійні, можуть заважати скануванню, це технологія, яку можна обійти за допомогою методів копіювання та реплікації відбитків пальців.

Метод ідентифікації по голосу. Переваги - не потрібно запам'ятовувати паролі, простота використання, низька вартість мікрофонів для запису звуку. Недоліки - шумні місця, зміна голосу з часом або при захворюванні можуть перешкодити успішній автентифікації, нездатність ідентифікувати глухих або німих людей.

Слід зауважити, що системи, які основані лише на одній біометричній характеристиці людини не забезпечують належної надійності [1]. Якщо висока надійність є вимогою для системи авторизації, то для таких випадків використовують багатофакторну біометричну ідентифікацію (ББІ) людини [2]. Звісно це займає більше часу для ідентифікації, але навіть дві характеристики значно підвищують надійність.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка задачі

Метою роботи є розробка системи біометричної ідентифікації за геометрією обличчя зі застосуванням реалізації на основі OpenCV бібліотеки.

Для досягнення поставленої мети повинні бути вирішені наступні завдання:

- аналіз основних методів для побудови системи біометричної ідентифікації,
- порівняльний аналіз існуючих систем біометричної ідентифікації,
- розробка системи біометричної ідентифікації людини,
- аналіз розробленої системи біометричної ідентифікації.

1.2. Аналіз існуючих рішень

Існує величезна різноманітність рішень для розпізнавання обличчя. Деякі з них готові до використання без навичок машинного навчання, а інші потребують набагато більше часу та досвіду.

Усі ці послуги розпізнавання обличчя можна розділити три типи, кожен зі своїми перевагами та недоліками:

1. Механізми розпізнавання обличчя на основі програмного забезпечення як послуги (SaaS). У цьому випадку постачальник послуг розпізнавання облич обробляє все: від підтримки технології машинного навчання до керування та підтримки високонавантажених серверів. Все, що вам потрібно зробити, це інтегрувати програмне забезпечення з вашими IT-

системами через API. Незважаючи на численні переваги, ці рішення також мають багато недоліків. По-перше, це найдорожчий варіант, оскільки все бере на себе провайдер. По-друге, вам потрібне стабільне підключення до Інтернету, оскільки вам потрібно буде надсилати важкі зображення на сервер десь в Інтернеті. По-третє, можуть виникнути проблеми з безпекою, оскільки ви повинні надсилати свої фотографії сторонній компанії та не можете контролювати, що вони з ними роблять. Приклад: Amazon Rekognition, Deep Vision AI.

2. Власні рішення REST API. Такі системи можна розгортати як локально, так і в хмарі. Вони не мають таких проблем, як продукти SaaS. Ви зберігаєте дані на власних серверах (або у своїй приватній хмарі), тож можете контролювати, куди вони надсилаються, і навіть можете створити систему, яка працює в автономному режимі. Звичайно, в обмін на це вам також доведеться самостійно керувати серверами. Але в більшості випадків сервери постачаються як докер-контейнери, тому ними досить легко керувати. Рішення, розміщені на власному хості, хоч і не такі дорогі, як їхні аналоги SaaS, але все ж достатньо дорогі. Приклад: FaceFirst, FaceX.

3. Фреймворки та бібліотеки з відкритим кодом. Зазвичай вони безкоштовні і досить добре забезпечують потреби проектів, що тільки починають розвиватися. Часто містять у собі найсучасніші рішення, так як основа таких безкоштовних бібліотек – люди, що займаються дослідженням у вільний час і лише заради створення чогось нового. Звісно з такими рішеннями є й багато проблем, а це – самостійне ознайомлення з кодовою базою, відсутність будь-яких гарантій, і звісно ж самостійна інтеграція функціоналу у ваш проект. Приклад: Ageitgey/face_recognition, CompreFace, DeepFace.

1.3. Застосування автентифікації за геометрією обличчя

Технологія є досить розповсюдженою та використовується в багатьох сферах життя.

Різні телефони, включаючи найновіші iPhone, використовують розпізнавання обличчя для розблокування пристрою. Технологія пропонує потужний спосіб захисту особистих даних і гарантує, що конфіденційні дані залишаться недоступними в разі викрадення телефону. Apple стверджує, що ймовірність того, що ваш телефон розблокує випадкове обличчя, становить приблизно один до мільйона [3].

Розпізнавання облич регулярно використовується правоохоронними органами. Відповідно до цього звіту NBC, технологія поширюється серед правоохоронних органів у США, і те саме стосується інших країн. Поліція збирає фотографії заарештованих і порівнює їх із місцевими, державними та федеральними базами даних розпізнавання облич. Після того, як заарештованого буде зроблено фото, його зображення буде додано до баз даних для сканування кожного разу, коли поліція проводить інший кримінальний обшук.

Крім того, мобільне розпізнавання обличчя дозволяє поліцейським використовувати смартфони, планшети чи інші портативні пристрої, щоб сфотографувати водія або пішохода на полі та негайно порівняти цю фотографію з однією або кількома базами даних розпізнавання облич, щоб спробувати ідентифікувати.

Розпізнавання облич стало звичним явищем у багатьох аеропортах світу. Все більше мандрівників мають біометричні паспорти, які дозволяють їм не стояти у зазвичай довгих чергах і натомість пройти через автоматичний електронний паспортний контроль, щоб швидше дістатися до воріт. Розпізнавання облич не тільки скорочує час очікування, але й дозволяє аеропортам підвищити рівень безпеки. Міністерство внутрішньої безпеки США прогнозує, що до 2023 року розпізнавання облич використовуватиметься на 97% мандрівників. Крім аеропортів і

прикордонних пунктів, ця технологія використовується для посилення безпеки під час масштабних заходів, таких як Олімпійські ігри.

Розпізнавання облич можна використовувати для пошуку зниклих безвісти та жертв торгівлі людьми. Припустімо, що відсутні особи додано до бази даних. У такому випадку правоохоронні органи можуть бути попереджені, щойно їх розпізнають за допомогою розпізнавання обличчя — незалежно від того, чи це відбувається в аеропорту, роздрібному магазині чи іншому громадському місці.

Розпізнавання облич використовується, щоб ідентифікувати, коли в магазини заходять відомі крадії, організовані злочинці в роздрібній торгівлі або люди з історією шахраїв. Фотографії окремих людей можна зіставити з великими базами даних злочинців, щоб спеціалісти із запобігання втратам і безпеки роздрібної торгівлі могли бути повідомлені, коли покупці, які потенційно становлять загрозу, заходять у магазин.

Технологія пропонує потенціал для покращення досвіду роздрібної торгівлі для клієнтів. Наприклад, кіоски в магазинах можуть розпізнавати клієнтів, пропонувати продукти на основі їхньої історії покупок і спрямовувати їх у правильному напрямку. Технологія «Face Pay» може дозволити покупцям пропускати довгі черги до каси за допомогою повільніших методів оплати.

Біометричний онлайн-банкінг є ще однією перевагою розпізнавання обличчя. Замість використання одноразових паролів клієнти можуть авторизувати транзакції, дивлячись на свій смартфон або комп’ютер. З розпізнаванням обличчя хакерам не потрібно зламати паролі. Якщо хакери викрадуть вашу базу даних фотографій, «неживе» виявлення – техніка, яка використовується для визначення того, чи є джерело біометричного зразка живою людиною чи підробленим зображенням – має (теоретично) завадити їм використовувати його для цілей видавання себе за іншу особу. Завдяки розпізнаванню обличчя дебетові картки та підписи можуть залишитися в минулому.

Маркетологи використовували розпізнавання облич для покращення споживчого досвіду. Наприклад, бренд замороженої піци DiGiorno використовував

розпізнавання обличчя для маркетингової кампанії 2017 року, де він аналізував вирази людей на тематичних вечірках DiGiorno, щоб оцінити емоційну реакцію людей на піцу.

Лікарні використовують розпізнавання облич для допомоги пацієнтам. Постачальники медичних послуг тестують використання розпізнавання облич, щоб отримати доступ до карт пацієнтів, спростити реєстрацію пацієнтів, виявити емоції та біль у пацієнтів і навіть допомогти визначити конкретні генетичні захворювання. AiCure розробила додаток [4], який використовує розпізнавання облич, щоб гарантувати, що люди приймають ліки відповідно до призначень. Оскільки біометрична технологія стає дешевшою, очікується, що її впровадження в секторі охорони здоров'я зростатиме.

Деякі навчальні заклади в Китаї використовують розпізнавання облич, щоб учні не пропускали заняття. Планшети використовуються для сканування облич студентів і зіставлення їх із фотографіями в базі даних для підтвердження їх особистості. У більш широкому плані технологія може бути використана для того, щоб працівники входили та виходили зі своїх робочих місць, щоб роботодавці могли відстежувати відвідуваність.

Відповідно до цього споживчого звіту, автомобільні компанії експериментують із розпізнаванням обличчя, щоб замінити автомобільні ключі. Технологія замінить ключ для доступу та запуску автомобіля та запам'ятає уподобання водія щодо положення сидінь і дзеркал і попередньо налаштованих радіостанцій.

1.4. Переваги розпізнавання обличчя

Окрім розблокування смартфона, розпізнавання обличчя має й інші переваги.

Розпізнавання облич полегшує пошук грабіжників, злодіїв і порушників. Лише знання про наявність системи розпізнавання облич може служити стримуючим фактором, особливо для дрібних злочинів. Окрім фізичної безпеки, кібербезпека також має переваги. Компанії можуть використовувати технологію розпізнавання облич замість паролів для доступу до комп'ютерів. Теоретично технологію неможливо зламати, оскільки нема чого красти чи змінювати, як у випадку з паролем.

Стурбованість громадськості необґрунтованими зупинками та обшуками є джерелом суперечок для поліції — технологія розпізнавання облич може покращити процес. Виділяючи підозрюваних серед натовпу за допомогою автоматизованого, а не людського процесу, технологія розпізнавання облич може допомогти зменшити потенційну упередженість і зменшити кількість зупинок і обшуків законослухняних громадян.

У міру того, як ця технологія стане все більш поширеною, клієнти зможуть розплачуватися в магазинах, використовуючи своє обличчя, а не витягувати кредитні картки або готівку. Це може заощадити час у чергах на касі. Оскільки для розпізнавання обличчя не потрібен контакт, як це буває під час зняття відбитків пальців чи інших заходів безпеки – корисних у світі після COVID-19, розпізнавання обличчя пропонує швидку, автоматичну та безпроблемну перевірку.

Процес розпізнавання обличчя займає лише секунду, що має переваги для компаній, які використовують розпізнавання обличчя. В епоху кібератак і передових інструментів злому компаніям потрібні як безпечні, так і швидкі технології. Розпізнавання обличчя дозволяє швидко та ефективно підтвердити особу людини.

Більшість рішень для розпізнавання обличчя сумісні з більшістю програмного забезпечення безпеки. Насправді він легко інтегрується. Це обмежує обсяг додаткових інвестицій, необхідних для його реалізації.

1.5. Недоліки розпізнавання осіб

Хоча деякі люди не заперечують проти того, щоб їх знімали в публічних місцях, і не заперечують проти використання розпізнавання обличчя там, де є очевидна користь або обґрунтування, ця технологія може викликати бурхливу реакцію в інших.

Деякі стурбовані тим, що використання розпізнавання облич разом із повсюдними відеокамерами, штучним інтелектом і аналітикою даних створює потенціал для масового стеження, яке може обмежити особисту свободу. Хоча технологія розпізнавання облич дозволяє урядам вистежувати злочинців, вона також може дозволити їм вистежувати звичайних і невинних людей у будь-який час.

Дані розпізнавання обличчя не позбавлені помилок, які можуть призвести до причетності людей до злочинів, яких вони не вчиняли. Наприклад, незначна зміна ракурсу камери або зміна зовнішності, як-от нова зачіска, може призвести до помилки. У 2018 році Newsweek повідомляв, що технологія розпізнавання облич Amazon помилково ідентифікувала 28 членів Конгресу США як людей, заарештованих за злочини [5].

Питання етики та приватності є найбільш дискусійним. Відомо, що уряди зберігають фотографії кількох громадян без їхньої згоди. У 2020 році Європейська комісія заявила, що розглядає можливість заборони технології розпізнавання облич у громадських місцях на термін до п’яти років, щоб дати час для розробки нормативної бази для запобігання конфіденційності та етичним порушенням.

Програмне забезпечення для розпізнавання облич базується на технології машинного навчання, яка потребує величезних наборів даних для «навчання», щоб надавати точні результати. Такі великі набори даних потребують надійного зберігання даних. Малі та середні компанії можуть не мати достатніх ресурсів для зберігання необхідних даних.

1.6. Огляд OpenCV бібліотеки

OpenCV (Open Source Computer Vision) — це набір програмних засобів для обробки зображень і відео в реальному часі, аналітики та машинного навчання. Набір інструментів OpenCV містить понад 2500 оптимізованих класичних і найсучасніших алгоритмів для комп'ютерного зору та машинного навчання [6]. Можливості OpenCV часто можна знайти лише у програмному забезпеченні комп'ютерного зору високого класу. Є багато компаній і дослідників, які використовують OpenCV у програмах комп'ютерного зору.

Приклади використання включають:

- розпізнавання обличчя;
- ідентифікація предметів;
- класифікація дій людини;
- відстежування руху камери;
- відстежування руху об'єктів;
- вилучення 3D моделі об'єктів;
- розпізнавання пейзажів.

API OpenCV надає бібліотеку будівельних блоків, необхідних для впровадження програм комп'ютерного зору. Бібліотека постійно доповнюється зусиллями багатьох розробників, що працюють у даній сфері. Функціонал бібліотеки дозволяє самостійно впроваджувати найсучасніші та високо оптимізовані алгоритми у свої проекти.

OpenCV має інтерфейси C++, Python, Java і MATLAB і підтримує Windows, Linux, Android і Mac OS операційні системи. Бібліотека здебільшого орієнтується на програми бачення в реальному часі та використовує переваги інструкцій MMX і SSE, якщо вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які створюють або підтримують ці алгоритми. OpenCV написаний на C++ і має шаблонний інтерфейс, який бездоганно працює з контейнерами STL.

1.7. Огляд EmguCV бібліотеки

Emgu CV — це кросплатформна бібліотека для обробки зображень. Вона тісно пов’язана з OpenCV, оскільки Emgu CV є оболонкою .NET для OpenCV. Можна сказати, що Emgu CV — це OpenCV у .NET. Ця оболонка дає змогу викликати функції OpenCV з мов програмування .NET. Серед підтримуваних мов є C#, VB, IronPython і VC++. Emgu CV можна скомпілювати в Mono, і вона працює під Linux, Windows, Mac OS X і під популярними мобільними платформами, такими як пристрої Android, iPhone, iPod Touch і iPad [7].

Через те, що .NET є інтерпретованою платформою, яка не може безпосередньо викликати функції чи методи, написані рідною мовою C або C++, розробники .NET можуть використовувати Emgu CV для вирішення проблеми. За допомогою P/Invoke і великої кількості власноруч створених хуків можна викликати C++ із C#. Але Emgu CV робить його більш прозорим для користувача.

Однією з цілей Emgu CV є створення простої у використанні інфраструктури комп’ютерного бачення для програмістів .NET, яка допомагає їм швидко створювати досить складні програми бачення. Бібліотека Emgu CV охоплює багато областей бачення, включаючи інспекцію продукції заводу, медичне зображення, інтерфейс користувача, калібрування камери, стереобачення та робототехніка. Оскільки комп’ютерний зір і машинне навчання часто йдуть рука об руку, Emgu CV також містить повну бібліотеку машинного навчання загального призначення з бібліотеки обробки зображень OpenCV.

1.8. Вимоги до проекту

Вимоги до системи біометричної автентифікації за геометрією обличчя:

- використання операційної системи Windows 10;
- використання функціоналу OpenCV бібліотеки;
- використання спеціальних ознак обличчя для реалізації алгоритму розпізнавання;

- наявність графічного інтерфейсу користувача для перевірки працездатності функціоналу програми;

1.9. Задача проекту

Задачі, які необхідно вирішити, для реалізації системи біометричної автентифікації за геометрією обличчя:

- порівняння та вибір біометричних алгоритмів для роботи системи;
- створення програмної бази для демонстрації функціоналу;
- власна реалізація алгоритму біометричної автентифікації за геометрією обличчя;
- пропозиції щодо оптимізації алгоритму;
- мануальне тестування програми.

ВИСНОВОК ДО РОЗДІЛУ 1

Отже, в першому розділі було розглянуто технологію біометричної автентифікації особи за геометрією обличчя. Безперечно можна сказати, що тема надзвичайно актуальна зараз і буде актуальною у майбутньому. Уже існує досить велика кількість рішень біометричної автентифікації для різних сфер життя, починаючи від простого розблокування телефонів і закінчуючи використанням у медицині для виявлення генетичних захворювань. Усі вони мають як свої переваги, так і недоліки. Одні з найважливіших недоліків це висока собівартість готового рішення і питання безпеки, яка є не абсолютною навіть у найкращих спеціалізованих компаніях.

Наявність великих open-source проектів, що займаються саме біометрією, додатково підтверджує сучасність тематики та недостатню досконаліть готових рішень. Кожного дня створюються нові алгоритми обробки візуальних даних та покращуються старі, заради зменшення похибки, пришвидшення роботи систем, підвищення безпеки обробки та зберігання даних. Часто саме рішення з таких проектів стають основою поштовху для подальшого розвитку технологій.

Проаналізувавши предметну область, було обрано EmguCV бібліотеку та мову програмування C# як основу для реалізації оптимізованого алгоритму ідентифікації особи за геометрією обличчя, саме через їх сучасність та швидкодію.

РОЗДІЛ 2. МЕТОДИ ТА ЗАСОБИ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ОБЛИЧ

2.1. Методи виявлення обличчя

Ming-Hsuan Yang, David Kriegman і Narendra Ahuja представили класифікацію методів виявлення обличчя [8]. Ці методи розділені на чотири категорії, а алгоритми виявлення обличчя можуть належати до двох або більше груп. Ці категорії такі: підхід на основі знань, відповідність шаблону, інваріантність ознак і підхід на основі зовнішнього вигляду (рис. 2.1).

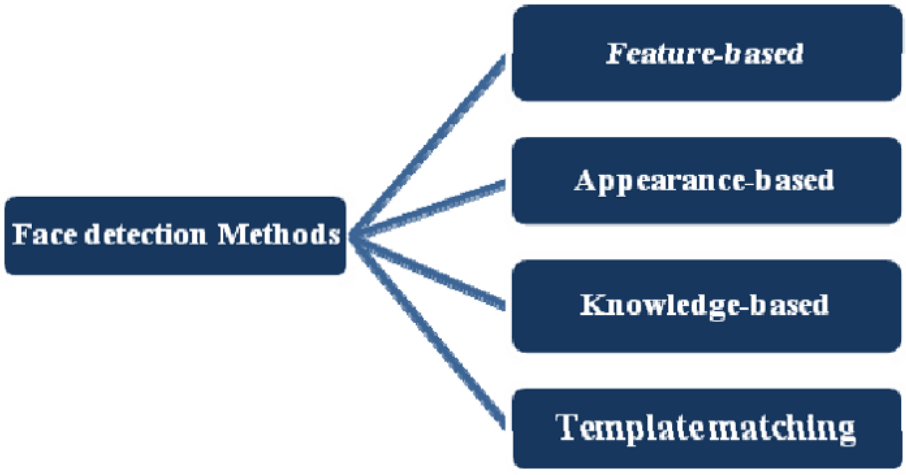

Рис. 2.1. Категорії методів виявлення обличчя

Метод, заснований на знаннях, залежить від набору правил і ґрунтується на знаннях людини для виявлення облич. Для прикладу обличчя повинно мати ніс, очі та рот на певній відстані та в певному положенні одне від одного. Великою проблемою цих методів є складність побудови відповідного набору правил. Може бути багато хибних спрацьовувань, якщо правила були надто загальними або надто детальними. Самого цього підходу недостатньо, і він не може знайти багато облич на кількох зображеннях.

Метод, заснований на ознаках, полягає у визначенні облич шляхом виділення структурних особливостей обличчя. Спочатку він тренується як класифікатор, а потім використовується для розрізнення між лицьовими та нелицевими областями. Ідея полягає в тому, щоб подолати межі нашого інстинктивного знання облич. Цей підхід, поділений на кілька етапів, і навіть фотографії з багатьма обличчями повідомляють про успіх у 94%.

Метод зіставлення шаблонів використовує попередньо визначені або параметризовані шаблони облич для пошуку або виявлення облич за кореляцією між шаблонами та вхідними зображеннями. Обличчя людини можна розділити на очі, контур обличчя, ніс і рот. Крім того, модель обличчя може бути побудована за ребрами, просто використовуючи метод виявлення країв. Цей підхід простий у реалізації, але він недостатній для виявлення обличчя. Однак для вирішення цих проблем були запропоновані шаблони, що деформуються.

Метод на основі зовнішнього вигляду залежить від набору зображень облич делегатів для навчання, щоб знайти моделі облич. Підхід, заснований на зовнішньому вигляді, кращий за інші способи виконання. Загалом метод на основі зовнішнього вигляду покладається на методи статистичного аналізу та машинного навчання, щоб знайти відповідні характеристики зображень обличчя. Цей метод також використовується для виділення ознак для розпізнавання обличчя.

2.2. Методи ідентифікації обличчя

За останнє десятиліття відбувся швидкий розвиток надійних алгоритмів розпізнавання облич. Традиційні алгоритми розпізнавання обличчя можна розділити на дві категорії: цілісні функції та підходи до локальних ознак. Цілісну групу можна додатково розділити на лінійні та нелінійні методи проектування.

Багато додатків показали хороші результати методів на основі зовнішнього вигляду лінійної проекції, таких як аналіз головних компонентів (Principal Component Analysis, PCA), аналіз незалежних компонентів (Independent Component Analysis, ICA), лінійний дискримінаційний аналіз (Linear Discriminant Analysis,

LDA), (Two-Dimensional PCA, 2DPCA) та класифікатора лінійної регресії (Linear Regression Classification, LRC).

Однак через значні варіації умов освітлення, виразу обличчя та інших факторів ці методи можуть не відобразити обличчя належним чином. Основна причина полягає в тому, що візерунки граней лежать на складному нелінійному та невипуклому різноманітті у високовимірному просторі.

Щоб мати справу з такими випадками, були запропоновані нелінійні розширення, такі як ядро PCA (Kernal PCA, KPCA), ядро LDA (Kernal LDA, KLDA) або локально лінійне вбудовування (Locally Linear Embedding, LLE). Найбільш нелінійні методи, що використовують методи ядра, де загальна ідея полягає у відображенні вхідних зображень граней у простір вищих розмірів, у якому різноманітність граней є лінійною та спрощеною. Тому можна застосовувати традиційні лінійні методи.

Хоча PCA, LDA і LRC розглядаються як лінійні алгоритми навчання підпростору, слід зазначити, що методи PCA і LDA фокусуються на глобальній структурі евклідового простору, тоді як підхід LRC фокусується на локальній структурі різноманіття.

Ці методи проектують обличчя на лінійний підпростір, охоплений зображеннями власних граней. Відстань від грані - це ортогональна відстань до площини, тоді як відстань в грані - це відстань уздовж площини від середнього зображення. Ці обидві відстані можна перетворити на відстані Махаланобіса та дати ймовірнісні інтерпретації.

Слідом за ними були розроблені: KPCA, ядро ICA і узагальнений лінійний дискримінантний аналіз.

Незважаючи на міцну теоретичну основу методів на основі ядра, практичне застосування цих методів у задачах розпізнавання обличчя, на жаль, не дає значного покращення порівняно з лінійними методами.

Було представлено інше сімейство методів нелінійного проектування. Вони успадкували простоту від лінійних методів і здатність працювати зі складними

даними від нелінійних. Серед цих методів варто виділити: LLE та проекцію зі збереженням локальності (Locality preserving projection, LPP). Вони створюють схему проекції лише для навчальних даних, але їх здатність проектувати нові елементи даних викликає сумніви.

У другій категорії локальні особливості зовнішнього вигляду мають певні переваги перед цілісними. Ці методи більш стійкі до локальних змін, таких як вираз, оклюзія та зміщення. Загальний репрезентативний метод називає локальні бінарні шаблони (Local Binary Patterns, LBP). Сусідні зміни навколо центрального пікселя простим, але ефективним способом описуються LBP. Це інваріантне монотонне перетворення інтенсивності та підтримує невеликі варіації освітлення. Пропонується багато варіантів LBP для покращення оригінального LBP, наприклад гістограма фазових шаблонів Габора і локальна послідовність бінарних шаблонів гістограми Габора. Як правило, LBP використовується для спільного моделювання сусідніх відносин у просторовій, частотній та орієнтаційній областях.

Це дозволяє ефективно досліджувати дискримінантну та надійну інформацію в шаблоні. Подальшим розвитком згаданих підпросторових підходів є підхід дискримінантних спільних векторів (Discriminant Common Vectors, DCVs).

Метод DCV збирає подібності між елементами в одному класі та видаляє їх відмінності. Таким чином, кожен клас може бути представлений загальним вектором, обчисленим із внутрішньої матриці розсіювання.

У разі тестування невідомого обличчя відповідний вектор ознак обчислюється та пов’язується з класом із найближчим загальним вектором. Іноді для завдання розпізнавання обличчя вводяться дискримінаційні загальні вектори ядра або покращені дискримінаційні загальні вектори та машина підтримки векторів (Support Vector Machine, SVM).

Подібно до методу LLE, проекція із збереженням сусідства (Neighbourhood Preserving Projection, NPP) і ортогональна NPP (Orthogonal NPP, ONPP). Ці підходи зберігають локальну структуру між зразками. Щоб відобразити внутрішню геометрію місцевих околиць, вони використовують ваги, керовані даними,

розв’язуючи проблему найменших квадратів. ONPP змушує відображення бути ортогональним, а потім вирішує звичайну проблему власних значень. NPP вимагає вирішення узагальненої задачі на власні значення щодо накладення умови ортогональності на прогнозовані дані.

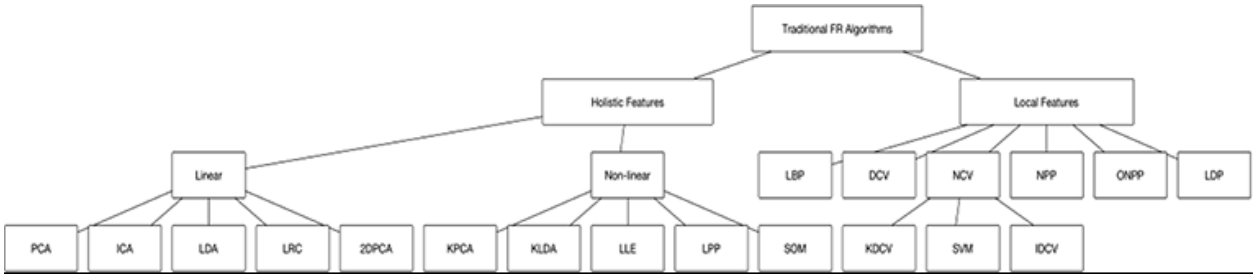

Рис. 2.2. Традиційні алгоритми розпізнавання обличчя

Однак досі незрозуміло, як вибрати розмір околиці та як призначити оптимальні значення для інших гіперпараметрів; для них також застосовуються проекції зі збереженням розрідженості і LPP для розпізнавання обличчя.

У [9] пропонується багатолінійне розширення методу LDA, що називається дискримінантним аналізом із тензорним представленням. Він відрізняється від методів збереження проекції та реалізує дискримінантний аналіз безпосередньо на природних тензоріальних даних, щоб зберегти структуру сусідства простору ознак тензора. Інший метод керованого та неконтрольованого багатолінійного NPP (Multi-line NPP, MNPP) для розпізнавання обличчя представлено в [10]. Огляд багатолінійних методів можна знайти в [11]. Вони працюють безпосередньо з тензоріальними даними, а не з векторами чи матрицями, і вирішують проблеми тензоріального представлення для виділення та розпізнавання багатовимірних ознак. Множинні взаємопов’язані підпростори отримують у методі MNPP шляхом розгортання тензора в різних тензорних напрямках. Порядок тензорного простору визначає кількість підпросторів, отриманих MNPP.

2.2.1. Метод на основі штучних нейронних мереж

В роботі [11] штучні нейронні мережі використовуються для вирішення нелінійної задачі. Для розпізнавання людських облич запропоновано незбіжну хаотичну нейронну мережу в [12].

Нейронна мережа з радіальною базисною функцією, інтегрована з невід'ємною матричною факторизацією для розпізнавання облич, представлена в роботі [13]. Крім того, для верифікації обличчя та мови використовується нейронна мережа зі зворотним розповсюдженням імпульсу. Метод невід'ємного розрідженого кодування для навчання ознак обличчя з використанням різних метрик відстані та нормалізованої крос-кореляції для розпізнавання облич застосовується в [14].

В [10] запропоновано підхід на основі штучної нейронної мережі, що базується на апостеріорному об'єднанні рішень. Він має елементи як нейромережевого, так і статистичного підходів і поповнює методи розпізнавання зображень облич з частковим спотворенням та оклюзією.

На жаль, цей підхід, як і інші статистичні методи, є неточним для моделювання класів за наявності лише однієї або невеликої кількості навчальних вибірок.

2.2.2. Метод на основі вейвлетів Габора

Вейвлети Габора широко використовуються для представлення облич розпізнувачами, і ознаки Габора визнані кращим представленням для розпізнавання облич з точки зору швидкості [15]. Крім того, було продемонстровано, що метод є дискримінативним та стійким до змін освітлення та виразу обличчя. У випадку, коли доступне лише одне зображення-зразок для кожного суб'єкта, пропонується адаптивно-зважений суб-габорівський масив для представлення та розпізнавання облич.

Крім того, є два види стратегій для захоплення інформації про текстуру Габора: Габорівське представлення текстури на основі величини (Gabor Magnitude-based Texture Representation, GMTR) та Габорівське представлення текстури на основі фази (Gabor Phase-based Texture Representation, GPTR), запропоновані в [16].

Підхід GMTR характеризується використанням гамма-щільності для моделювання розподілу амплітуди Габора. GPTR характеризується узагальненою гауссовою щільністю для моделювання фазового розподілу Габора. Це дозволяє використовувати оцінені параметри моделі як текстурне представлення обличчя.

Вейвлет Габора, застосований у фіксованих позиціях, що відповідають вузлам квадратної сітки, накладеної на зображення обличчя. Кожен підшаблон розбитого зображення обличчя визначається як виділені габорівські ознаки, що належать одному рядку квадратної сітки, які потім проектуються в простір меншої розмірності за допомогою перетворення Кархунена-Лоєва. Отримані ознаки кожного підшаблону, які зважуються за допомогою генетичного алгоритму (ГА), використовуються для навчання віконного класифікатора Парзена. Нарешті, процес зіставлення здійснюється шляхом об'єднання класифікаторів за правилом зваженої суми.

Підхід до навчання на основі ознак Габора для розпізнавання облич в рамках концепції злиття класифікаторів представлено в [17]. Ознаки Габора, отримані з кожного каналу як новий зразок того ж класу, використовуються для прийняття

стратегії злиття класифікаторів. Такий підхід є корисним для покращення якості результатів розпізнавання.

Гістограма фазової ознаки Габора розглянута в [18]. В даній роботі гістограми локальних шаблонів на основі патчів об'єднуються разом, щоб сформувати представлення зображення обличчя за допомогою вивчених локальних шаблонів Габора. Проблема представлення ознак за допомогою методу навчання замість простої конкатенації або гістограми ознак представлена в [19]. Ознаки Габора були прийняті для класифікації на основі розрідженого представлення (РП), а словник оклюзій Габора був навчений в рамках відомої системи РП.

Основним недоліком методів на основі Габора є те, що розмірність простору ознак Габора є суттєво великою, оскільки зображення обличчя згорнуті за допомогою фільтрів Габора.

Для подолання цієї проблеми використовують алгоритм AdaBoost та ентропійні і генетичні алгоритми для вибору найбільш дискримінативних габорівських ознак.

Однак, вибір найбільш корисного методу з такої кількості ознак Габора є дуже трудомістким. Крім того, вилучення ознак Габора вимагає значних обчислювальних витрат, тому в даний час ці ознаки не можуть бути використані для додатків, що працюють в реальному часі. Спрощена версія вейвлетів Габора представлена в роботі [20]. На жаль, спрощені вейвлети Габора більш чутливі до змін освітлення, ніж оригінальні.

2.2.3. Метод на основі дескрипторів обличчя

Опис зображення обличчя на основі локальних ознак забезпечує глобальний опис. Таким чином, локальні ознаки зображення оцінюються в сусідніх пікселях, а потім агрегуються для формування остаточного глобального опису. Це відрізняється від глобальних методів, в яких для отримання кожної ознаки використовується все зображення, де перші кроки починаються з опису обличчя, реалізованого на рівні пікселів, з використанням локального оточення кожного пікселя. Потім зображення

розбивається на ряд субрегіонів, і з кожного субрегіону формується локальний опис у вигляді гістограми описів на рівні пікселів, розрахованих на попередньому кроці. Далі інформація регіонів об'єднується в кінцевий дескриптор шляхом конкатенації часткових гістограм.

Визначення дескрипторів зображень, які здатні покращити класифікаційні характеристики багатоваріантного розпізнавання, а також парного зіставлення зображень облич, є складною проблемою.

Основна ідея підходу полягає у вивченні найбільш дискримінантних локальних ознак, які можуть мінімізувати різницю ознак між зображеннями однієї особи та максимізувати різницю між зображеннями інших людей в залежності від природи цих дескрипторів, які обчислюють представлення зображення з локальної статистики патчів, що є основною ідеєю підходу.

Точність верифікації обличчя, оцінена на бенчмарку LFW після верифікації обличчя з використанням декількох локальних дескрипторів, призначених для захоплення статистики локальної схожості патчів. Підвищення ефективності розпізнавання облич шляхом введення дискримінативного навчання в три етапи виділення LBP-подібних ознак представлено в [21].

Дискримінантні фільтри зображень, оптимальна матриця м'якої дискретизації та домінуючі шаблони навчаються на основі зображень. Загальною перевагою цих методів є компактність, висока дискримінативність та легкість вилучення дескриптора на основі навчання. Ці методи є дискримінативними та стійкими до змін освітлення та експресії.

2.2.4. Метод на основі 3D-зображень

Оскільки процес 3D-зйомки стає дешевшим і швидшим, прийнято вважати, що використання 3D-зйомки має потенціал для більшої точності розпізнавання, ніж 2D-зйомка. Перевага використання 3D-даних полягає в тому, що інформація про глибину не залежить від пози та освітлення, а отже, зображення об'єкта не змінюється зі зміною цих параметрів, що робить всю систему більш надійною. Методи на основі 3D-даних дозволяють досягти кращої стійкості до проблеми варіації поз, ніж методи на основі 2D-даних. Комплексний огляд підходів до 3D розпізнавання облич представлено в роботі [22].

Метод розпізнавання облич за варіаціями поз, який поєднує деформівні 3D-моделі з комп'ютерною графічною симуляцією проекції та освітлення, можна знайти в [23]. У цьому методі обличчя представлені параметрами моделі для 3D форми та текстури. Їх 3D морфіновані моделі поєднуються з представленням освітлення сферичними гармоніками для розпізнавання облич при довільному невідомому освітленні.

Використання симетрії обличчя для обробки варіацій пози при розпізнаванні 3D-обличчя представлено в [24], де використовується автоматичний детектор орієнтирів. Він допомагає оцінити позу і виявляє закриті області для кожного сканування обличчя. Після цього анотована модель обличчя реєструється та підганяється до скана. Під час підгонки використовується симетрія обличчя для подолання проблем, пов'язаних з відсутністю даних.

В [25] запропонована загальна 3D еластична модель для інваріантного розпізнавання обличчя в позі. Вона будується для кожного суб'єкта в базі даних з використанням лише одного 2D зображення шляхом застосування підходу 3D загальної еластичної моделі (3D Generic Elastic Model, 3DGEM). Кожна 3D-модель згодом рендериться в різних позах в обмеженому просторі пошуку щодо оцінюваної пози, і отримані зображення зіставляються з тестовим запитом. Нарешті, відстані між синтезованими зображеннями і тестовим запитом обчислюються за допомогою

простого нормалізованого кореляційного співставлення, щоб показати ефективність методу синтезу поз для реальних даних.

В роботі [26] пропонується геометрична основа для аналізу 3D-обличчя, з конкретними цілями порівняння, зіставлення та усереднення їх форм, для представлення поверхонь обличчя радіальними кривими, що виходять з кінчиків носа.

Підхід до розпізнавання 3D-обличчя на основі локальних геометричних ознак, які називаються кутовими радіальними ознаками обличчя (ARS), що можуть апроксимувати напівжорстку область 3D-обличчя, запропоновано в роботі [27]. Автори використовували KPCA для відображення необроблених ознак обличчя ARS на ознаки середнього рівня для покращення дискримінаційної здатності. Нарешті, отримані ознаки середнього рівня об'єднуються в один єдиний вектор ознак і подаються в SVM для виконання розпізнавання обличчя.

Недоліком використання 3D-даних для розпізнавання облич є те, що ці підходи до розпізнавання облич потребують, щоб всі елементи системи були добре відкалібровані та синхронізовані для отримання точних 3D-даних (карти текстур та глибини). Існуючі підходи до 3D-розпізнавання облич покладаються на реєстрацію поверхні або на складні методи вилучення та співставлення характеристик (дескрипторів поверхні). Таким чином, вони є обчислювально дорогими і не підходять для практичного застосування. Крім того, вони вимагають співпраці з об'єктом, що робить їх непридатними для неконтрольованих або напівконтрольованих сценаріїв, де єдиним вхідним сигналом для алгоритмів буде двомірне зображення інтенсивності, отримане з однієї камери.

2.2.5. Метод на основі відео

Аналізу відеопотоків зображень облич приділяється все більше уваги в біометрії. Безпосередньою перевагою використання відеоінформації є можливість використання надмірності, присутньої у відеопослідовності, для покращення систем розпізнавання нерухомих зображень. Хоча в області зіставлення нерухомих зображень облич проведено значну кількість досліджень, використання відео для розпізнавання облич є відносно менш вивченим. Першим етапом розпізнавання облич на основі відео (Video-based Face Recognition, VFR) є виконання повторної ідентифікації, де колекція відеозаписів перехресно зіставляється для визначення місцезнаходження всіх випадків появи особи, що цікавить.

Як правило, підходи VFR можна розділити на дві категорії на основі того, як вони використовують безліч інформації, доступної у відеопослідовності: на основі послідовності та на основі набору. Найбільше відрізняє ці два підходи - це те, чи використовують вони часову інформацію.

Формулювання імовірнісного підходу до розпізнавання облич на основі зовнішності розглянуто в [28]. Спочатку було визначено, що розпізнавання здійснюється за одним нерухомим зображенням, як пояснювалося раніше, для роботи з декількома зображеннями та відео послідовностями. В роботі [28] існує обмежений підпростір, що охоплює зображення обличчя кліпу в опуклу оболонку, а потім обчислюється найближча відстань між двома опуклими оболонками як між множинна схожість. Таким чином, кожен тестовий та навчальний приклад є набором зображень обличчя об'єкта, а не просто одним зображенням, тому рішення про розпізнавання повинні базуватися на порівнянні наборів зображень.

В роботі [29] задача VFR перетворюється в задачу вимірювання схожості двох наборів зображень, де приклади з відео кліпу будують один набір зображень. Автори розглядають зображення облич з кожного кліпу як ансамбль і формулюють задачу VFR у вигляді задачі спільного розрідженого представлення (Joint Sparse Representation, JSR). В JSR для адаптивного навчання розрідженого представлення пробного кліпу вони одночасно розглядають розрідженість на рівні класів і на рівні

атомів, де перша структурує зареєстровані кліпи за допомогою структурованого розрідженого регуляризатора, а друга шукає кілька пов'язаних прикладів.

2.3. Огляд алгоритму Віоли-Джонса

Алгоритм Віоли-Джонса — це техніка машинного навчання для виявлення об’єктів, запропонована в 2001 році Полом Віолою та Майклом Джонсом у їхній статті «Швидке виявлення об’єктів за допомогою розширеного каскаду простих функцій» [30]. Алгоритм в першу чергу був задуманий для виявлення обличчя. Незважаючи на нижчу точність, ніж сучасні методи виявлення обличчя на основі згорткових нейронних мереж (Convolutional Neural Networks, CNN), алгоритм Віоли-Джонса все ще є ефективним рішенням для пристроїв з обмеженими ресурсами.

Враховуючи зображення в градаціях сірого, алгоритм аналізує багато вікон різних розмірів і позицій і намагається виявити цільовий об’єкт, шукаючи певні особливості зображення в кожному вікні.

Алгоритм Віоли-Джонса базується на чотирьох основних ідеях:

- Особливості Хаара;
- Інтегральні зображення для прискорення обчислення ознак;
- Навчання AdaBoost для вибору функцій;
- Каскад класифікаторів для швидкого відхилення вікон без облич.

2.3.1. Особливості Хаара

У 19 столітті угорський математик Альфред Хаар дав поняття вейвлетів Хаара, які являють собою послідовність перемасштабованих функцій «квадратної форми», які разом утворюють сімейство вейвлетів або базис. Віола та Джонс адаптували ідею використання вейвлетів Хаара та розробили так звані Хаара-подібні функції.

Характеристики подібні до Хаара — це ознаки цифрового зображення, які використовуються для розпізнавання об’єктів. Усі людські обличчя мають деякі універсальні властивості людського обличчя, наприклад область очей темніша за сусідні пікселі, а область носа яскравіша за область очей.

Простий спосіб дізнатися, яка область світліша чи темніша, полягає в тому, щоб підсумувати значення пікселів обох областей і порівняти їх. Сума значень пікселів у темнішій області буде меншою, ніж сума пікселів у світлішій області. Якщо одна сторона світліша за іншу, це може бути край брови або інколи середня частина може бути блискучою за навколишні прямокутники, що можна інтерпретувати як ніс. Це можна зробити за допомогою рис, схожих на Хаара, і за допомогою з них ми можемо інтерпретувати різні частини обличчя.

У своєму дослідженні Віола та Джонс виявили 3 типи ознак, подібних до Хаара (рис. 2.3):

- Особливості краю;
- Лінія-особливості;
- Чотиристоронні особливості.

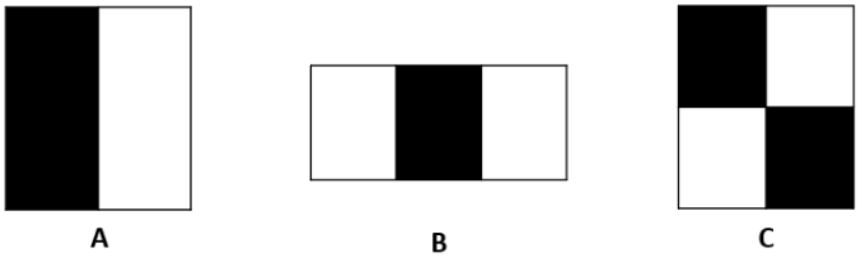

Рис. 2.3. Типи властивостей

Функції країв і ліній корисні для виявлення країв і ліній відповідно. Чотиристоронні елементи використовуються для знаходження діагональних елементів.

Значення функції обчислюється як одне число: сума значень пікселів у чорній області мінус сума значень пікселів у білій області. Значення дорівнює нулю для рівної поверхні, на якій усі пікселі мають однакове значення, і, отже, не надають корисної інформації.

Оскільки наші обличчя мають складну форму з темнішими та яскравішими плямами, функція, подібна до Хаара, дає вам велике число, коли області в чорних і білих прямокутниках сильно відрізняються. Використовуючи це значення, ми отримуємо частину дійсної інформації із зображення.

Щоб бути корисною, функція, подібна до Хаара, повинна давати вам велике число, тобто області в чорному та білому прямокутниках дуже відрізняються. Відомі функції, які дуже добре розпізнають людські обличчя (рис. 2.4).

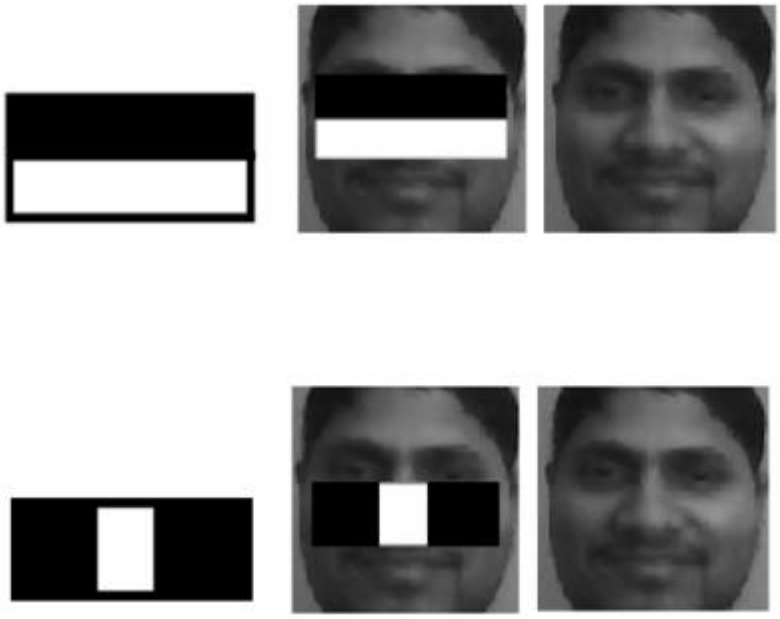

Рис. 2.4. Виявлення ознак обличчя

Наприклад, коли ми застосовуємо цю особливість Хаара до перенісся, ми отримуємо хорошу реакцію. Так само ми об’єднуємо багато з цих функцій, щоб зрозуміти, чи містить область зображення обличчя людини.

2.3.2. Інтегральні зображення для прискорення обчислення ознак

Інтегральне зображення — це структура даних для ефективного обчислення суми значень пікселів у вікні прямокутного зображення. Таким чином, властивості, подібні до Хаара, можуть бути обчислені дуже швидко, використовуючи інтегральне представлення зображення.

Для зображення I у відтінках сірого інтегральне значення зображення ii(x, y) у точці (x, y) є сумою всіх пікселів вище та ліворуч від (x, y), включно:

$$ii(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y'), \qquad (2.1)$$

Інтегральне зображення можна обчислити за один прохід над зображенням I за допомогою такого рівняння:

$$ii(x,y) = I(x,y) + ii(x,y-1) + ii(x-1,y) - ii(x-1,y-1), \qquad (2.2)$$

Для зображення з n пікселями часова складність обчислення інтегрального зображення становить O(n).

Можна продемонструвати, що сума в будь-якій прямокутній області вимагає чотирьох значень інтегрального зображення, незалежно від розміру вікна (рис. 2.5).

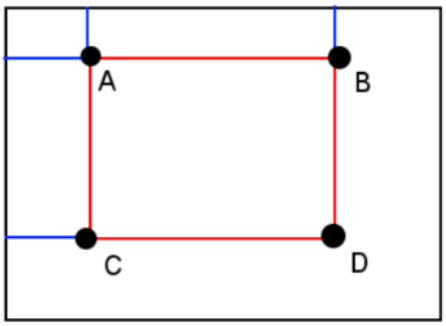

Рис. 2.5. Прямокутна область інтегрального зображення

Точніше, сума значень пікселів у прямокутнику ABCD, показаному вище, може бути обчислена як:

$$\sum_{\substack{x_0 < x \le x_1 \\ y_0 < y \le y_1}} I(x,y) = ii(D) + ii(A) - ii(B) - ii(C), \qquad (2.3)$$

Отже, використовуючи цю формулу, часова складність обчислення подібної функції Хаара становить O(1).

2.3.3. Навчання AdaBoost для вибору функцій

Далі ми використовуємо алгоритм машинного навчання, відомий як AdaBoost. Але навіщо нам взагалі потрібен алгоритм?

Кількість функцій, присутніх у вікні детектора 24×24, становить майже 160 000, але лише деякі з них важливі для ідентифікації обличчя. Тому ми використовуємо алгоритм AdaBoost, щоб визначити найкращі функції серед 160 000 функцій.

В алгоритмі Віоли-Джонса кожна властивість, подібна до Хаара, представляє слабкого учня. Щоб визначити тип і розмір функції, яка входить до остаточного класифікатора, AdaBoost перевіряє продуктивність усіх класифікаторів, які ви надаєте йому.

Щоб обчислити продуктивність класифікатора (рис. 8), ви оцінюєте його на всіх під областях усіх зображень, які використовуються для навчання. Деякі субрегіони дадуть сильну відповідь у класифікаторі. Вони будуть класифіковані як позитивні, тобто класифікатор вважає, що вони містять людське обличчя. На думку класифікаторів, субрегіони, які не дають сильної відповіді, не містять людського обличчя. Вони будуть класифіковані як негативні.

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Рис. 2.6. Обчислення продуктивності класифікатора

Класифікаторам, які показали хороші результати, надається більша важливість або вага. Кінцевим результатом є сильний класифікатор, також званий посиленим класифікатором, який містить найефективніші слабкі класифікатори.

Отже, коли ми навчаємо AdaBoost визначати важливі функції, ми передаємо йому інформацію у формі навчальних даних, а потім навчаємо його навчатися на

основі інформації для прогнозування. Таким чином, зрештою, алгоритм встановлює мінімальний поріг, щоб визначити, чи можна щось класифікувати як корисну функцію чи ні.

2.3.4. Каскад класифікаторів для швидкого відхилення вікон без облич

Можливо, AdaBoost нарешті вибере найкращі функції, скажімо, 2500, але обчислення цих функцій для кожного регіону все одно займає багато часу. У нас є вікно розміром 24 × 24, яке ми ковзаємо по вхідному зображенню, і нам потрібно знайти, чи є в одній із цих областей обличчя. Завдання каскаду полягає в тому, щоб швидко відкидати не-обличчя та уникати втрати дорогоцінного часу та обчислень. Таким чином, досягається швидкість, необхідна для виявлення обличчя в реальному часі.

Ми встановили каскадну систему (рис 2.3.5), у якій ми розділили процес ідентифікації обличчя на кілька етапів. На першому етапі у нас є класифікатор, який складається з наших найкращих ознак, іншими словами, на першому етапі субрегіон проходить через найкращі ознаки, такі як ознака, яка ідентифікує перенісся, або та, яка ідентифікує очі. На наступних етапах у нас є всі інші функції.

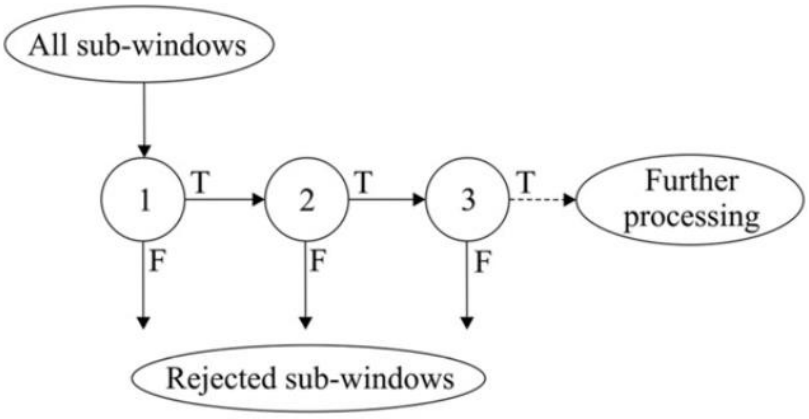

Рис 2.7. Каскад класифікаторів

Коли субрегіон зображення потрапляє в каскад, він оцінюється першим етапом. Якщо цей етап оцінює субрегіон як позитивний, тобто він вважає себе обличчям, результат етапу – це можливо.

Коли субрегіон отримує можливо, він переходить до наступного етапу каскаду, і процес продовжується, доки ми не досягнемо останнього етапу.

Якщо всі класифікатори схвалюють зображення, воно остаточно класифікується як людське обличчя та представляється користувачеві як виявлення.

Тепер як це допомагає нам збільшити нашу швидкість? Загалом, якщо перший етап дає негативну оцінку, то зображення негайно відкидається як таке, що не містить людського обличчя. Якщо він пройшов перший етап, але не пройшов другий етап, він також викидається. По суті, зображення може бути відкинуто на будь-якому етапі класифікатора.

2.4. Огляд алгоритму власних поверхонь

В основі методу власних поверхонь (Eigenfaces) лежить аналіз головних компонентів (PCA). Eigenfaces та PCA були використані Сіровичем та Кірбі для ефективного представлення зображень облич [31]. Вони почали з групи оригінальних зображень облич і розрахували найкращу векторну систему для стиснення зображень. Потім Терк і Пентланд застосували Eigenfaces для задачі розпізнавання облич. PCA є методом проекції на підпростір і широко використовується в розпізнаванні образів. Метою PCA є заміна корельованих векторів великої розмірності на некорельовані вектори меншої розмірності. Іншою метою є обчислення базису для набора даних. Основними перевагами PCA є низька чутливість до шума, зменшення вимог до пам'яті та пропускної здатності, а також підвищення ефективності за рахунок роботи у просторі меншої розмірності. Стратегія методу власних поверхонь полягає у виділенні характерних ознак на обличчі та представленні даного обличчя у вигляді лінійної комбінації так званих "власних поверхонь", отриманих в результаті процесу виділення ознак. Обчислюються головні компоненти облич у навчальній вибірці. Розпізнавання

здійснюється за допомогою проекції обличчя на простір, утворений власними поверхнями. Проводиться порівняння на основі евклідової відстані власних векторів поверхонь і власної поверхні зображення, що розпізнається. Якщо ця відстань досить мала, то особа ідентифікується. З іншого боку, якщо відстань занадто велика, то зображення розцінюється як таке, що належить людині, на яке систему необхідно натренувати. Блок-схема алгоритму наведена на рис. 2.8.

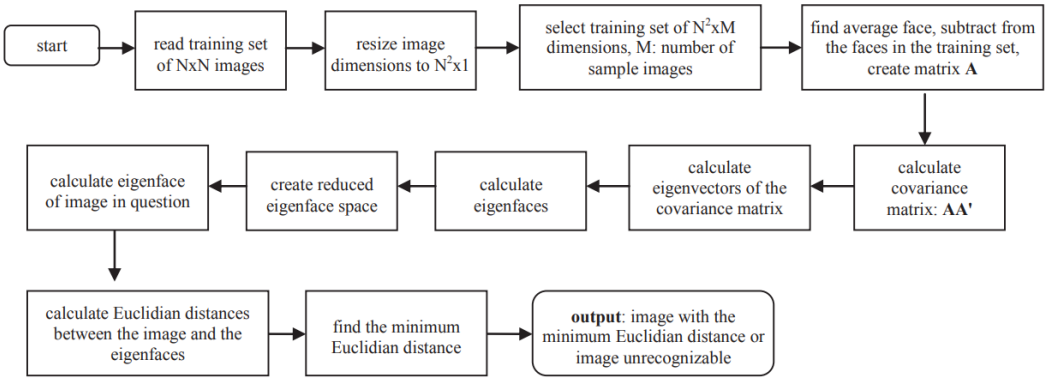

Рис. 2.8. Блок-схема алгоритму власних поверхонь

Спочатку зчитуються навчальні зображення розмірності $N * N$, які перетворюються в розмірність $N^2 * 1$. Таким чином створюється навчальний набір розмірністю $N^2 * M$, де M - кількість зображень-зразків. Середнє значення набору зображень обчислюється як:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i \,, \qquad (2.4)$$

де Ψ - середнє зображення, M - кількість зображень, Γ_i - вектор зображення.

Власні поверхні, що відповідають найбільшим власним значенням, зберігаються. Ці власні грані визначають лицевий простір. Власний простір створюється шляхом проектування зображення на лицевий простір, утворений власними поверхнями. Таким чином обчислюються вагові вектори. Розміри зображення коригуються відповідно до специфікацій, і зображення покращується на етапах попередньої обробки розпізнавання. Порівнюються ваговий вектор зображення та вагові вектори облич у базі даних.

Обчислюється середнє значення обличчя і віднімається від кожного обличчя в навчальній вибірці. За результатами операції віднімання формується матриця (A). Різниця між кожним зображенням та середнім зображенням обчислюється як:

$$\phi_i = \Gamma_i - \Psi,\ i = 1,2, \dots, M\ , \qquad (2.5)$$

де ϕ_i різниця між зображенням та середнім зображенням.

Матриця, отримана операцією віднімання (A), множиться на її транспонування і таким чином формується коваріаційна матриця дисперсій C:

$$C = A^T A,\ i = 1,2, \dots, M\ , \qquad (2.6)$$

де A утворено різницевими векторами, тобто:

$$A = [\phi_1, \phi_2, \dots, \phi_i]\ , \qquad (2.7)$$

Розмірність матриці C дорівнює N*N. Для формування C використовується M зображень. На практиці розмірність C дорівнює N*M. З іншого боку, оскільки ранг A дорівнює M, то з N власних векторів тільки M ненульові.

Обчислюються власні значення коваріаційної матриці.

Створюються власні поверхні, використовуючи кількість навчальних зображень мінус кількість класів (загальна кількість людей) власних векторів.

Відібраний набір власних векторів множиться на матрицю A для створення скороченого підпростору власних поверхонь.

Власні вектори з меншими власними значеннями відповідають меншим варіаціям коваріаційної матриці. Дискримінаційні ознаки обличчя при цьому зберігаються. Кількість власних векторів залежить від точності, з якою визначена база даних, і може бути оптимізована. Група виділених власних векторів називається власними гранями. Після отримання власних поверхонь зображення в базі даних проектуються в простір власних поверхонь і зберігаються ваги зображення в цьому просторі. Для визначення ідентичності зображення власні коефіцієнти порівнюються з власними коефіцієнтами в базі даних.

Формується власна поверхня зображення, що розглядається.

Обчислюються евклідові відстані між власним обличчям зображення та власними обличчями, збереженими раніше.

Особа, яка ідентифікується, визначається як така, евклідова відстань якої є мінімально меншою за порогове значення в базі даних власних облич. Якщо всі розраховані евклідові відстані більші за порогове значення, то зображення є нерозпізнаваним.

Причини вибору методу власних поверхонь для розпізнавання облич:

- Незалежність від геометрії обличчя;
- Простота реалізації;
- Можливість реалізації в реальному часі навіть без спеціального апаратного забезпечення;
- Легкість і швидкість розпізнавання по відношенню до інших методів;
- Більш високий відсоток успішності в порівнянні з іншими методами.

Проблемою методу розпізнавання облич за власними обличчями є час обчислень. Якщо база даних велика, може знадобитися деякий час для отримання інформації про особу, яка розпізнається.

ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі були розглянуті існуючі методи за засоби реалізації системи, що може виконувати функції біометричної ідентифікації особи. Було з’ясовано, що необхідно чітко розрізняти процеси виявлення, обробки та розпізнавання облич як по відношенню до алгоритмів так і до кодової частини.

Кожен з методів має свої переваги та недоліки. Ідеального рішення в даний момент не існує, тому підхід завжди визначається ситуативно в залежності від поставленої задачі та обмежень. Серед найбільш поширених обмежень: дуже швидка обробка зображень (для систем, що працюють в реальному часі), використання мінімально можливого об’єму пам’яті (для систем поглибленої обробки та аналізу) та відносна простота обчислень (для систем зі значно обмеженим процесорним ресурсом).

Серед розглянутих методів біометричної ідентифікації за геометрією облич слід виділити підхід на основі 3D моделей. У сучасній практиці цей метод показує себе найбільш точним та безпечним. Таку ідентифікацію важко обійти сторонньому користувачеві (наприклад за допомогою фотографії чи відео), адже система перевіряє глибину зображення та поведінку під час авторизації.

Для виконання практичного завдання, на основі отриманих знань, був обраний метод Віоли-Джонса для виявлення обличчя та метод власних поверхонь для ідентифікації особи.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Система побудови проекту

Для створення програми у Visual Studio, все починається з проекту. У логічному сенсі проект містить усі файли, скомпільовані у виконуваний файл, бібліотеку, тощо. Ці файли можуть містити вихідний код, значки, зображення, файли даних тощо. Проект також містить параметри компілятора та інші конфігураційні файли, які можуть знадобитися різним службам або компонентам, з якими спілкується ваша програма.

Visual Studio використовує MSBuild [32] для створення кожного проекту в рішенні, і кожен проект містить файл проекту MSBuild. Розширення файлу відображає тип проекту, наприклад, проект C# (.csproj), проект Visual Basic (.vbproj) або проект бази даних (.dbproj).

MSBuild використовує відкритий та розширюваний формат файлів проекту на базі XML. Формат файлу проекту MSBuild дозволяє розробникам описувати створювані елементи, а також способи їх побудови для різних операційних систем та конфігурацій. Крім того, формат файлу проекту дозволяє розробникам створювати багаторазово використовувані правила збирання, які можна розкласти на окремі файли, щоб збирання могли виконуватися однаково в різних проектах у складі відповідного продукту.

Проект міститься в рішенні. Незважаючи на свою назву, рішення не є «відповіддю». Це просто контейнер для одного або кількох пов’язаних проектів разом із інформацією про збірку, налаштуваннями вікна Visual Studio та будь-якими іншими файлами, не пов’язаними з конкретним проектом.

Visual Studio використовує два типи файлів (.sln і .suo) для зберігання налаштувань для рішень:

1. .sln (Visual Studio Solution) упорядковує проекти та елементи проекту в рішенні.

2. .suo (Solution User Options) зберігає налаштування на рівні користувача, наприклад точки зупину.

Рішення додатка біометричної ідентифікації обличчя складається з одного проекту FaceRecognition.csproj, у якому міститься вся інформація про вихідні файли коду, використані бібліотеки, конфігурації тощо (рис 3.1).

```
  <Reference Include="System.Xml" />
  <Reference Include="Microsoft.CSharp" />
  <Reference Include="System.Core" />
  <Reference Include="System.Xml.Linq" />
  <Reference Include="System.Data.DataSetExtensions" />
  <Reference Include="System.Net.Http" />
  <Reference Include="System.Xaml">
    <RequiredTargetFramework>4.0</RequiredTargetFramework>
  </Reference>
  <Reference Include="WindowsBase" />
  <Reference Include="PresentationCore" />
  <Reference Include="PresentationFramework" />
</ItemGroup>
<ItemGroup>
  <ApplicationDefinition Include="App.xaml">
    <Generator>MSBuild:Compile</Generator>
    <SubType>Designer</SubType>
  </ApplicationDefinition>
  <Compile Include="Config.cs" />
  <Compile Include="FaceData.cs" />
  <Compile Include="WFAbout.xaml.cs">
    <DependentUpon>WFAbout.xaml</DependentUpon>
  </Compile>
  <Compile Include="WFFaceRecognition.xaml.cs">
    <DependentUpon>WFFaceRecognition.xaml</DependentUpon>
  </Compile>
  <Page Include="WFAbout.xaml">
    <SubType>Designer</SubType>
    <Generator>MSBuild:Compile</Generator>
  </Page>
  <Compile Include="App.xaml.cs">
    <DependentUpon>App.xaml</DependentUpon>
    <SubType>Code</SubType>
  </Compile>
  <Page Include="WFFaceRecognition.xaml">
    <SubType>Designer</SubType>
    <Generator>MSBuild:Compile</Generator>
  </Page>
</ItemGroup>
```

Рис 3.1. Частина змісту FaceRecognition.csproj

Файл FaceRecognition.sln зберігає загальну інформацію про всі проекти рішення, їх взаємодію між собою, глобальні налаштування компілятора під різні типи побудови (Debug, Release) та архітектури (рис 3.2).

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio Version 17
VisualStudioVersion = 17.3.32922.545
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "FaceRecognition", "FaceRecognition\FaceRecognition.csproj", "{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}"
EndProject
Global
	GlobalSection(SolutionConfigurationPlatforms) = preSolution
		Debug|Any CPU = Debug|Any CPU
		Debug|x64 = Debug|x64
		Release|Any CPU = Release|Any CPU
		Release|x64 = Release|x64
	EndGlobalSection
	GlobalSection(ProjectConfigurationPlatforms) = postSolution
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Debug|Any CPU.Build.0 = Debug|Any CPU
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Debug|x64.ActiveCfg = Debug|x64
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Debug|x64.Build.0 = Debug|x64
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Release|Any CPU.ActiveCfg = Release|Any CPU
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Release|Any CPU.Build.0 = Release|Any CPU
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Release|x64.ActiveCfg = Release|x64
		{8F3FAEC5-5BFE-47C6-88DB-B22E0DD2CA00}.Release|x64.Build.0 = Release|x64
	EndGlobalSection
	GlobalSection(SolutionProperties) = preSolution
		HideSolutionNode = FALSE
	EndGlobalSection
	GlobalSection(ExtensibilityGlobals) = postSolution
		SolutionGuid = {480C19C3-D42F-4796-8F8C-BBBD0698E5C0}
	EndGlobalSection
EndGlobal
```

Рис 3.2. Зміст FaceRecognition.sln

У міру того, як світ дедалі більше об’єднувався через Інтернет, розробники програмного забезпечення також ставали все більш взаємопов’язаними та обізнаними про роботу один одного. Як побічний ефект, спільне використання та повторне використання коду між проектами перетворилося з бажаного на норму та на необхідність.

Але повторне використання коду як збірок може бути виснажливим і складним процесом, особливо коли мова йде про великі проекти, які мають багато залежностей. Відстеження всіх необхідних залежностей, переконання, що вони доступні в усіх середовищах і з правильною версією, узгодження версій у дереві залежностей і керування каскадними залежностями може зайняти багато часу та бути схильним до помилок.

Для світу .NET рішення з’явилося в 2010 році, коли вперше було запущено NuGet. NuGet — це менеджер пакетів для .NET. Це рішення «програмне забезпечення як послуга», яке дозволяє розробникам створювати, ділитися та використовувати корисний код у формі «пакетів».

Пакети NuGet — це не що інше, як zip-файли (з розширенням .nupkg), які містять скомпільований код (файли .dll) та інші файли, пов'язані з цим кодом, включаючи маніфест.

Для підключення EmguCV бібліотеки в проект було застосовано саме інструмент NuGet Package Manager [33], що дозволяє швидко інтегрувати існуючі рішення у власний додаток. На рис. 3.3 зображені усі використані NuGet пакети.

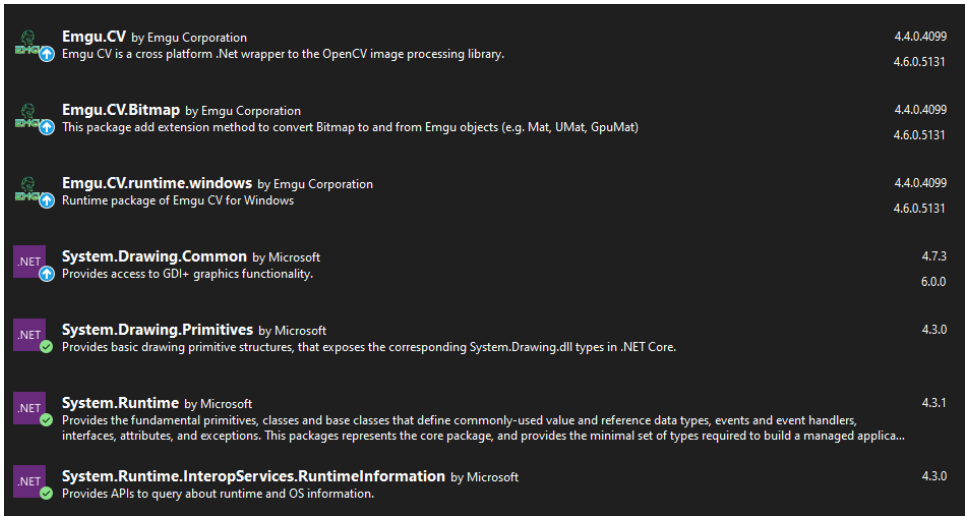

Рис 3.3. Зміст NuGet Package Manager'а

Загалом структура проекту є досить невеликою і простою, так як додаток є лише рішенням однієї функції великих систем програмного забезпечення (рис 3.4).

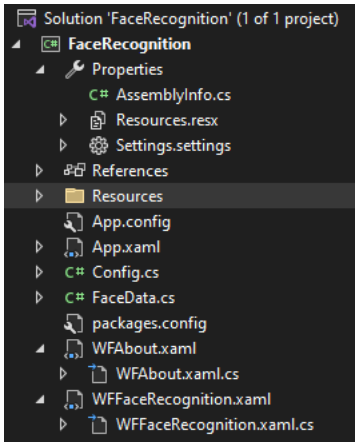

Рис 3.4. Структура проекту

3.2. Реалізація графічного інтерфейсу

WPF (Windows Presentation Foundation) — це технологія Microsoft, яка дозволяє створювати розширені інтерфейси користувача для настільних програм Windows. WPF було представлено разом із платформою .NET версії 3.0 як альтернативна технологія Windows Forms [34].

WPF має на меті дозволити створювати набагато багатший інтерфейс користувача, ніж можна розробити за допомогою Windows Forms. Щоб досягти цього, WPF повністю змінює базову технологію відображення. Тоді як Windows Forms базувалася на піксельному інтерфейсі графічного пристрою (GDI), WPF використовує векторні системи DirectX (рис. 3.5).

Оскільки більшість сучасних відеокарт підтримують DirectX, програми WPF дозволяють масштабувати та обертати текст і векторні зображення без пікселізації. Можна включити дво- та тривимірну векторну графіку та анімацію, яка автоматично використовує доступне апаратне прискорення для покращення продуктивності. Також можна використовувати WPF для бізнес-додатків, які не потребують таких графічних розквітів. Наприклад, Visual Studio 2010 і 2012 містять інтерфейси користувача, розроблені за допомогою WPF.

Чудовою особливістю WPF і програм на основі XAML загалом є розділення інтерфейсу користувача та іншого коду у програмному забезпеченні. Програми Windows Forms значною мірою залежать від подій, а обробники подій прив’язані до візуальних елементів керування. WPF розділяє код XAML, який визначає дизайн програми, і фоновий код, який ви створюєте за допомогою мови .NET, наприклад C#.

Хоча можна підключити код для керування подіями. Більш поширеним є використання переваг потужних технологій прив’язки даних і команд XAML. Це дозволяє вільно посилатися на властивості та методи за назвою в атрибутах XML. Під час виконання, елементи керування налаштовуються відповідно до названих властивостей. Кнопки та інші елементи керування пов’язані з командами, які

виконуються у відповідь на дії користувача, при цьому стан пов’язаних елементів керування змінюється автоматично залежно від доступності команд.

Дизайн і код настільки слабо пов’язані, що над окремими елементами можуть працювати різні люди. Наприклад, розробник може створювати класи, що містять властивості для прив’язки та команди для виконання, не знаючи дизайну інтерфейсу користувача. Маючи доступні мінімальні деталі базового коду, дизайнер інтерфейсу користувача може створити XAML окремо для розробника, щоб пізніше інтегрувати його в проект. Microsoft навіть створила окремі інструменти для цих двох ролей. Розробникам рекомендується використовувати Visual Studio, тоді як дизайнери мають інструменти Expression Studio, включаючи Blend.

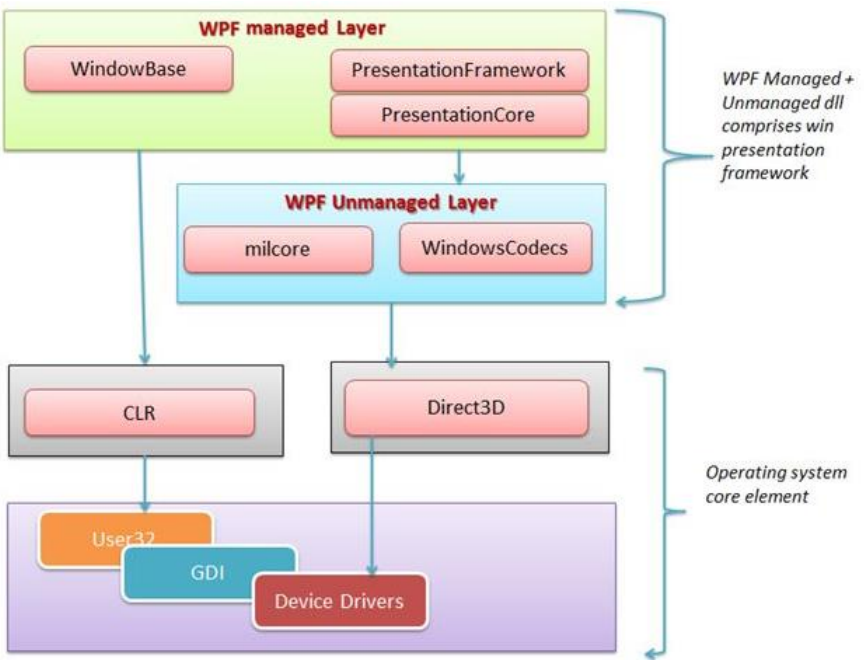

Рис. 3.5. Архітектура WPF

Для цілей додатка біометричної ідентифікації за допомогою інструменту WPF було розроблене головне вікно інтерфейсу користувача (рис. 3.6).

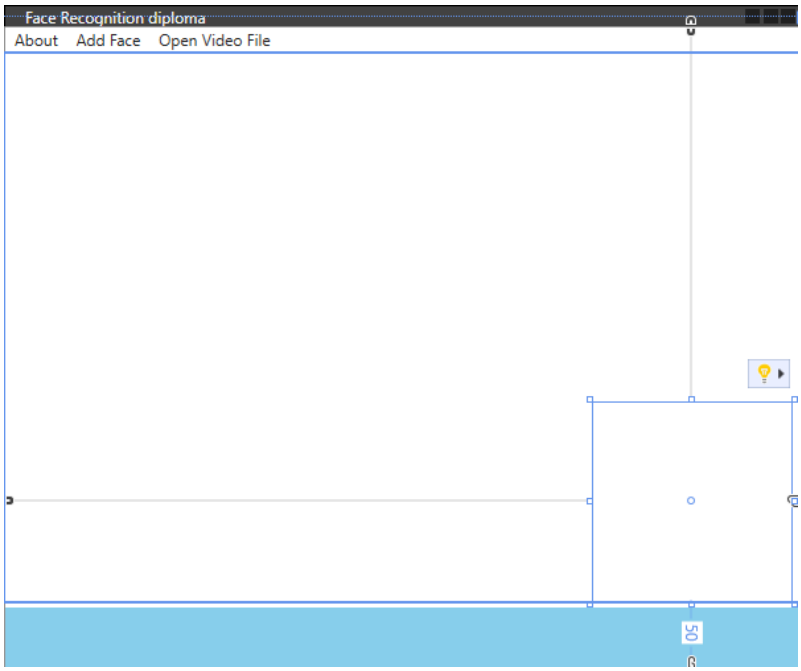

Рис. 3.6. Головне вікно додатка

Головне вікно додатка розподілене на чотири зони:

1. Інструменти керування стандартного системного вікна.
2. Інструменти керування додатка.
3. Вихідний потік даних з відеокамери або відео.
4. Відео потік розпізнаного обличчя.
5. Стрічка результату обробки зображення.

Відповідний XAML опис вікна зображено на рис. 3.7.

```
<Window x:Class="FaceDetectionAndRecognition.WPFaceRecognition"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:FaceRecognition"
        mc:Ignorable="d"
        Title="Face Recognition diploma" Height="500" Width="600" Loaded="Window_Loaded" WindowStartupLocation="CenterScreen">
    <Grid>
        <Image x:Name="imgCamera"
               HorizontalAlignment="Stretch"
               VerticalAlignment="Stretch"
               Width="Auto"
               Height="Auto"
               Margin="0,20,0,50" />
        <Label x:Name="lblFaceName"
               HorizontalAlignment="Stretch"
               Width="Auto"
               Margin="0,414,0,0"
               HorizontalContentAlignment="Center"
               Background="#87CEEB"
               VerticalAlignment="Bottom"
               Height="45" FontSize="22" Foreground="White"/>
        <Image x:Name="imgDetectFace"
               HorizontalAlignment="Right"
               Height="150"
               Margin="432,269,5,50"
               VerticalAlignment="Bottom"
               Width="150"/>
        <Menu x:Name="menu"
              HorizontalAlignment="Stretch"
              Height="20"
              VerticalAlignment="Top"
              Width="Auto"
              Background="{x:Null}" >
            <MenuItem Cursor="Hand" Header="About" Click="AboutButton_Click" />
            <MenuItem Cursor="Hand" Header="Add Face" Click="NewFaceButton_Click"/>
            <MenuItem Cursor="Hand" Header="Open Video File" Click="OpenVideoFile_Click"/>
        </Menu>
    </Grid>
</Window>
```

Рис 3.7. Вихідний код розмітки головного вікна

Розроблений інтерфейс додатка є достатньою, спрощеною версією реального інтерфейсу користувача, який необхідний для мануального тестування функціоналу програми. Більш деталізований та багатий інтерфейсі немає доцільності при виконанні задачі дипломного проекту.

3.3. Реалізація логіки додатка

Перш ніж розглядати код логіки програми слід визначити основні функціональні блоки та їх послідовність. Всю логіку додатку можна поділити на п’ять функціональних блоків (рис. 3.8):

1. Захоплення відео кадру.
2. Виявлення обличчя.
3. Кадрування обличчя.
4. Масштабування обличчя.

5. Обчислення значення схожості обличчя.

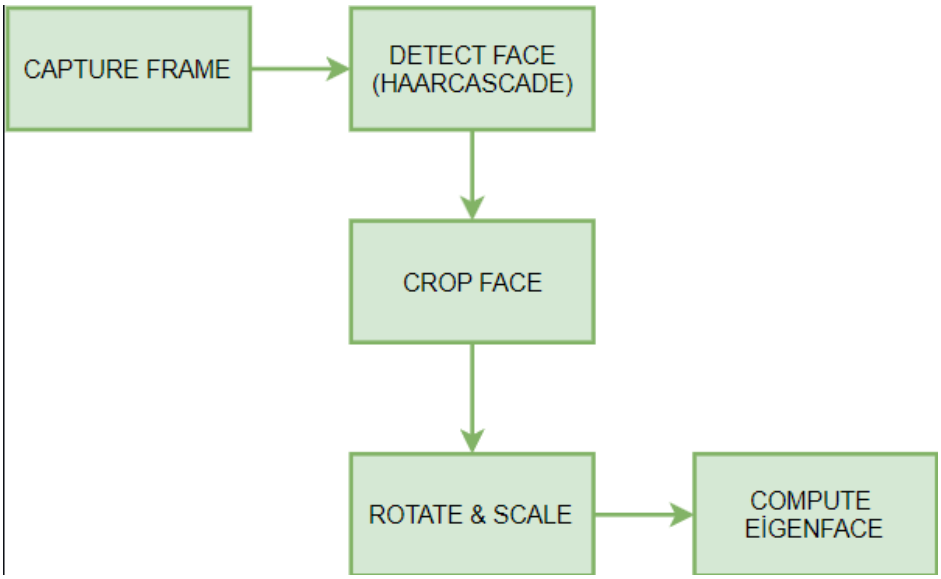

Рис. 3.8. Функціональна схема додатку розпізнавання обличчя

Послідовність функціональних блоків відповідає нумерації їх перечислення та додатково зображена на рис. 3.8.

В першу чергу при запуску програми відбувається конструкція та ініціалізація об’єктів інтерфейсу користувача. На рисунку 3.9 зображено виклик базового методу ініціалізації компонентів інтерфейсу та задання частоти обновлення таймера кадрів.

```
public WFFaceRecognition()
{
    InitializeComponent();
    captureTimer = new Timer()
    {
        Interval = Config.TimerResponseValue
    };
    captureTimer.Elapsed += CaptureTimer_Elapsed;
}
```

Рис. 3.9. UI ініціалізація

Наступним кроком є ініціалізація об’єктів, що безпосередньо беруть участь у логіці програми. На рисунку 3.10 зображено функцію обробки події завантаження головного вікна, що відбувається при запуску системи. За допомогою методу GetFacesList() в оперативну пам’ять завантажуються існуючі данні про відомі

обличчя, після чого створюється об’єкт VideoCapture() з індексом 0, що означає використовувати системний девайс камери за замовчуванням.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    GetFacesList();
    videoCapture = new VideoCapture(Config.ActiveCameraIndex);
    videoCapture.SetCaptureProperty(CapProp.Fps, 30);
    videoCapture.SetCaptureProperty(CapProp.FrameHeight, 450);
    videoCapture.SetCaptureProperty(CapProp.FrameWidth, 370);
    captureTimer.Start();
}
```

Рис. 3.10. Функція Window_Loaded

У функції GetFacesList() спочатку відбувається ініціалізація каскадного класифікатора Хаара за допомогою haarcascade_frontalface_default.xml файла (рис. 3.11), що лежить в ресурс папці проекту. Цей файл містить у собі набір констант, що допомагають налаштувати класифікатор на роботу по виявленню облич.

```
public void GetFacesList()
{
    //haar cascade classifier
    if (!File.Exists(Config.HaarCascadePath))
    {
        string text = "Cannot find Haar cascade data file:\n\n";
        text += Config.HaarCascadePath;
        MessageBoxResult result = MessageBox.Show(text, "Error",
                MessageBoxButton.OK, MessageBoxImage.Error);
    }

    haarCascade = new CascadeClassifier(Config.HaarCascadePath);
```

Рис. 3.11. Ініціалізація каскадного класифікатора Хаара

Далі у функції GetFacesList() відбувається вичитка збережених зображень з відповідними до них назвами. Усі ці данні надалі зберігаються в оперативній пам’яті додатку та передаються на обробку в EigenFaceRecognizer (рис. 3.12). На основі вичитаних зображень, об’єкт розпізнавача надалі зможе порівнювати вхідні зображення з вже існуючими і видавати результат на UI.

```
    StreamReader reader = new StreamReader(Config.FaceListTextFile);
    int i = 0;
    while ((line = reader.ReadLine()) != null)
    {
        string[] lineParts = line.Split(':');
        faceInstance = new FaceData();
        faceInstance.FaceImage = new Image<Gray, byte>(Config.FacePhotosPath + lineParts[0] + Config.ImageFileExtension);
        faceInstance.PersonName = lineParts[1];
        faceList.Add(faceInstance);
    }
    foreach (var face in faceList)
    {
        imageList.Push(face.FaceImage.Mat);
        nameList.Add(face.PersonName);
        labelList.Push(new[] { i++ });
    }
    reader.Close();

    // Train recogniser
    if (imageList.Size > 0)
    {
        recognizer = new EigenFaceRecognizer(imageList.Size);
        recognizer.Train(imageList, labelList);
    }
}
```

Рис. 3.12. Навчання EigenFace розпізнувача облич

Відповідно до проініціалізованого значення частоти оновлення таймеру з певною періодичністю (500 мс.), постійно викликається функція обробки події таймеру (рис. 3.13), що містить у собі виклик методу ProcessFrame() (рис. 3.14).

```
private void CaptureTimer_Elapsed(object sender, ElapsedEventArgs e)
{
    ProcessFrame();
}
```

Рис. 3.13. Функція обробки події таймера

Функція ProcessFrame() обробляє кадр захоплений джерелом відео (рис. 3.14). Спочатку RGB кадр повністю конвертується в чорно-біле зображення. Після чого, за допомогою каскадного класифікатора Хаара, на чорно-білому кадрі відбувається пошук обличчя. Якщо обличчя знайдене, то воно підкреслюється прямокутною рамкою і дублюється в detectedFace контейнер. По завершенню логіки виявлення обличчя, кадр передається на обробку в метод FaceRecognition() (рис. 3.15).

```
private void ProcessFrame()
{
    bgrFrame = videoCapture.QueryFrame().ToImage<Bgr, Byte>();

    if (bgrFrame != null)
    {
        try
        {
            Image<Gray, byte> grayframe = bgrFrame.Convert<Gray, byte>();

            Rectangle[] faces = haarCascade.DetectMultiScale(grayframe, 1.2, 10, new System.Drawing.Size(50, 50), new System.Drawing.Size(200, 200));

            FaceName = "No face";
            foreach (var face in faces)
            {
                bgrFrame.Draw(face, new Bgr(175, 255, 210), 2);
                detectedFace = bgrFrame.Copy(face).Convert<Gray, byte>();
                FaceRecognition();
                break;
            }
            CameraCapture = bgrFrame.ToBitmap();
        }
        catch (Exception)
        {
            // TODO log
        }

    }
}
```

Рис. 3.14. Функція обробки отриманого зображення

Метод FaceRecognition() (рис. 3.15) безпосередньо звертається до зарання навченого об’єкту EigenFaceRecognizer і робить запит на розпізнавання обличчя. Результатом розпізнавання є фото, яке видало найбільше eigen число при обробці. Після співставлення картинки та обличчя, відповідне ім’я результату записується у властивість FaceName.

```
private void FaceRecognition()
{
    if (imageList.Size != 0)
    {
        // Eigen Face Algorithm
        FaceRecognizer.PredictionResult result = recognizer.Predict(detectedFace.Resize(100, 100, Inter.Cubic));
        FaceName = nameList[result.Label];
        CameraCaptureFace = detectedFace.ToBitmap();
    }
    else
    {
        FaceName = "Please Add Face";
    }
}
```

Рис. 3.15. Функція розпізнавання обличчя

Слід окремо розглянути функцію додавання обличчя в постійне сховище. При натисканні на кнопку NewFace автоматично спрацьовує функція обробки сигналу натискання (рис. 3.16), у якій останнє виявлене обличчя зберігається у FacePhotosPath директорії з відповідною назвою, що запитується в користувача.

```
private void NewFaceButton_Click(object sender, RoutedEventArgs e)
{
    if (detectedFace == null)
    {
        MessageBox.Show("No face");
        return;
    }
    //Save detected face
    detectedFace = detectedFace.Resize(100, 100, Inter.Cubic);
    detectedFace.Save(Config.FacePhotosPath +"face"+ (faceList.Count + 1) + Config.ImageFileExtension);
    StreamWriter writer = new StreamWriter(Config.FaceListTextFile, true);
    string personName = Microsoft.VisualBasic.Interaction.InputBox("Your Name");
    writer.WriteLine(String.Format("face{0}:{1}", (faceList.Count + 1), personName));
    writer.Close();
    GetFacesList();
}
```

Рис. 3.16. Функція додавання нового обличчя

Щодо обробки відео файлів, то немає ніякої різниці звідки йде вхідний потік відео даних чи то з веб камери чи з записаного mp4 файла, тому додатково була додана функція виявлення і розпізнавання облич на збережених відео (рис. 3.17). Для цієї функціональності достатньо лише змінити вхідний потік даних, тому в обробці сигналу натискання кнопки відкриття відео файлу, перестворюємо об’єкт VideoCapture з орієнтацією на відео ресурс.

```
private void OpenVideoFile_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog openDialog = new OpenFileDialog();
    if (openDialog.ShowDialog().Value == true)
    {
        captureTimer.Stop();
        videoCapture.Dispose();

        videoCapture = new VideoCapture(openDialog.FileName);
        captureTimer.Start();
        this.Title = openDialog.FileName;
        return;
    }
}
```

Рис. 3.17. Функція зміни джерела вхідного потоку відео

3.4. Реалізація алгоритму власних поверхонь

Для порівняння з реалізацією Eigenface алгоритму бібліотеки EmguCV (OpenCV), була створена власна імплементація відповідно до розділу 2.4. Кодова база є доволі великою, тому будуть розглянуті тільки основні функції алгоритму.

Перш за все створюємо вектор-матрицю стовпців (рис. 3.18). Розміри матриці — це кількість зображень * (висота зображення * ширина зображення — кількість байтів).

```
private void CreateColumnVectorMatrix()
{
    var matrixHeightRef = (byte[])_vectorSet[0];
    _columnMatrixHeight = matrixHeightRef.Length;
    _columnMatrixWidth = _vectorSet.Count;

    _columnVectorMatrix = new byte[_columnMatrixWidth, _columnMatrixHeight];

    for (int i = 0; i < _vectorSet.Count; i++)
    {
        var vector = (byte[])_vectorSet[i];
        for (int j = 0; j < vector.Length; j++)
        {
            byte color = vector[j];
            _columnVectorMatrix[i, j] = color;
        }
    }
}
```

Рис. 3.18. Функція CreateColumnVectorMatrix

Далі обчислюємо середній вектор ColumnVectorMatrix. Для цього оцінюємо середнє значення кожного рядка (рисю 3.19).

```
private void CalculateMeanOfColumMatrix()
{
    _meanVector = new byte[_columnMatrixHeight];

    for (var i = 0; i < _columnMatrixHeight; i++)
    {
        var sum = 0;
        for (var j = 0; j < _columnMatrixWidth; j++)
        {
            byte color = _columnVectorMatrix[j, i];
            sum += color;
        }
        var avg = 0;

        avg = sum / _columnMatrixWidth;
        _meanVector[i] = (byte)avg;
    }
}
```

Рис. 3.19. Функція CalculateMeanOfColumMatrix

Віднімаємо вектор середнього значення від ColumnVectorMatrix (рис 3.20).

```
private void SubstractMeanVectorFromVectorFaceMatrix()
{
    for (var j = 0; j < _columnMatrixWidth; j++)
    {
        for (var i = 0; i < _columnMatrixHeight; i++)
        {
            var subtractedValue = _columnVectorMatrix[j, i] - _meanVector[i];
            if (subtractedValue < 0) subtractedValue = 0;
            _columnVectorMatrix[j, i] = (byte)subtractedValue;
        }
    }

}
```

Рис. 3.20. Функція SubstractMeanVectorFromVectorFaceMatrix

Перетворюємо ColumnVectorMatrix (тип byte) у подвійний масив на основі кольорів (тип double) (рис 3.21).

```
private void ConvertColorToDoubleMatrix()
{
    _columnVectorIntMatrix = new double[_columnMatrixWidth, _columnMatrixHeight];
    for (var i = 0; i < _columnMatrixWidth; i++)
    {
        for (var j = 0; j < _columnMatrixHeight; j++)
        {
            _columnVectorIntMatrix[i, j] = (double)_columnVectorMatrix[i, j];
        }
    }
}
```

Рис. 3.21. Функція ConvertColorToDoubleMatrix

Створюємо коваріаційну матрицю з використанням бібліотеки ILNumerics (рис. 3.22).

```
private void CreateCovarianceMatrix()
{
    _a = _columnVectorIntMatrix;
    _covariance = ILMath.multiply(_a.T, _a);
}
```

Рис. 3.22. Функція CreateCovarianceMatrix

Викликаємо декомпозицію сингулярного значення для коваріаційної матриці з використанням бібліотеки ILNumerics (рис. 3.23).

```
private void CreateSvdOnCorrelation()
{
    _s = ILMath.svd(_covariance, ref _v, ref _u);
}
```

Рис. 3.23. Функція CreateSvdOnCorrelation

Обчислюємо власні значення для кожного зображення (рис. 3.24).

```
private void CalculateEigenValues()
{
    this._eigenValues = new double[25];
    _s.Diagonal.ExportValues(ref this._eigenValues);
    var eigenValuesPow = new double[_eigenValues.Length];

    for (var i = 0; i < eigenValuesPow.Length; i++)
    {
        eigenValuesPow[i] = (double)Math.Pow((double)_eigenValues[i], -0.5);
    }
    this._eigenVectors.Clear();

    for (var i = 0; i < eigenValuesPow.Length; i++)
    {
        var oneEigenVector = (ILArray<double>)_v.Subarray(new string[]
            {
                        ":",
                        i.ToString()
            });
        var oneFace = ILMath.multiply(_a, oneEigenVector);
        var oneFace2 = ILMath.multiply(oneFace, eigenValuesPow[i]);

        //25 = magicnumber, just needed to initialize double array
        var eigenVectorArray = new double[25];
        oneFace2.ExportValues(ref eigenVectorArray);

        var distance = 0.0;
        for (var j = 0; j < eigenVectorArray.Length; j++)
        {
            distance += Math.Pow((double)eigenVectorArray[j], 2.0);
        }

        distance = Math.Sqrt(distance);
        for (var j = 0; j < eigenVectorArray.Length; j++)
        {
            eigenVectorArray[j] /= (float)distance;
        }
        this._eigenVectors.Add(eigenVectorArray);
    }
    this._eigenWeights.Clear();

    for (var i = 0; i < this._vectorSet.Count; i++)
    {
        var existingWeight = this.GetEigenWeight((byte[])this._vectorSet[i], this._eigenVectors.Count);
        this._eigenWeights.Add(existingWeight);
    }
}
```

Рис. 3.24. Функція CalculateEigenValues

Оцінюємо найближчу подібну грань для вхідного зображення. Результатом є зображення з найбільшою вагою (рис.3.25).

```
public Bitmap GetFaceForInput(string imageSource)
{
    var image = new Bitmap(Image.FromFile(imageSource));
    InputPic = image;

    double[] newWeight = this.GetEigenWeight(ImageManager.ConvertImageToVector(image), _eigenVectors.Count);
    var sortedWeights = new Collection<double>();
    var sortedPictures = new Collection<byte[]>();
    _foundFaces.Clear();
    for (var i = 0; i < this._trainingSet.Count; i++)
    {
        var distance = this.GetDistance(newWeight, this._eigenWeights[i]);

        //0.05 = best distance
        if (distance <= 0.05)
        {
            if (sortedWeights.Count == 0)
            {
                sortedWeights.Add(distance);
                sortedPictures.Add((byte[])_vectorSet[i]);
            }
            else
            {
                for (var j = 0; j < sortedWeights.Count; j++)
                {
                    if (distance < sortedWeights[j])
                    {
                        sortedWeights.Insert(j, distance);
                        sortedPictures.Insert(j, (byte[])_vectorSet[i]);
                        break;
                    }
                }
            }
        }
    }
    for (var i = 0; i < Math.Min(10, sortedWeights.Count); i++)
    {
        this._foundFaces.Add(sortedPictures[i]);
    }

    return ImageManager.ConvertVectorToImage(_foundFaces[0]);
}
```

Рис. 3.25. Функція GetFaceForInput

3.5. Мануальне тестування додатка

При запуску програми ідентифікації обличчя можемо бачити головне вікно програми (рис 3.26). Місце для виводу вхідного зображення з відеокамери пусте, так як веб камера не увімкнена.

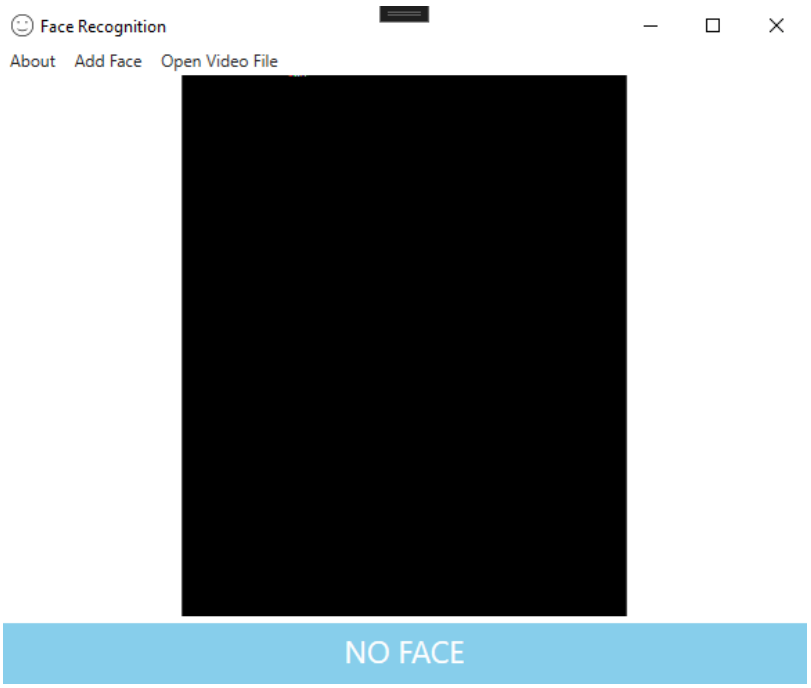

Рис. 3.26. Головне вікно додатку

Після підключення веб камери, додаток відразу ж виявляє фронтальне обличчя і пропонує додати його в сховище (рис. 3.27).

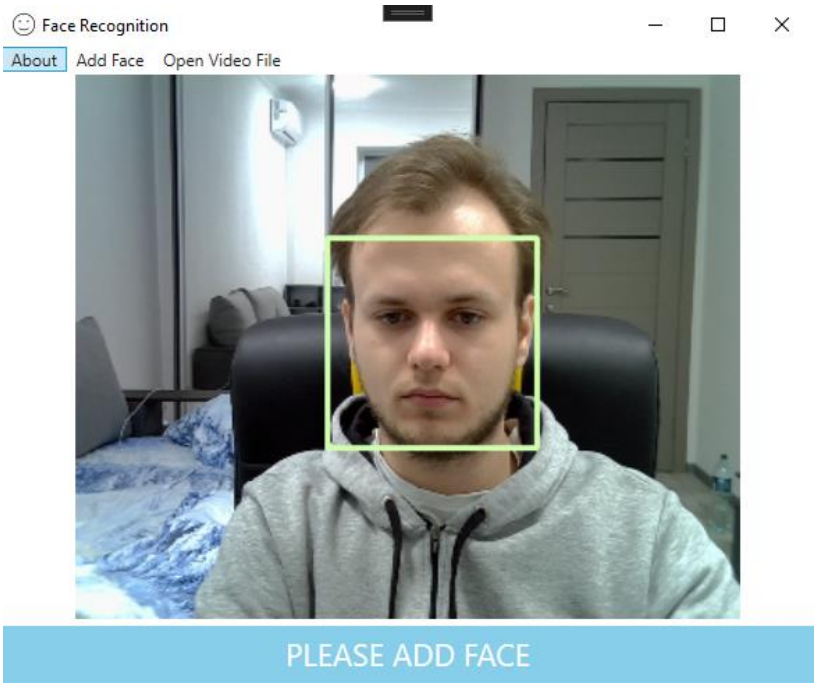

Рис. 3.27. Виявлення обличчя

При натисканні на кнопку “Add Face”, з’являється додаткове вікно для вводу імені зображення (рис. 3.28)

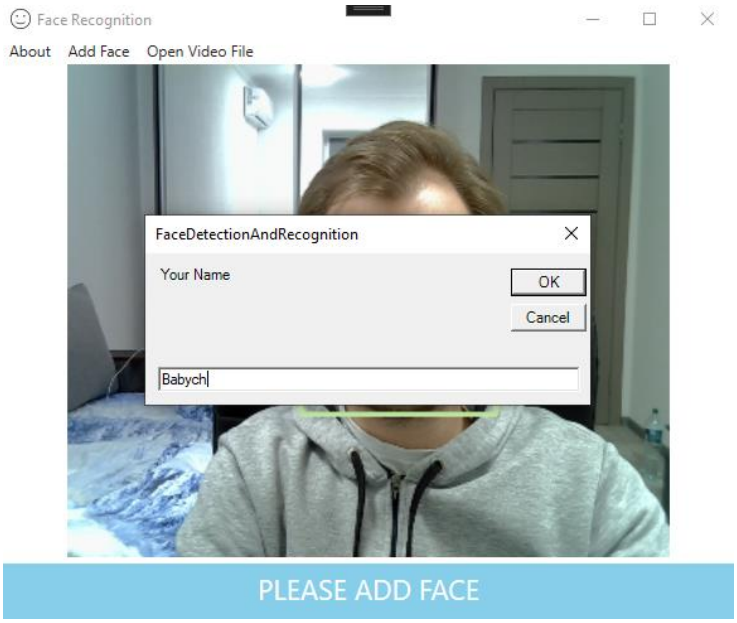

Рис. 3.28. Функція додавання обличчя

Після додавання зображення у постійне сховище, додаток вже починає розпізнавати обличчя, так як має мінімальні для цього данні (рис. 3.29).

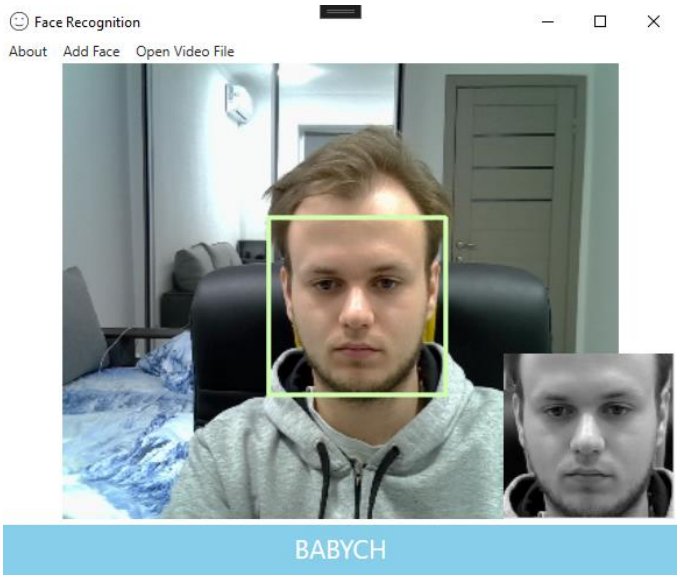

Рис. 3.29. Ідентифікація власного обличчя

Виводимо на веб камеру фотографію невідомої особи і як результат отримуємо повідомлення “can’t match”, що означає збігів не знайдено (рис. 3.30).

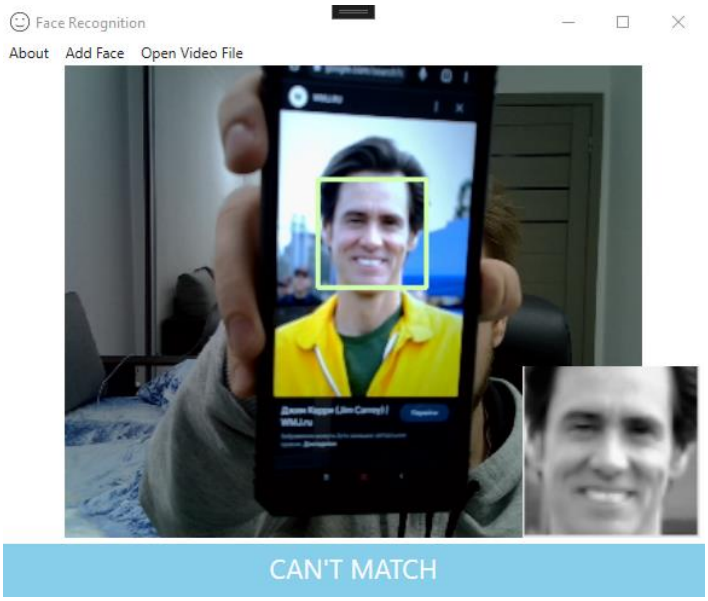

Рис. 3.30. Збігів не знайдено

Додаємо додаткове зображення. На цей раз це фотографія Джима Керрі (рис. 3.31).

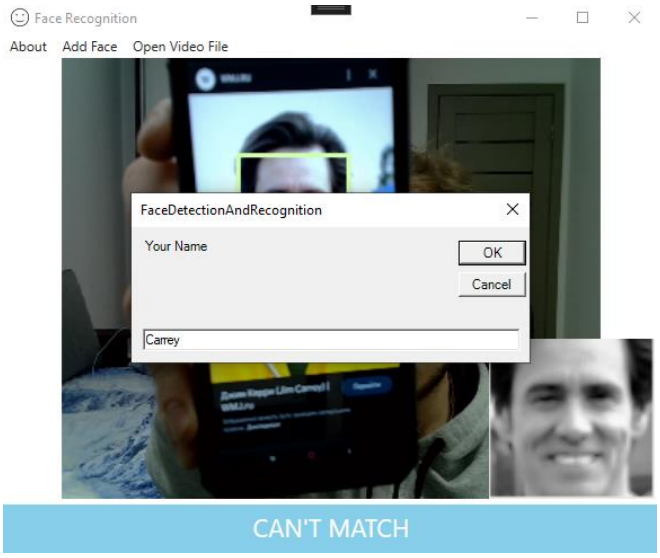

Рис. 3.31. Додавання нової особи

Після додавання інформації про невідому особу, додаток знову починає розуміти людину зображену на відео потоці (рис. 3.32).

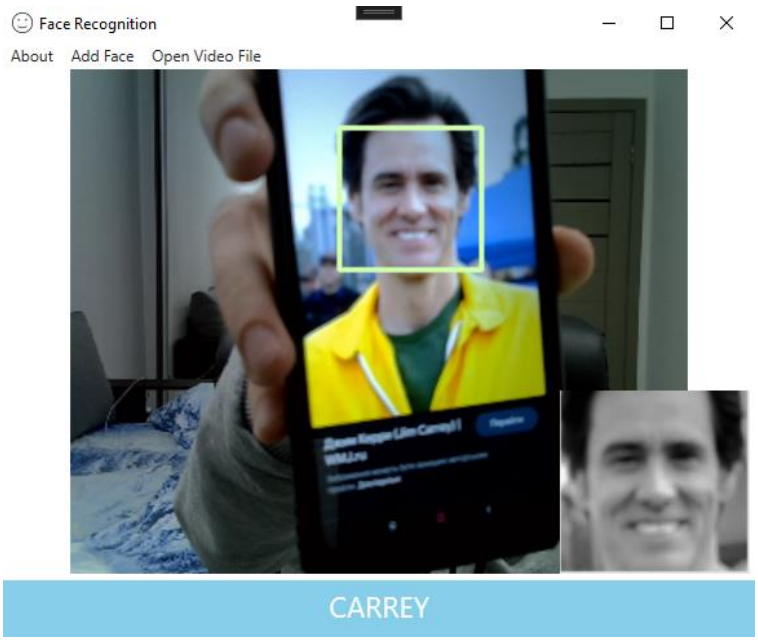

Рис. 3.32. Ідентифікація нової особи

Якщо виявлення обличчя в кадрі не знаходить збігів, то виводиться відповідне повідомлення “No face”, що означає обличчя не знайдене (рис. 3.33).

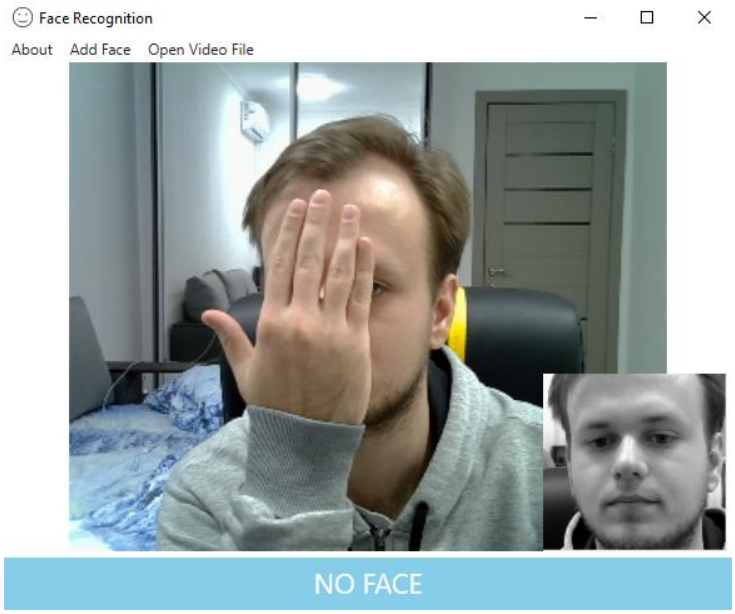

Рис. 3.33. Обличчя не виявлено при відео потоці

Спробуємо використати функцію “Open Video File”. Натискаємо на відповідну кнопку, та обираємо місце знаходження відео файлу. Заранє в постійне сховище

було додано інформацію про особу і як результат бачимо, що додаток успішно ідентифікує людину (рис. 3.34).

Рис. 3.34. Ідентифікація особи на відео

3.6. Аналіз отриманих результатів

Мануальне тестування програми пройшло успішно. Додаток виконує необхідну функцію, а саме виявлення та ідентифікацію особи за обличчям.

Головним недоліком даної реалізації є недостатня безпека. Система не відрізняє фотографію від реального користувача. Однак працює достатньо швидко і точно. Проблему фотографій можна вирішити за допомогою обробки декількох кадрів замість одного. Але знову ж таки, тепер постає питання, як вирішувати оману за допомогою відео. Це вже набагато складніше, доречніше було б вдатися до іншого методу реалізації біометричної ідентифікації, наприклад за 3D моделлю (розділ 2.2.4). Даний метод повинен чудово підійти для систем по збору інформації, так як є швидким і має відносно просту логіку для інтеграції. Інколи система дає похибку і може сплутати особи, однак це нормально, адже 100%-вих систем ідентифікації на даний момент не існує. Для підвищення точності знову ж таки

можна додати метод обробки декількох кадрів (наприклад середнє значення серед 10-ти кадрів), а також можна розширити постійне сховище і для кожної особи зберігати більше ніж одну фотографію для порівняння.

Додатково була протестована власна реалізація алгоритму власних поверхонь. Сам алгоритм є робочим, але повністю програє у порівнянні з існуючою реалізацією в EmguCV (OpenCV). Швидкість розпізнавання обличчя довша на ~25%, а артефактів хибної ідентифікації значно більше (приблизно 4 до 1). Швидкість алгоритму можна збільшити за допомогою розпаралелювання функцій роботи з матрицями, а точність підняти за рахунок зміни константних значень (наприклад distance) та виявлення залежностей негативних результатів

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі було розглянуто програмну реалізацію та тестування додатку ідентифікації особи за обличчям.

Стандартна система побудови проекту Microsoft Visual Studio показала себе дуже зручною у використанні, адже всі конфігураційні файли, що необхідні для компіляції або налаштування рішення генеруються автоматично відповідно до налаштувань у графічному інтерфейсі. Інструмент насправді потужний і може бути використаний у проектах будь-якого рівня складності.

Фреймворк WPF для побудови інтерфейсу користувача також відмінно зарекомендував себе. Серед переваг – це висока гнучкість до реалізації будь-яких UI елементів та ізольованість складової візуалізації від логіки самого додатку. Увесь користувацький інтерфейс описується за допомогою мови розмітки XAML, тому межі креативності додатку обмежуються лише уявою розробника. Якісна ізольованість UI складової дає можливість працювати дизайнеру та розробнику паралельно над проектом, що пришвидшує процес самої розробки.

Логіка додатку є доволі простою і це підкреслює, що дана реалізації є чудовим варіантом для PoC чи MVP проекту. Дане рішення за наявності певних навичок може бути інтегроване у проект менше ніж за добу. Ліцензії OpenCV та EmguCV не обмежують використання додатку у комерційних цілях. Рекомендується використовувати вже готові рішення алгоритмів, адже в таких відомих бібліотеках, вони пройшли велику кількість ітерацій оптимізації та модифікації.

Тестування додатку було лише мануальним, тому що проект є доволі специфічним і для якісної перевірки програми необхідно мати достатньо велику базу вирішених зображень.

ВИСНОВКИ

Темою дипломної роботи є біометрична ідентифікація за геометрією обличчя. Головна мета роботи це розробка та тестування додатка біометричної ідентифікації людей за геометрією їх облич.

Перед реалізацією проекту була ретельно проаналізована область застосування технології та її актуальність. Можна лише підкреслити необхідність та важливість даної теми в сьогоденності. Біометрична ідентифікації має великий потенціал у розвитку та ще знаходиться у стані активної фази розробки та покращення. Прогнозується, що вже у скорому майбутньому, люди зможуть відійти від платіжних карток та документів, що засвідчують особу і це лише один з напрямків розвитку біометричної ідентифікації.

Уже зараз існує величезна кількість алгоритмів виявлення та ідентифікації облич. Звісно немає ідеального варіанту який би влаштовував усі сфери застосування у швидкості виконання, використаної пам’яті, тощо. Головна методологія біометричної ідентифікації зараз – це раціональне застосування.

Однією з цілей дипломної роботи є перевірка життєздатності подібних самостійних рішень біометричної ідентифікації у стартап, PoC, MWP проектах. Можна підтвердити, що з відносною легкістю та без особливих обмежень ліцензій використання подібного рішення на начальних етапах розробки проекту є доцільним. Однак open-source рішення неконкурентоспроможні, адже у порівнянні з продуктами компаній, що спеціалізуються у даній сфері, такі рішення є низькоякісними.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Paul Viola, Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. Computer Vision and Pattern Recognition. – 2001. – Том 1. – С. 511-518.
2. 50 years of biometric research: Accomplishments, challenges, and opportunities. [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0167865515004365?via%3Dihub
3. Cyber researcher pulls public talk on hacking Apple's Face ID. [Електронний ресурс]. – Режим доступу: https://www.reuters.com/article/us-apple-cyber-conference-idUSKCN1OX1TA
4. AiCure Technologies participates at MHealth. [Електронний ресурс]. – Режим доступу: https://aicure.com/news/aicure-technologies-participates-at-mhealth/
5. Amazon’s Face Recognition Falsely Matched 28 Members of Congress With Mugshots. [Електронний ресурс]. – Режим доступу: https://www.aclu.org/news/privacy-technology/amazons-face-recognition-falsely-matched-28
6. About OpenCV. [Електронний ресурс]. – Режим доступу: https://opencv.org/about/
7. EmguCV Main Page. [Електронний ресурс]. – Режим доступу: https://www.emgu.com/wiki/index.php/Main_Page
8. Ming-Hsuan Yang, D.J. Kriegman, N. Ahuja. Detecting faces in images: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2002. – Том 24. – Випуск 1. – С. 34 – 58.
9. Juwei Lu, K.N. Plataniotis, A.N. Venetsanopoulos. Ensemble-based discriminant learning with boosting for face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligen. – 2006. – Том 17. – Вип. 1. – С. 166 – 178.

10. A.M. Abeer, W.L. Woo, S.S. Dlay. Multi-linear neighborhood preserving projection for face recognition. Pattern Recognition. – 2014. – Том 47. Вип. 2. – С. 544–555.
11. Juwei Lu, K.N. Plataniotis, A.N. Venetsanopoulos. Face recognition using kernel direct discriminant analysis algorithms. IEEE Trans. Neural Netw. – 2003. – Том 14. – Вип. 1. – С. 117–126.
12. G. Li, J. Zhang, Y. Wang, W.J. Freeman. Face recognition using a neural network simulating olfactory systems. Lect. Notes Comput. Sci. – 2006. – Том 3972. – С. 93–97.
13. W. Zhou, X. Pu, Z. Zheng. Parts-based holistic face recognition with RBF neural networks. Lect. Notes Comput. Sci. – 2006. – Том 3972. – С. 110–115.
14. J.S. Bhavin, D.L. Martin. Face recognition using localized features based on nonnegative sparse coding. Mach. Vis. Appl. – 2007. – Том 18. – Вип. 2. – С. 107–122.
15. W.C. Zhang, S.G. Shan, X.L. Chen, W. Gao. Are Gabor phases really useless for face recognition? Int. J. Pattern Anal. Appl. – 2009. – Том 12. – Вип. 3. – С. 301–307.
16. L. Yu, Z. He, Q. Cao. Gabor texture representation method for face recognition using the Gamma and generalized Gaussian models. Image Vis. Comput. – 2010. – Том 28. – Вип. 1. – С. 177–187.
17. Z.S. Zhao, L. Zhang, M. Zhao, Z.G. Hou, C.S. Zhang. Gabor face recognition by multi-channel classi_er fusion of supervised kernel manifold learning. Neurocomputing. – 2012. – Том 97. – С. 398–404.
18. B. Zhang, S. Shan, X. Chen, W. Gao. Histogram of Gabor phase patterns: a novel object representation approach for face recognition. IEEE Trans. Image Process. – 2007. – Том 16. – Вип. 1. – С. 57–68.
19. C.X. Ren, D.Q. Dai, X. Li, Z.R. Lai. Band-reweighed Gabor kernel embedding for face image representation and recognition. IEEE Trans. Image Process. – 2014. – Том 32. – Вип. 2. – С. 725–740.

20. J. Oh, S. Choi, C. Kimc, J. Cho, C. Choi. Selective generation of Gabor features for fast face recognition on mobile devices. Pattern Recognit. Lett. – 2013. – Том 34. – Вип. 13. – С. 1540–1547.
21. Z. Lei, M. Pietikäinen, Z.L. Stan. Learning discriminant face descriptor. IEEE Trans. Pattern Anal. Mach. Intell. – 2014. – Том 36. – Вип. 2. – С. 289–302.
22. K. Bowyer, K.P. Chang, P. Flynn. A survey of approaches and challenges in 3D and multi-modal 3D + 2D face recognition. Comput. Vis. Image Underst. – 2006. – Том 101. – Вип. 1. – С. 1–15.
23. V. Blanz, T. Vetter. Face recognition based on _tting a 3D morphable model. IEEE Trans. Pattern Anal. Mach. Intell. – 2003. – Том 25. – Вип. 9. – С. 1063–1074.
24. G. Passalis, P. Panagiotis, T. Theoharis, I.A. Kakadiaris. Using facial symmetry to handle pose variations in real-world 3D face recognition. IEEE Trans. Pattern Anal. Mach. Intell. – 2011. – Том 33. – Вип. 10. – С. 1938–1951.
25. U. Prabhu, H. Jingu, S. Marios. Unconstrained pose-invariant face recognition using 3D generic elastic models. IEEE Trans. Pattern Anal. Mach. Intell. – 2011. – Том 33. – Вип. 10. – С. 1952–1961.
26. H. Drira, B. Ben Amor, A. Srivastava, M. Daoudi, R. Slama. 3D face recognition under expressions, occlusions and pose variations. IEEE Trans. Pattern Anal. Mach. Intell. – 2013. – Том 35. – Вип. 9. – С. 2270–2283.
27. Y. Lei, M. Bennamoun, M. Hayat, Y. Guo. An ef_cient 3D face recognition approach using local geometrical signatures. Pattern Recognit. – 2014. – Том 47. – Вип. 2. – С. 509–524.
28. Y. Zhang, A. Martinez. A weighted probabilistic approach to face recognition from multiple images and video sequences. Image Vis. Comput. – 2006. – Том 24. – Вип. 6. – С. 626–638.
29. Z. Cui, H. Chang, S. Shan, B. Ma, X. Chen. Joint sparse representation for video-based face recognition. Neurocomputing. – 2014. – Том 135. – Вип. 5. – С. 306–312.

30. Breaking Down Facial Recognition: The Viola-Jones Algorithm. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999
31. M. Turk, A. Pentland. Eigenfaces for Recognition. Cognitive Neuroscience. – 1991. – Том 3. – С. 71-86.
32. MSBuild. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/ru-ru/visualstudio/msbuild/msbuild?view=vs-2022
33. An introduction to NuGet. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/nuget/what-is-nuget
34. Windows Presentation Foundation. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation