НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

“На правах рукопису”

УДК 004.05

До захисту допущено

Завідуючий кафедрою СКС

Віталій РОМАНКЕВИЧ

“___”_____2022р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системне програмування та спеціалізовані комп’ютерні системи»

зі спеціальності 123 Комп‘ютерна інженерія

на тему: «Метод розпізнавання обличчя на базі модифікованого алгоритму HAAR»

Виконав:

студент II курсу, групи КВ-11мп

Дадиверін Віталій Валерійович_____ _____

Науковий керівник:

доцент, к.т.н.

Потапова Катерина Романівна_____ _____

Консультант з нормоконтролю:
доцент, к.т.н., с.н.с.
Боярінова Юлія Євгенівна _____ _____

Рецензент:

_____ _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2022 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність 123. «Комп’ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

(підпис)

«___» _____20___ р.

ЗАВДАННЯ

на магістерську дисертацію студента

Дадиверіна Віталія Валерійовича

1. Тема дисертації Метод розпізнавання обличчя на базі модифікованого алгоритму HAAR

Науковий керівник дисертації Потапова Катерина Романівна, к.т.н. доцент кафедри СПСКС

затверджені наказом по університету від 25 листопада 2022 р. № 4317-С

2. Термін подання студентом проекту _____

3. Об’єкт дослідження: процес ідентифікації людини за рисами обличчя

4. Предмет дослідження: методи розпізнавання та ідентифікації обличчя

5. Перелік завдань: аналіз існуючих рішень, обґрунтування обраних технологій, створення модифікованого методу, описання модулів та алгоритмів створеної програми, тестування, висновки.

6. Перелік ілюстративного матеріалу: презентація.

7. Перелік публікацій: XV наукова конференція магістрантів «Прикладна математика та комп'ютинг» ПМК-2022 тези «ВИКОРИСТАННЯ ТА МОДИФІКУВАННЯ HAAR CASCADE FACE DETECTOR» (Київ, 16-18 листопада 2022 р., 314-318 с.), VI Міжнародна наукова конференція під назвою «MODERN RESEARCH IN WORLD SCIENCE» тези «ПРО ОРГАНІЗАЦІЮ ТА ВИКОРИСТАННЯ ДЕТЕКТОРА ОБЛИЧЧЯ» (Львів, 4-6 вересня 2022 р., 226-232 с.), Науковий журнал «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки» стаття «СИСТЕМА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ З ВИКОРИСТАННЯМ ПОЄДНАННЯ МЕТОДІВ HAAR ТА SVM» (Херсон, 2022 р.), Свідоцтво про реєстрацію авторського права №114734 у Державному підприємстві «Український інститут інтелектуальної власності» – комп'ютерна програма «GrubCut module» (Київ, 7 вересня 2022 р.).

8. Дата видачі завдання – 10 жовтня 2021

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Обрання предметної області та розбір теоретичної частини	09.09.2022	
2	Визначення структури програмного рішення та його наповнення	19.09.2022	
3	Формування програмних модулів	2.10.2022	
4	Формування візуальної частини програми	16.10.2022	
5	Об'єднання всіх частин програми в одну систему	28.11.2022	
6	Тестування	7.11.2022	
7	Формування звіту	18.11.2022	
8	Попередній перегляд дисертації на кафедрі	01.12.2022	

Студент _____ Віталій ДАДИВЕРІН

Керівник проєкту _____ Катерина ПОТАПОВА

РЕФЕРАТ

Актуальність теми.

Системи розпізнавання обличчя все більше захоплюють різні області нашого життя, допомагаючи його зробити більш комфортним та достатньо захищеним. В цьому році ситуація з віддаленою роботою змінилась і стала майже неможливою через постійні термінові відключення електроенергії та інших систем, що щільно пов’язані з нею. Саме тому збільшується кількість компаній приватного та державного фінансування, що цікавляться різними системами безпеки, які можна використати для захисту внутрішньо інформації та продуктів розробки. Такий запит ринку створює попит на різні системи ідентифікації і збільшує конкуренцію між виробниками такого роду продукції, тому існує потреба у модифікаціях та покращеннях подібних систем. Саме тому було обрано тематику покращення ефективності систем ідентифікації.

Об’єктом дослідження є процес ідентифікації людини за рисами обличчя.

Предметом дослідження є методика ідентифікації людини за біометричними рисами, використовуючи засоби комп’ютерного зору.

Мета роботи: покращення якості розпізнавання алгоритму Haar за двома кількісними критеріями (CBP та F-міра), розробка власної системи ідентифікації з використанням даного алгоритму, ґрунтовне тестування готового програмного продукту для демонстрації переваг у порівнянні з іншими алгоритмами.

Наукова новизна роботи полягає в наступному:

1. Запропоновано модернізований алгоритм Haar, який покладається на аналіз за трьома кольоровими моделями, використання алгоритму класифікації SVM зі зміненою стратегією вибірок для навчання та застосування методу генерації негативних фотозображень з наявних позитивних;
2. Використання алгоритму розпізнавання обличчя для підвищення якості ідентифікації.

Практична цінність отриманих в роботі результатів полягає в тому, що запропоновані методи можуть бути використані у різних сферах застосування з обмеженими обчислюваними можливостями, такими як: ідентифікація у офісних приміщеннях, ідентифікація у системах розумного будинку або «домофонах», програми з пошуку відповідних людей на фото- та відеозображеннях для великого розмаїття питань, зокрема і військового напрямку. Наведений модифікований алгоритм підвищує якість розпізнавання на значення з даного інтервалу 3,2-5,1 за критерієм F-міри. в залежності від кількості проаналізованих зображень.

Апробація роботи. Більшість положень щодо досліджень викладені на таких наукових конференціях:

1. XV наукова конференція магістрантів «Прикладна математика та комп'ютинг» ПМК-2022 тези «ВИКОРИСТАННЯ ТА МОДИФІКУВАННЯ HAAR CASCADE FACE DETECTOR» (Київ, 16-18 листопада 2022 р., 314-318 с.);
2. VI Міжнародна наукова конференція під назвою «MODERN RESEARCH IN WORLD SCIENCE» тези «ПРО ОРГАНІЗАЦІЮ ТА ВИКОРИСТАННЯ ДЕТЕКТОРА ОБЛИЧЧЯ» (Львів, 4-6 вересня 2022 р., 226-232 с.);
3. Науковий журнал «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки» стаття «СИСТЕМА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ З ВИКОРИСТАННЯМ ПОЄДНАННЯ МЕТОДІВ HAAR ТА SVM» (Херсон, 2022 р.);
4. Свідоцтво про реєстрацію авторського права №114734 у Державному підприємстві «Український інститут інтелектуальної власності» – комп'ютерна програма «GrubCut module» (Київ, 7 вересня 2022 р.).

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, трьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів розробки.

У першому розділі розглянуто існуючі алгоритми та системи ідентифікації обличчя, проаналізовано їх та визначено недоліки та переваги, що впливають на розвиток технологій, приналежних до цієї царини.

У другому розділі визначено інструменти розробки, обрані технології та алгоритми, представлено модифікацію наявного підходу до розпізнавання.

У третьому розділі представлено структуру розробленого програмного доробку та розписано наповнення кожного модуля.

У *четвертому розділі* продемонстровано тестування модифікованого алгоритму за визначеними критеріями оцінки та якості роботи всього програмного доробку.

У висновку описані результати проведеної роботи та покращення, що стосуються алгоритму розпізнавання.

Робота представлена на 97 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: Haar, SVM, HOG, RGB, середовище розробки, ідентифікатор, класифікатор.

ABSTRACT

Actuality of theme.

Facial recognition systems are increasingly taking over various areas of our lives, helping to make it more comfortable and sufficiently protected. This year, the situation with remote work changed and became almost impossible due to constant emergency power outages and other systems closely related to it. That is why an increasing number of private and public funding companies are interested in various security systems that can be used to protect internal information and development products. Such market demand creates a demand for different identification systems and increases competition between manufacturers of this kind of products, so there is a need for modifications and improvements of such systems. That is why the topic of improving the efficiency of identification systems was chosen.

The object of research is the process of identifying a person by facial features.

The subject of the study is the method of identifying a person by biometric features using computer vision.

The purpose of the work: improving the recognition quality of the Haar algorithm based on two multiple criteria (KVR and F-measure), developing the identification of one's own system using this algorithm, reasonable testing of the finished software product to demonstrate its advantages compared to other algorithms.

The scientific novelty of the work arises in the following:

1. A modernized Haar algorithm is proposed, which is invested in the analysis of three color models, using the SVM classification algorithm with a modified strategy for choosing classifications for training and applying the method of generating negative photo images from existing positive ones;

2. Using a face recognition algorithm to improve the quality of identification.

The practical value of the results obtained in the work is that the proposed methods can be used in various fields of application with limited computing capabilities,

such as: identification in office premises, identification in smart home systems or "intercoms", programs for finding relevant people in a photo . - and video images for a wide variety of issues, including the military direction. A modified algorithm for achieving recognition quality for values from the given interval of 3.2-5.1 according to the F-measure criterion is given. depending on the many images analyzed.

Approbation of work. Most of the provisions on research are laid out at the following scientific conferences:

1. XV scientific conference of master's students "Applied mathematics and computing" PMK-2022 theses "USE AND MODIFICATION OF HAAR CASCADE FACE DETECTOR" (Kyiv, November 16-18, 2022, pp. 314-318);

2. VI International Scientific Conference entitled "MODERN RESEARCH IN WORLD SCIENCE" theses "ON THE ORGANIZATION AND USE OF THE FACE DETECTOR" (Lviv, September 4-6, 2022, p. 226-232);

3. Scientific journal "Scientific notes of TNU named after V.I. Vernadskyi. Series: Technical Sciences" article "FACE RECOGNITION SYSTEM USING COMBINATION OF HAAR TA SVM METHODS" (Kherson, 2022);

4. Certificate of copyright registration No. 114734 at the State Enterprise "Ukrainian Institute of Intellectual Property" - computer program "GrubCut module" (Kyiv, September 7, 2022).

Structure and scope of work. The master's thesis consists of an introduction, three sections and conclusions.

In the introduction, the general characteristics of the work are given, an assessment of the current state of the problem is made, the relevance of the research direction is substantiated, the goal and tasks of the research are formulated, the scientific novelty of the obtained results and the practical value of the work are shown, and information about the approbation of the development results is given.

In the first chapter, the existing algorithms and face identification systems are considered, they are analyzed and the disadvantages and advantages affecting the development of technologies belonging to this field are determined.

The second section defines development tools, selected technologies and algorithms, and presents a modification of the existing approach to recognition.

The third section presents the structure of the developed software and describes the contents of each module.

The fourth chapter demonstrates the testing of the modified algorithm according to the defined criteria for the assessment and quality of the entire software development.

The conclusions present the results of the work carried out and the improvements achieved during the development of the algorithm.

The work is presented on 97 sheets, contains links to the list of used literary sources.

Keywords: Haar, SVM, HOG, RGB, development environment, identifier, classifier.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Haar – ознаки цифрового зображення.

HOG – гістограма направлених градієнтів.

IDE – Integrated Development Environment. Інтегроване середовище розробки.

OpenCV – Open Source Computer Vision Library. Бібліотека з наявними алгоритмами, що пов’язані з комп’ютерним зором.

QML – Qt Meta Language або Qt Modeling Language. Похідна мова програмування від JavaScript для написання UI.

STL – Standard Template Library. Бібліотека, в якій знаходять стандартні шаблони C++.

UI – User Interface. Інтерфейс взаємодії користувача з програмою.

ВСТУП

На сьогоднішній день програми, що спираються на алгоритми розпізнавання обличчя, відіграють ключову роль у наших життєвих процесах, бо різного роду біометричні системи задовольняють потреби у аутентифікації та ідентифікації людей для подальшого використання у конструкціях, які гарантують різного роду безпекові питання: охорона місцевості та режимних об’єктів різного рангу секретності, гарантування безпеки фінансових операцій, забезпечення секретності інформації інтимного характеру тощо. Існує вже багато гарних прикладів комерційних продуктів, що стосуються приватних даних, які повинні бути максимально захищеними та в той же час достатньо комфортний у використанні, і на допомогу у цій справі приходить технологія розпізнавання обличчя з інтеграцією у більшу систему алгоритмів обробки та аналізу. Місцевими прикладами таких успішних комерційних продуктів є фінансові операції за використанням додатку Приват 24 або його екосистеми, система ідентифікації у Дії, яка дає змогу верифікувати особистість максимально швидко та зручно. Також варто зазначити, що ця технологія стає невід’ємною частиною систем контролю правопорядку, яка може дуже допомогти слідчим органам притягнути обвинувачуваного до відповідальності, чи військових систем виявлення противника або локалізації району майбутнього ураження.

Чіткий алгоритм, висока швидкодія та точність є визначальними рисами методів, що обираються для використання у вищеназваних системах, бо в цьому питанні кількість допущених помилок у геометричні й прогресії погіршує можливі наслідки, саме тому в цій дисертації буде надано поліпшення існуючого алгоритму з вибудуваною архітектурою.

Для досягнення поставленої мети буде виконано наступні пункти:

- Розгляд наявних систем ідентифікації та формулювання модифікації алгоритму;

- Проектування застосунку з ідентифікації обличчя по типу СКУД;
- Програмна реалізація та тестування з використанням параметрів оцінки (коефіцієнт вдалого розпізнавання і F-міра).

1. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ІДЕНТИФІКАЦІЇ

1.1. Новітні системи та способи ідентифікації

Ідентифікація [1] – це процедура розпізнавання персоналії у деякій системі, використовуючи завчасно натреновану модель, що містить потрібну біометричну або іншу апріорну інформації про осіб, які є постійними користувачами даного продукту.

Ідентифікація [1] є інструментом здобуття інформації про суб’єкта системи з використанням завчасно наданого ним ідентифікатора. Це своєрідний перший крок у наданні доступу до системи, а далі відбувається автентифікація чи авторизація, дуже гарним прикладом останнього кроку є різного роду двохфакторні автентифікації.

Дуже важливим моментом є розуміння різниці між авторизацією та автентифікацією.

Автентифікація [1] є процесом встановленням приналежності інформації в системі особі користувача за наданими ключовими ідентифікаторами, які виступають своєрідним ключем для замку. Цей процес передує іншому під назвою авторизація.

Авторизація [1] є процесом, що керує засобами доступу до закритого ресурсу, фізичним шляхом (прикладання магнітного ключа до домофона) та електронно-цифровим («логінування» до системи управління камерами, що розташовані на території вашого ЖК), в залежності від пароля або ідентифікатора суб’єкта, для того щоб останній міг впливати на внутрішні зміни деякого комплексу об’єднаних методів збору та обробки даних.

Схема роботи типової системи збору та обробки даних, що має вбудовану підсистему захисту, представлена нижче на рисунку 1.1.

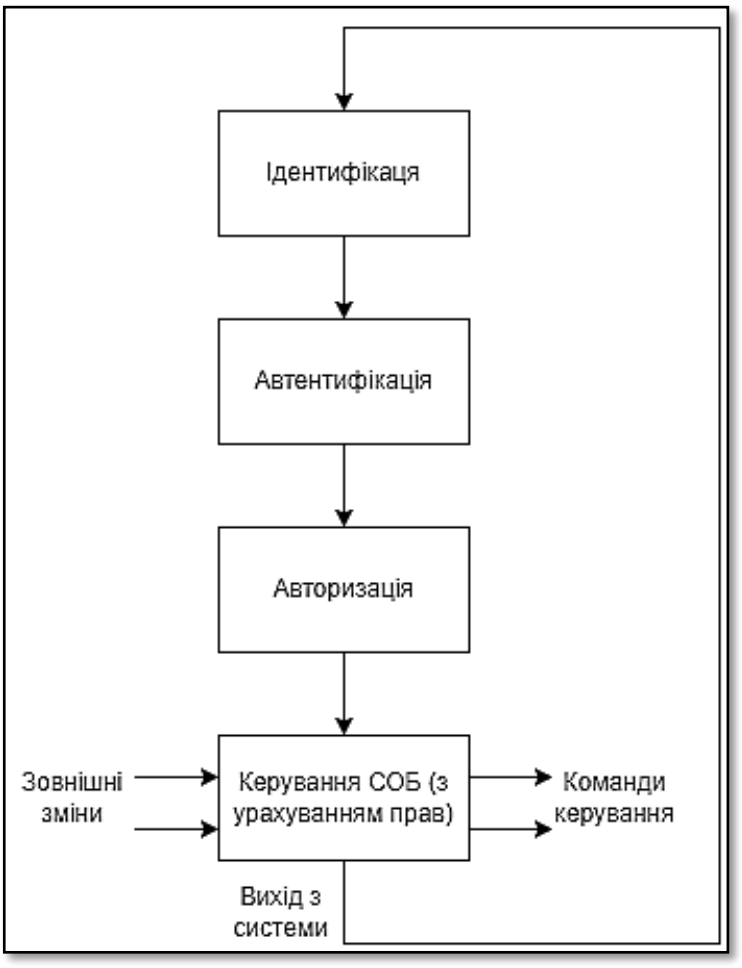

Рисунок 1.1 – Система роботи типової СОБ

Способи ідентифікації в типових системах обробки даних визначаються особливостями або точковими ознаками, що є вхідними даними. Найпопулярнішими та найбільш комфортними у використання є біометричні ознаки, бо вони чітко відповідають критерію індивідуальності та особливості, а зручність досягається швидкістю вводу даних за допомоги сенсорів або інших схожих пристроїв та нескладністю доставки цього набору сутностей. Також варто зазначити, що такий метод вводу даних зменшує можливість шахрайських зчитувань чи зломів на відміну від стандартних паролів.

Біометрія [2] – це набір автоматизованих методів і засобів ідентифікації осіб на основі їх фізіологічних або поведінкових характеристик, які широко використовуються у різних системах захисту (рисунок 1.2).

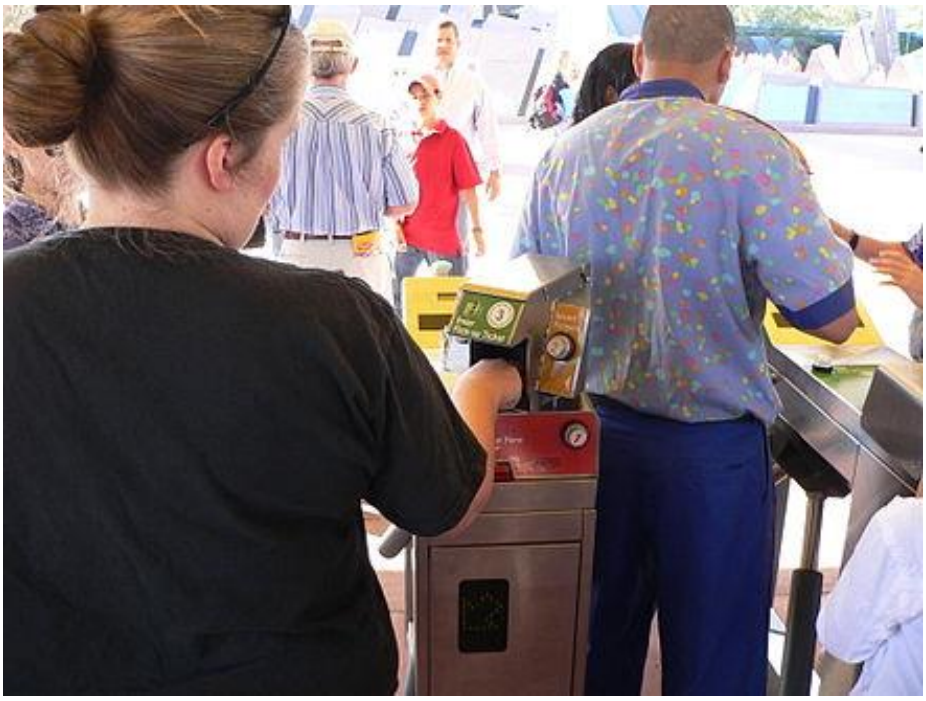

Рисунок 1.2 – Біометричне розпізнавання відбитків пальців перевіряє, щоб квиток використовується тільки однією людиною.

Зазвичай виділяють два різновиди методів біометрії:

- Статичні;
- Динамічні.

Статичні методи [2] мають таку назву через те, що при вводі такого роду даних особа не виконує ніяких рухів чи вхідними даними являються окремі обрані кадри з мінливої вибірки, а динамічні – з точністю навпаки (рисунок 1.3).

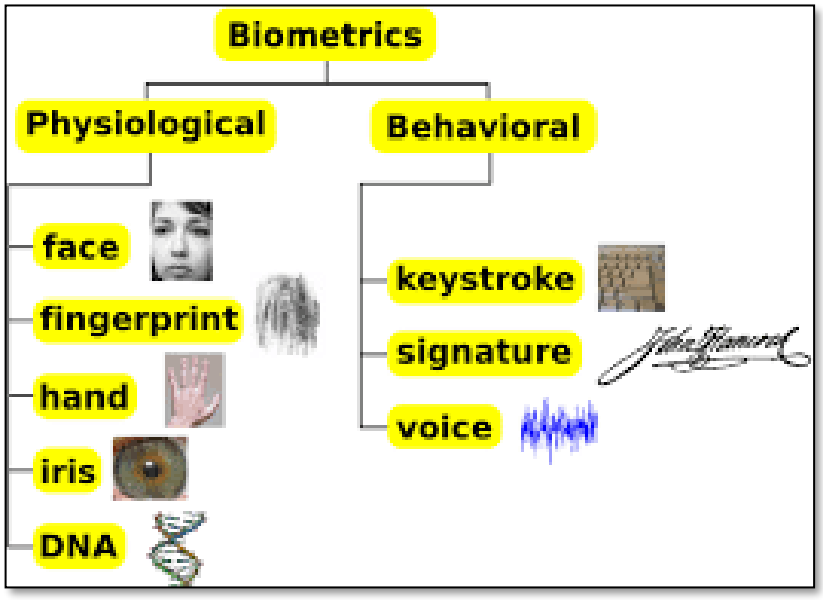

Рисунок 1.3 – Види біометричної ідентифікації

Статичні методи:

- Аналіз відбитку подушечки пальця. Найпоширеніший вид біометричної ідентифікації, цей метод заснований на унікальності малюнку подушечки пальця кожного суб'єкта зчитування. Завдяки спеціальному сканеру отримується зображення відбитку подушечки пальця, яке згодом переводиться у бінарний згортковий код і порівнюється з завчасно створеним набором шаблонів для перевірки на істинність;
- Аналіз форми долоні. Цей метод заснований на детальному вивченні форми руки та її геометричних особливостей. Завдяки спеціальному пристрою, що вміє створювати тривимірне зображення долоні та кисті, отримуються розмірності, що також як і в випадку з відбитками пальця переводяться у бінарний згортковий код правильної послідовності, який потім використовується для ідентифікації особистості;
- Аналіз розташування судин на тильній стороні долоні або кисті. Завдяки сенсорам інфрачервоної камери отримується бінарний

згортковий код за визначеною схемою розташування вен на обраній частині руки;

- Аналіз сітківки ока або точніше розпізнавання карти кровоносних судин очного дна. Для зчитування малюнку кровоносних судин використовується спеціальна камера та віддалена яскрава світлова пляма, на яку споглядає об’єкт ідентифікації. Далі малюнок переводиться у кодований бінарний варіант і порівнюється з набором шаблонів;
- Аналіз райдужної оболонки ока. Цей метод заснований на унікальності малюнку радужки ока. Як і в деяких попередніх методах тут потрібна спеціальна камера, що може зчитувати малюнок радужки, а далі відбувається переведення у кодований бінарний варіант для порівняння;
- Аналіз форми обличчя. Цей метод покладається на створені 2D та 3D зображення обличчя особи, що ідентифікується. Завдяки камері та ПЗ визначаються та позначаються контури брів, очей, носа та губ, а також визначається відстань між ними, це все переводиться у бінарний код та порівнюється з шаблонами;
- Аналіз термограми обличчя людини. Цей метод заснований на унікальному розподілі судин (артерій) на обличчі, що доставляють кров до обличчя та створюють так звані теплові зони, тому зйомка проводиться спеціальною інфрачервоною камерою. Далі проводиться програмний переклад у цифровий код та подальше порівняння з шаблонами;
- Інші схожі методи, подібні до ідентифікації за кутикулою, ДНК, запахом тіла тощо.

Динамічні методи:

- Аналіз поведінкових та динамічних особливостей об’єкта спостереження. Тобто враховуються унікальні підсвідомі рухи у процесі яких специфічних дій;
- Аналіз почерку під час письма. Цей метод заснований на унікальності підпису людини або якоїсь ключової фрази. Далі складається двійковий цифровий код за ключовими динамічними особливостями, такими як: часова характеристика написання, графічні характеристики, значення зміни поверхневого тиску на папері, тощо;
- Аналіз введення тексту з клавіатури. Динамічні особливості подібні до тих, що наведені в методі вище, але підпис на клавіатурі ввести неможливо, тому перевірка проводиться по ключовій фразі. Основною динамічною особливістю для аналізу являється динаміка набору кодової фрази;
- Аналіз голосового вводу. Створюється цифровий код звукового ідентифікатора, що включає в себе статистичні та частотні особливості голосу особи.
- Інші схожі методи, подібні до ідентифікації за динамікою повороту ключа у замку, за рухом губ тощо.

1.2. Комп’ютерний зір, фото- та відеоаналітика, різновиди технік розпізнавання обличчя

Комп'ютерний зір [3] (англійською Computer Vision або просто CV) – це використання комп’ютерних інструментів для автоматизації захоплення та обробки зображень, де є статичні об’єкти розпізнавання чи такі, що знаходять у русі. Основним підвидом комп’ютерного зору, що найбільше відповідає описаним

вище критеріям є так званий Машинний зір (англійською Machine Vision або просто MV), у наших широтах також використовують поняття «технічний зір» або «технологічний зір».

Перші зрушення у цій царині [3], аби змусити комп'ютер «бачити» як людина, відносять до першої половини 60-тих років минулого століття. Проте, максимально масштабні зрушення у цій сфері діяльності були досягнуті за останні 10-15 років, завдяки колосальному збільшенню обчислюваних потужностей, швидкодії процесорів, об'єму пам'яті різного типу та застосування, багатьох параметрів професійних камер, зокрема роздільної здатності, покращення пропускних можливостей каналів зв'язку, зменшення так званого «тротлення» під час перегріву, логічних розвиток методів, що спричинив появу методів машинного та поглибленого навчання (англійською Machine та Deep Learning), систем штучного інтелекту (англійською Artificial Intelligence або просто AI). В особливості, останнім часом технології комп'ютерного зору з великою інтенсивністю проникають у різні галузі промисловості та повсякденного життя людей, забираючи на себе левову частку тяжкої та нудної роботи.

Останні десять років використання комп'ютерного зору у промисловості стало трендом і набуло широкого поширення у таких галузях:

- Автомобілебудування;
- Харчова промисловість;
- Фармацевтика;
- Проектування медичних інструментів та інші дотичні до медицини технологічні процеси;
- Різнотипні виробництва мікроелектронних та мікропроцесорних виробів.

Наприклад, автомобільна промисловість використовує системи комп'ютерного зору для швидкого зчитування маркування збірних компонентів для покращення процесу конвеєрної збірки. Також технічний зір використовуються

для тестування, калібрування, перевірок розмірностей та зазорів, вирівнювання деталей на складальних лініях.

У складних технологічно виробництвах харчових продуктів комп’ютерний зір є критично важливим для перевірки відповідності використаних інгредієнтів до вказаних на упаковці, особливо це стає актуальним для виробів, що не мають містити в собі алергени або складаються виключно з екологічних продуктів.

Фармацевтика та інша високотехнологічна медична продукція передбачає підвищений рівень відповідальності за якість кінцевої продукції, тому точність налаштування та рівень композиції медичних компонентів надійно контролюється методами технічного зору.

Виробництво мікропроцесорів та інших електронних компонентів широко використовує переваги комп’ютерного зору, щоб створити умови «чистої збірки», тобто такої яка мінімізує можливість контакту людини до компонування елементів у єдине ціле, наприклад, контролюється розміщення кремнієвих пластин, маркування та позиціонування матриці інтегральних схем та інших допоміжних елементів.

На сьогоднішній день методи [3] комп’ютерного зору широко використовуються для багатьох компонентів цифрової економіки:

- Інтелектуальні транспортні системи ITC (англійською Intelligent Transportation System);
- «Розумне місто» (англійською Smart City) та його системи, останнім переможцем в цій номінації стало місто Київ;
- Безпілотні літаючі системи та аеротаксі, за типом аналогічні до дронів;
- Автономні автомобілі (англійською Driverless Car) або спеціалізовані системи допомоги водію (англійською Advanced driver-assistance systems або просто ADAS);

- Високотехнологічне сільське господарство (англійською Smart Agriculture);
- Система електронної медицини та записів до профільних лікарів (по типу eHealth);
- Системи ідентифікації в офісних приміщеннях;
- Системи подвійного та військового призначення;
- Адитивне виробництво, в основному великі 3D-машини для створення літальних компонентів;
- Багато інших, менших за своїм обсягом поширення через нижчу маржинальність.

Сьогоднішня стадія розвитку технологій комп'ютерного зору дуже далека від стелі свого потенціалу, але варто зазначити, що ця галузь є надзвичайно популярною, чому доказом слугують мільярдні у еквіваленті інвестиції.

Варто зазначити, що технічний зір дуже часто плутають з поняттям відеоаналітики, але вони не є тотожними поняттями та скоріше комп'ютерний зір є великою підмножиною, що включає в себе відео- чи фотоаналітику.

Відео- та фотоаналітика [3] (англійською Video Content Analysis та Image Analysis) – це якась програма комп'ютерного зору, що отримує базу знань та інформації з фото- та відеоконтенту. Якщо це розглядати з більш прикладної сторони, то дає відповідь на наступні запитання:

- Де: геолокація, 2D чи 3Д локація (їх також називають планарними та просторовими);
- Коли: пора року, сезонні позначки та особливості, дата та час з приміткою про часовий пояс;
- Що: предмети, особи, дії у момент зйомки, події, поведінка, відношення.

За типами програмного забезпечення фото- та відеоаналізу розрізняють такі:

- Ретроспектива: що сталося і як? Аналіз архівів фото та відеозаписів, що можуть містити ємнісні ознаки щодо причетності окремих особ, пошук на матеріалах окремі особливості, сортування останніх та отримання юридичних підстав для висування обвинувачення;
- Теперішній час: що відбувається на файлі в даний момент, підтримання контролю над ситуацією, формування одномоментних попереджень, кодування реакцій на ситуацію, компресування даних, отриманих з потоку:
- Аналітика на майбутнє: що може бути спричинене через вхідні дані чи що станеться з більшою ймовірністю, таке собі передбачення, що базується на даних отриманих в минулому чи прямо зараз, також можливе детектування аномальних закономірностей, що можуть стати впливовими чинниками.

Аналіз обличчя в теперішньому часі і є одною з головних складових, що розглядатиметься у цій роботі. Далі буде представлено більше інформації щодо розпізнавання обличчя як частини аналізу.

Безпосередньо під час процесу аналізу розпізнавання обличчя допомагає зрозуміти на яких частинах та об’єктах, що присутні на зображенні чи на відеозаписі, треба зосередитись для вірного визначення таких характеристик як: стать, вік та емоції на основі міміки. У системах розпізнавання обличчя математично відображаються риси обличчя людини (створення ідентифікатора) та зберігаються ці дані у вигляді відбитка обличчя, саме розпізнані дані є вхідними для системи алгоритмів, що визначають різні особливості без яких подальша авторизація не представляється можливою. Після розпізнавання новий відбиток вже можна порівнювати з бібліотекою інших обличь, для співставлення та допуску до системи.

Як працює розпізнавач обличчя?

Програми розпізнавання обличь [3] використовують алгоритми комп’ютерного зору, щоб проводити детектування людських обличь на зображеннях різної розмірності, які мають в собі достатню кількість інших об’єктів, що не є лицем, це можуть бути: пейзажі, будівлі, офісне нагромадження та інші частині тіла, як-от тулуб, ноги чи руки. В більшості випадків алгоритми починають свою роботу з пошуку людських очей, саме тої особливості, яку легше за все виявити (рисунок 1.4). Далі алгоритм [3] буде шукати інші менш виразні та більш складні для виявлення елементі, такі як: ніс, ніздрі, брови, рот, губи, райдужку ока та інші. Наступним кроком буде підтвердження гіпотези, що обличчя було знайдене, ця перевірка буде використовувати велику кількість додаткових тестів, що вказують саме на риси обличчя, а не будь якого іншого предмета чи його частини.

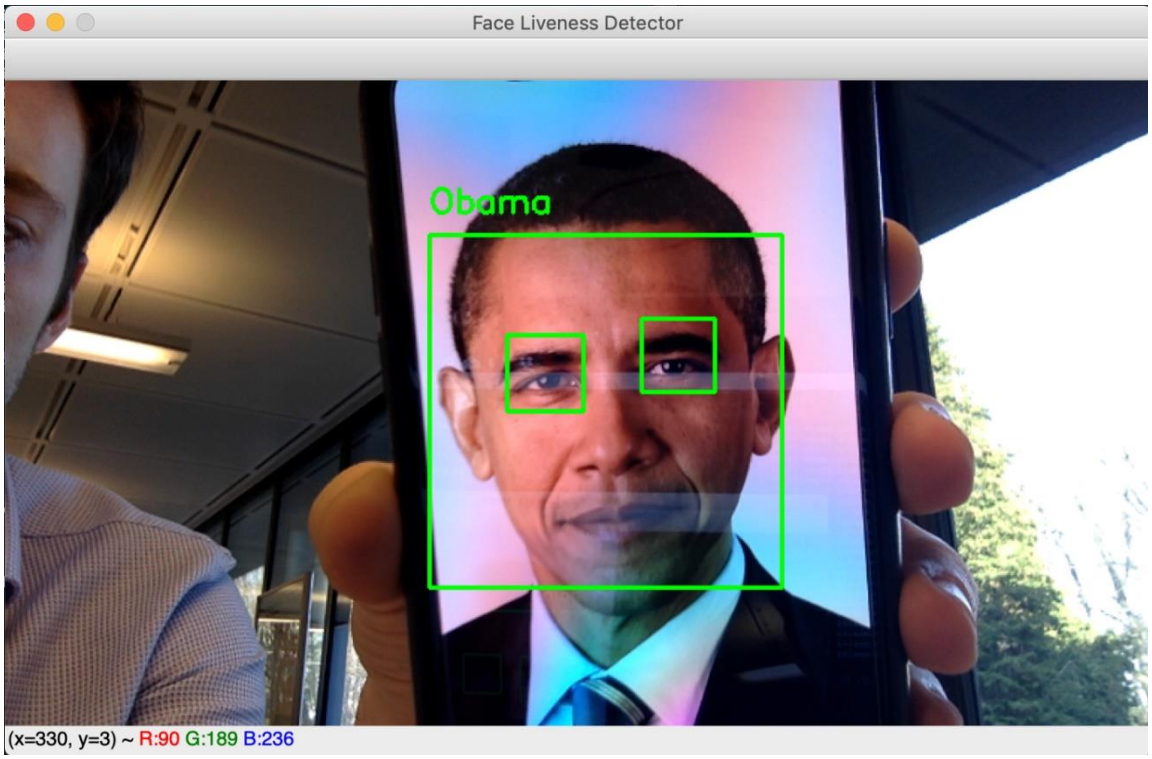

Рисунок 1.4 – Визначення очей як головної риси розпізнавання

Для підвищення точності алгоритму, проводиться навчання на збільшених наборах вхідних даних, що мають містити тисячі позитивних і негативних зображень у відповідному співвідношенні. Таке навчання у рази підвищує можливість позитивного визначення обличчя на фотозображенні та вказання місця його знаходження.

В основному методики[3] розпізнавання обличчя покладають на відповідні знання, що спираються на ключові ознаки, використовують в основі шаблони або за основу беремо критерії зовнішнього вигляду. Різність підходів виливається у своєрідні переваги та недоліки:

- Методи, що жорстко покладаються на знання та правила, програмний опис формується виключно на основі правил. Проблемою такого підходу є велика складність чіткого складення правил, які будуть задовольняти всі запити стосовно самого розпізнавання.
- Інваріантні методі, що беруть у основу визначення обличчя такі ознаки як очі, рот та ніс людини. Великою проблемою таких методів є різного роду шуми та засвіти на фотозображеннях, що є дуже частими гостями останніх.
- Методи шаблонного зіставлення, вони спираються на порівняння зображень зі стандартними зображеннями обличчя чи характеристиками, які були завантажені до моделі раніше і мають чітке співвідношення розмірностей. Проблемою для таких методів можуть стати зміни масштабу, переконфігурація форми зображення та варіації можливих поз;
- Методи зіставлення на основі зовнішнього вигляду, вони використовують статистичний аналіз і машинне навчання, щоб знайти релевантні характеристики зображень обличчя. Цей метод, який також використовується для виділення ознак для розпізнавання обличчя, поділяється на підметоди. Проблема є схожою до методів першої підгрупи, бо полягає в складності філігранного поєднання методів

машинного навчання між собою, щоб не було створення перевантаження системних компонентів через високий запит програмного забезпечення.

Часто достатньо специфічні методи розпізнавання обличчя включають в себе наступний функціонал:

- Видалення заднього фону, зазвичай таке стається за наявності одноколірного фону або простого визначеного статичного фону і така операція дає змогу більш чітко визначати межі обличчя, що розпізнається.
- На яскравих та кольорових фотозображеннях береться колір шкіри для більш простого визначення положення обличчя, але не для всіх відтінків цей функціонал є ефективним, а іноді це працює з точністю та і навпаки;
- Вдалою ідеєю є визначення рухомих об’єктів на відео для пошуку обличчя. На відеозаписах у режимі реального часу зазвичай рухаються окремо взяті частини живого суб’єкта і дуже часто саме такою частиною тіла є голова, тому при використанні такого підходу обчислюється площа руху, а далі до справи приєднається пошук за пропорціями. Але серйозним недоліком такого підходу є динамічний фон, бо якщо на задньому плані фіксується велика кількість рухової активності, то система може створювати велику кількість хибних або напівхибних детектувань. Напівхибними є ті, що зливаються в одне ціле з «позитивними» об’єктами.

Поєднання перерахованих вище стратегій може забезпечити комплексний метод виявлення обличчя.

Детектування обличь [3] на фотозображення іноді стає дуже складною задачею через змінюваність (мінливість) наступних факторів:

- Поза;

- Позиція та орієнтація у просторі;
- Вираз обличчя;
- Колір шкіри об’єкта та якість картинки;
- Кількість волосся в кадрі та на тілі;
- Наявність окулярів;
- Освітлення у кадрі та роздільна здатність фотозображення.

1.3. Розбір роботи детекторів систем контролю та управління доступу (СКУД)

Системи контролю та управління доступом або просто СКУД [4] (англійською – це сукупність технологічних та суто програмованих засобів, що виконує функцію регулювання доступу до певної зони або програмного забезпечення у кластерах та територіях, де присутня підвищена цінність вироблюваного продукту або секретність розробки, зазвичай персонал такого роду об’єктів підписує документ про нерозголошення з довготривалим строком дії (англійською Non-disclosure agreement або просто NDA).

Програмне забезпечення для розпізнавання обличчя [4], яке використовується в СКУД, завжди має перелік обов’язкових вимог, що потрібні для якісної та стабільної роботи (рисунок 1.5).

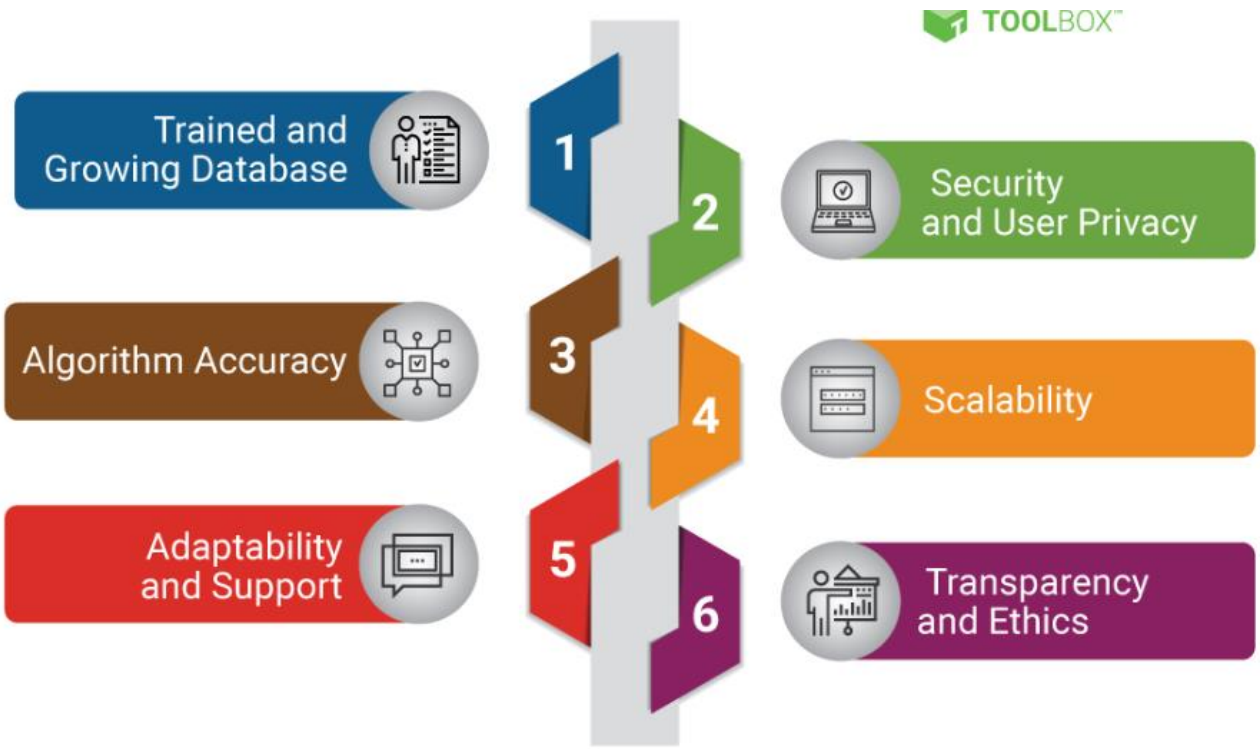

Рисунок 1.5 – Вимоги до системи розпізнавання обличь

Перелік основних вимог [4]:

- Навченість та зростання бази даних: рівень точності систем розпізнання обличчя (СРО) залежить від кількості вхідних даних, на основі яких буде створюватись модель. Кількість даних постійно повинна збільшуватись, урізноманітнюватись за різними параметрами, такими як стать, етнічне походження, напрямок світла та його походження, ракурси та вирази обличь. Хорошу тренувальну базу відрізняє різноманіття розподільної здатності обличчя, з яким система має справу. Ефективність машинного навчання завжди вимірюється у якості навчальних даних, тому СРО у цьому випадку не є винятком;
- Конфіденційність та безпека безпосередньо користувача: будь-яке програмне забезпечення, що спирається на біометрію, має тісний контакт з особистими даними користувача. Отже, це означає, що моделі (в нашому випадку скани обличчя), накопичені СРО, мають

бути конфіденційними. Конфіденційність має досягатись необхідною кількістю шифрувань та очищень кешу через заданий часовий проміжок. У випадку неочікуваного витоку даних має реалізовуватись чіткий план виправлень проблеми та блокування бази секретних даних;

- Точність визначення: якісними показниками, на які треба концентруватись під час розгляду СРО, є коефіцієнт помилкового прийняття (КПП) і коефіцієнт помилкового відхилення (КПВ). КПП – це ті зображення, що хибно визначені як позитивні. КПП – це ті зображення, що хибно визначені як негативні. Значення КПП повинно бути низький, а КПВ – високим. Саме тому при використанні таких систем часто присутня особа (контроль), що виконує додаткове адміністрування.
- Масштабування: для великих підприємств має бути можливість збільшення кількості пропускних пунктів до того часу, коли буде досягнуто задоволення запиту;
- Підтримка: постачальник послуг СКУД має передбачати випадки збою роботи системи, тому має бути створена резервна спрощена система або надано персонал, що може перейняти на себе функції адміністрування;
- Прозорість роботи і етика: лише за останній рік СРО кілька разів критикували через відсутність прозорості. Програмне забезпечення не має вдаватись до неетичних методів, як-от завантаження даних з приватних аккаунтів у соціальних мережах або порушення конфіденційності користувачів.

Далі будуть наведені успішні комерційні приклади СРО, що використовують у СКУД.

Amazon Rekognition [4]

Основні послуги: Amazon Rekognition — одне з найнадійніших імен у програмному забезпеченні розпізнавання обличь.

Аналіз обличчя і пошук по обличчю використовуються для верифікації користувачів, підрахунку людей і забезпечення громадської безпеки. Розпізнавання може ідентифікувати об’єкти та сцени, надаючи їм мітки. Amazon має у своєму розпорядженні величезний набір даних і в цьому його величезна перевага. Це забезпечує точність. Розпізнавання дозволяє користувачам додавати власні мітки до об’єктів і сцен відповідно до потреб бізнесу. Це забезпечує вищі показники успіху матчів. Все, що потрібно зробити компанії, це надати зображення об’єктів і місць, які потрібно ідентифікувати.

Amazon [4] забезпечує модерацію вмісту. Він має перевірений набір даних, який використовує для позначення неприйнятного вмісту. Він також забезпечує виявлення тексту для розпізнавання імен. Останньою пропозицією є виявлення ЗІЗ, яке ідентифікує працівників, які носять ЗІЗ для безпеки. Він також забезпечує пошук обличчя та перевірку в приватному сховищі фотографій. Зверніть увагу, що в цьому випадку стягуються витрати на зберігання. Розпізнавання також забезпечує виявлення обличь і аналіз у відео — живому або збереженому.

Клієнтська база: поточна клієнтська база Amazon Rekognition включає медіа-компанії, аналітичні компанії ринку, сайти електронної комерції та кредитні рішення.

Підтримка клієнтів: рішення надає документи, зразки та навчальні посібники.

Модель ціноутворення: модель ціноутворення базується на використаних послугах і кількості проаналізованих зображень. У рамках безкоштовного рівня AWS обробка зображень може використовуватися безкоштовно. Безкоштовний рівень триває 12 місяців і дозволяє аналізувати 5000 зображень на місяць і зберігати 1000 фрагментів метаданих обличчя на місяць. Після цього місячна вартість розраховується на основі кількості оброблених зображень (плюс плата за зберігання, якщо є). Подібні багаторівневі пакети доступні для обробки відео та настроюваних міток.

Додаткові коментарі: Amazon Rekognition можуть використовувати організації, які не мають досвіду машинного навчання. Це дороге рішення, тому для підприємств із вільним бюджетом. Єдине, на що користувачі попереджають звернути увагу, – це додавання вартості зберігання.

Betaface [4]

Основні послуги: Betaface в основному зосереджується на аналізі зображень і відео, а також на розпізнаванні обличчя і заперечень останнього.

Він пропонує три види послуг — SDK для розпізнавання обличчя, послуги з розробки програмного забезпечення для клієнтів і розміщені веб-сервіси. Послугами можуть бути просте або складне розпізнавання обличчя, ідентифікація та перевірка. Він використовує біометричні вимірювання для відстеження рис обличчя як на зображеннях, так і на відео. Він може розпізнавати емоції та етнічну приналежність. Він також може відстежувати шкіру, волосся, риси обличчя та форму зачіски. Він надає програмні рішення для відеоспостереження та безпеки.

Клієнтська база: Клієнтська база Betaface включає агентства веб-реклами та розваг, виробників медіа-контенту та розробників програмного забезпечення B2B.

Підтримка клієнтів: Betaface надає допомогу в інтеграції. Це також допомагає з повною підтримкою веб-сервісу та інфраструктури бази даних, якщо це вибрано.

Модель ціноутворення: Betaface дозволяє безкоштовно використовувати його API для 500 зображень на день і 0,035 євро за зображення після цього. Він також пропонує три плани передплати на місяць: базовий план за 245 доларів США за обробку 40 000 зображень, преміум-план за 490 доларів США за 100 000 зображень і ультраплан за 1595 доларів США за 300 000 зображень. За обробку кожного додаткового зображення стягується додаткова плата відповідно до плану передплати.

Додаткові коментарі: Betaface можуть використовувати організації, які розуміються на техніці та, можливо, мають власних розробників, які хочуть

інтегрувати свою програму з FRS. Це рішення може добре задовольнити невеликі підприємства.

BioID

Основні послуги: BioID [4] — це рішення, сумісне з GDPR, яке надає біометричні дані як послугу. Він надає хмарні служби FRS, до яких ваш продукт може отримати доступ за допомогою API. Програмне забезпечення пропонує три продукти:

Веб-сервіс BioID: це пропозиція SaaS, яку можна розгорнути локально або в хмарі.

Виявлення живих істот: це служба розпізнавання для визначення присутності користувача за допомогою розпізнавання обличчя, очей і голосу. Він використовується для запобігання онлайн-шахрайству та загрозам ідентифікації.

PhotoVerify: це рішення поєднує технологію виявлення обличь із службою Liveness Detection від BioID для перевірки фотографій, які використовуються як підтвердження особи.

Клієнтська база: це рішення призначене для компаній, яким потрібна електронна ідентифікація та електронний підпис. Він надає послуги фінансовим службам, KYC з біометрією, оренді автомобілів із самообслуговуванням, охороні здоров’я та організаторам онлайн-іспитів.

Підтримка клієнтів: BioID має надійну онлайн-документацію своїх API.

Модель ціноутворення: послуги BioID можна придбати на рівні оплати за користувача. Ви можете зв’язатися з BioID, щоб дізнатися ціни.

Додаткові коментарі: організації з власними програмами або продуктами можуть розглянути BioID. Він ідеально підходить для підприємств, де перевірка особи є важливою, наприклад для оренди автомобілів. Компанії, які представлені на різних континентах, можуть розглянути це рішення, оскільки воно відповідає GDPR.

Cognitec [4]

Основні послуги: Cognitec надає клієнтам FRS, що масштабується та налаштовується, завдяки своїй архітектурі відкритої системи через «FaceVacs». Cognitec пропонує п’ять рішень:

FaceVACS-VideoScan ES Live: це можна використовувати для розпізнавання обличь у прямих відеопотоках. BioID розширює це рішення, надаючи можливість підраховувати людей, генерувати демографічну статистику та відстежувати потоки людей.

FaceVACS-VideoScan ES: це корпоративне рішення Cognitec на основі передплати для встановлення та керування скануванням відео. Cognitec вибирає комп’ютерне обладнання та обладнання для камери та керує ним локально або в хмарі, розгортаючи партнера Cognitec.

FaceVACS-DBScan ID: це рішення для біометричної перевірки та ідентифікації.

FaceVACS-DBScan LE: це рішення для біометричної перевірки для правоохоронних органів.

FaceVACS-Entry: це рішення інтегрує програмне забезпечення FRS з апаратним забезпеченням найвищого класу для створення електронних воріт на контрольно-пропускних пунктах кордону.

Клієнтська база: Cognitec наразі має клієнтів у сфері управління документами, правоохоронними органами, фізичною безпекою та прикордонним контролем. Програмне забезпечення також працює з комерційними продуктами.

Підтримка клієнтів: клієнти Cognitec можуть порушувати проблеми на порталі клієнтів Cognitec. Він також надає підтримку на місці, навчання та консультаційні послуги. Також доступна команда віддаленої підтримки.

Модель ціноутворення: ціни починаються від 0,01 дол. США за раз за користувача та відрізняються залежно від використання.

Додаткові коментарі: Cognitec найкраще підходить для організацій, які шукають рішення безпеки та контролю натовпу на основі FRS. Він має значну присутність на ринку розпізнавання облич і може надавати ефективні масштабні рішення з мінімальними витратами на налаштування.

DeepVision AI [4]

Основні послуги: DeepVision AI надає рішення FRS для маркетингу та планування, а також для компаній, які хочуть використовувати перевірку обличчя для безпеки.

Він збирає дані про кількість відвідувачів у певному районі міста за віком, статтю та етнічною приналежністю. Ці дані використовуються, щоб допомогти рекламодавцям і брендам орієнтуватися на клієнтів за допомогою персоналізованих оголошень. Розпізнавання обличчя та перевірка для безпеки.

Рішення DeepVision для перевірки обличчя ідентифікує людей користувачами, як правоохоронні органи для безпеки. Він надає інформаційну панель аналітики в реальному часі, яку можна налаштувати.

Клієнтська база: DeepVision наразі орієнтується на розумних міських планувальників. Його клієнтами в основному є роздрібні фірми та рекламні агентства.

Підтримка клієнтів: DeepVision AI надає службу запитань і відповідей електронною поштою для запитів щодо інтеграції веб-служб. Він також надає практичну підтримку експертів з комп’ютерного зору. Він також пропонує цілодобову підтримку протягом усього року.

Модель ціноутворення: це програмне забезпечення оцінюється відповідно до розміру споживання. Ваш рахунок буде визначено кількістю ваших запитів. Програмне забезпечення стягує 0,008 доларів США за 10 запитів на основі розпізнавання обличчя, віку та статі, розпізнавання автомобіля, візуального контексту, візуального пошуку та розпізнавання бренду.

Додаткові коментарі: DeepVision найкраще підходить для забудовників нерухомості, фірм роздрібної торгівлі та маркетингових агентств. Функція на основі штучного інтелекту тут може бути застосована до безпеки та мобільності пішоходів, виявлення інцидентів і розпізнавання транспортних засобів, серед іншого, для забезпечення автоматизованого аналізу відео. Він ідеально підходить для середніх і великих підприємств, яким потрібні масштабовані рішення.

Face++ [4]

Основні послуги: Face++ надає чотири типи технологічних рішень:

- Розпізнавання обличь для визначення обличь, порівняння обличь і пошуку обличь;
- Розпізнавання тіла людини для виявлення тіла, виявлення скелета та контуру тіла;
- Прикрашання зображення для об’єднання обличь на кількох фотографіях;
- Розпізнавання зображень – для позначення обличь на фотографіях.

Face++ надає SDK та API для використання кожної з цих технологій, а також спеціальні хмарні служби. Пропозиції API включають розпізнавання обличчя, порівняння, пошук, інформацію про орієнтири обличчя, розпізнавання емоцій, оцінку погляду, аналіз стану шкіри та 3D-реконструкцію обличчя. Його пропозиції SDK включають порівняння обличь і інформацію про орієнтири обличчя.

Клієнтська база: сьогодні Face++ використовується в автомобільній промисловості, освіті, онлайн-маркетингу та індустрії мобільних телефонів.

Підтримка клієнтів: Face++ має вичерпну онлайн-документацію для API та SDK. Він також має онлайн-консоль підтримки для своїх користувачів.

Модель ціноутворення: пропонує чотири моделі ціноутворення: безкоштовна, платна за ходом, щоденний/місячний план і оплата за ліцензування. Безкоштовна модель дозволяє користувачам тестувати свої API та SDK протягом

необмеженого часу, обмежуючи кількість QPS. Щоденні та місячні ставки варіюються від $1000 до $10 500 на місяць залежно від вибраних функцій.

Додаткові коментарі: Face++ чудово підходить для порівняння обличь зі збереженими зображеннями. Деталізація функцій, які він пропонує, і його гнучкі моделі ціноутворення роблять його природним вибором для компаній, які все ще намагаються з’ясувати, як вони хочуть інтегруватися з FRS.

Висновки до першого розділу

Розглянуто способи ідентифікації, які є найбільш використовуваними, проаналізовано їх вплив на розвиток технологій, представлено чому саме такі біометричні методи є настільки широко розповсюдженими у сучасних програмованих системах та звичайних гаджетах, визначено їх недоліки та переваги.

Досліджено поняття комп’ютерного зору та підсистем, що належать до останнього. Визначено початок зрушення у цій царині, найважливіші місця застосування та можливі точки зростання.

Розглянуто різновиди підходів систем розпізнавання обличь, основних переваг та недоліків методів детекції, останнім проаналізованим пунктом стали додаткові специфічні методики, що можуть покращувати якість розпізнавання та зменшувати кількість затрат у абсолютних значеннях за рахунок зменшення навантаження на системні пристрої.

Визначено позитивні та негативні сторони систем контролю та управління за доступом з комерційних та некомерційних точок, що допомогли прийти до розуміння тенденції розвитку таких систем.

2. ОБҐРУНТУВАННЯ ОБРАНИХ ТЕХНОЛОГІЙ РОЗРОБКИ ТА МОДИФІКАЦІЯ АЛГОРИТМУ HAAR

2.1. Вибір технологій розробки

2.1.1. Мова програмування

C++ [5] є однією з суворо типізованих мов програмування, що може бути скомпільована для різнобарвного використання. Парадигми програмування, щ підтримуються цією мовою:

- Функціональна;
- Процедурна;
- Загальна;
- ООП (об’єктно-орієнтоване програмування).

Головною ідеєю [5] створення цієї було розширення можливостей мінімалістичної мови програмування C, що у великій кількості використовувала вставки коду мови низького рівня, по типу Assembler. Початково C++ називався C з класами, бо планувався лише як доповнення, але переріс у повноцінну мову програмування, що має деякі спільні моменти роботи мови C. Голові особливості, що відрізняють C++ від його предка:

- Зміна підходу до типів даних;
- Робота з винятками;
- Namespace;
- Inline-функції;
- Різні види перевантажень (наприклад, функціональне);
- Reference і memory control operators;
- STL-бібліотека, що була створена для стандартизації використання різних видів конейнерів;
- ООП (об’єктно-орієнтоване програмування).

Ця мова [5] використовується різними за складністю проектами, зокрема одним з напрямків являється Automotive, який використовує максимально звужені у ресурсному плані види «заліза» і тому проблема мінімального використання пам’яті та інших системних параметрів є актуальною як ніколи. Те саме можна сказати про вбудовані системи, комп’ютерні ігри і так далі.

Отже, можна підсумувати наступне, мова забезпечена [5] такими прерогативами:

- Швидкість, бо є виважена типізація і можливість використання машинного коду;
- Універсальність та кросплатформенність, все це організовано завдяки різним framework-інструментам та компіляторам;
- Популярність та підтримка, ці два компоненти є важливими для подальшого розвитку мов і вони є справедливими для цього випадку.

2.1.2. Qt фреймворк

Qt [5] є не лише лінійним графічним інструментом, але і дуже популярним C++ фреймворком, що використовується для проектування мережевих систем, інструментів контролю та модифікації даних у базі даних (наприклад, за допомоги SQL-запитів) і інших подібних додатків.

Гарною перевагою цього фреймворку [5] є кросплатформна сумісність, що дає підстави до написання програми, яка може використовуватись на різних типах девайсів. Все це дає можливість використовувати тільки вбудовані інструменти розробки і виключити прямий контакт з операційною системою (ОС) при розробці великих програмних додатків або продуктів. Саме тому об’єм специфічних частин коду, який призначений для пристосування до операційної системи зменшується

до неймовірно малих засобів і завдяки цьому не потребує колосального фінансування для контролю різниці між версіями продукту.

Перелік визначних особливостей цього фреймворку [5]:

1. Сигнали та слоти [5];

Одна з головних особливостей Qt, яка якісно відрізняє цю бібліотеку від її аналогів. За допомогою цього механізму будуються зв'язки між об'єктами. Коли відбувається запрограмована подія, сигнал надсилається до слота як своє «рідне» повідомлення. Загалом, слот повинен «реагувати» набудь-який підключений до нього сигнал. Точніше, це функції або методи об'єктів, головним завданням яких є реагування на дії користувача, що впливають на роботу програми.

2. Designer [5].

Це багаторівневий інструмент [5] для створення UI-вікон з великою кількістю впорядкованих між собою графічних компонент. Важливою якістю цього інструмента є його конструкція, що дозволяє істотно зменшити кількість операцій потрібних для створення потрібного елементу за допомогою збереження останнього у відповідних шаблонах.

Найкращим рішенням для роботи з цим фреймворком є QT Creator [5], що є «рідним» середовищем розробки, тому дуже важливо використовувати максимально між собою сумісні інструменти для покращення роботи системи в цілому та передбачення виникнення можливих помилок через велику кількість конструктивних відмінностей.

Qt Creator [5] є багатоплатформним середовищем розробки, що дозволяє працювати з наступними мовами програмування: C, C++, QML, JavaScript, Python та за великого завзяття і іншими. Головними опціями для програмування є можливість зручно проводити «дебагінг», завдяки наявним графічним рішенням, та інструменти створення UI-інтерфейсу, що можуть бути реалізовані як у вигляді компонентів класу Qt Widgets, так і у вигляді, компонент написаних мовою QML, що має пряму сумісність з JavaScript.

Зазвичай достатнім набором інструментів навіть для комерційної розробки наділена базова версія середовища розробки під назвою Community [5]. Вигляд цієї IDE можна побачити на рисунку 2.1.

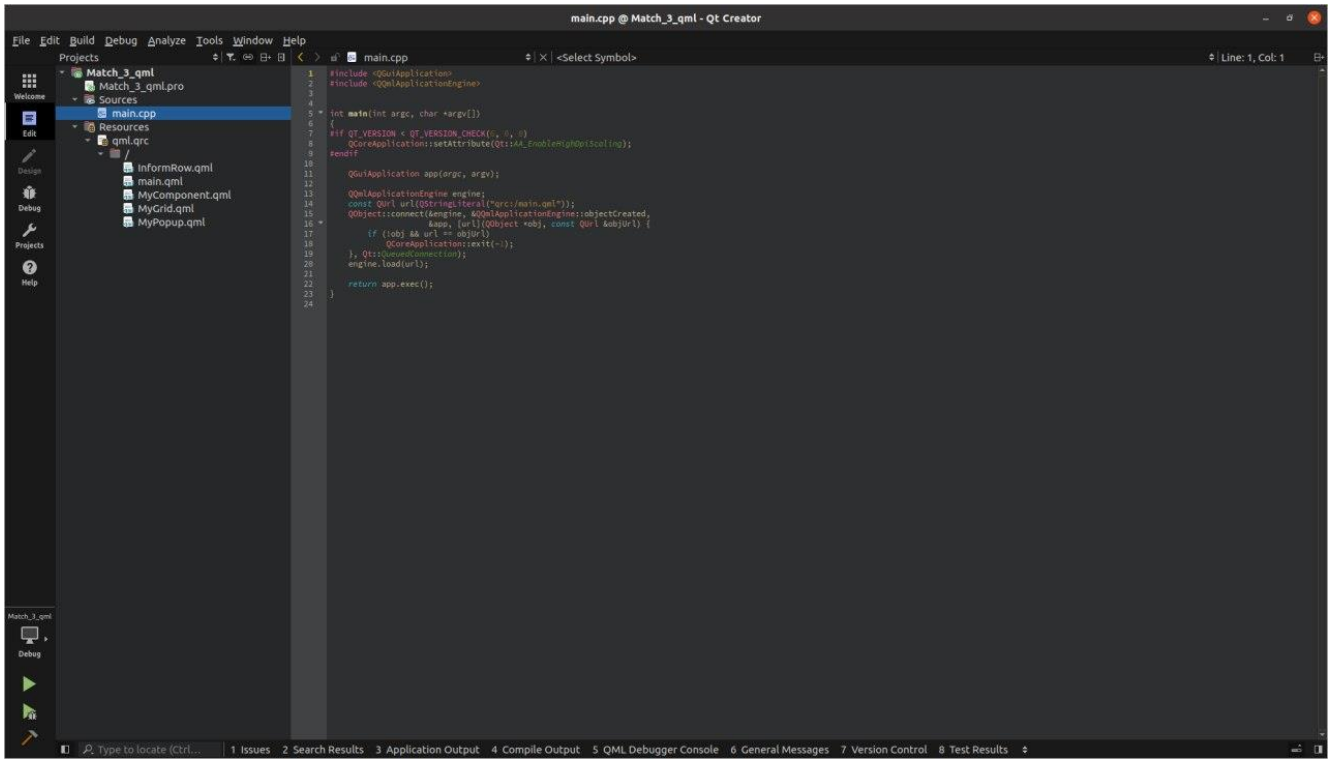

Рисунок 2.1 – Вигляд Qt Creator

Перелічені раніше причини лягли в основу вибору середовища розробки для створення UI-інтерфейсу, але з подальшим переїздом до іншого IDE з назвою Visual Studio для злагодження модулів між собою.

2.1.3. Середовище розробки Visual Studio 2022

Visual Studio 2022 [5] є інтегрованим середовищем розробки або просто IDE від відомої світової компанії Microsoft та використовується для створення програм з різним призначенням, таких як:

- Web-сервіс;
- Data-сервіс;
- Android та IOS додатки;
- Та інші, шо не схожі за структурою на попередні.

Є багато плагінів [5], які роблять це середовище більш досконалим за особливих обставин, тобто поліпшення існують на кожному рівні особливим чином, починаючи від систем контролю джерел (наприклад, GitLab) і закінчуючи додаванням різних інструментів, включаючи різні редактори для мов та інших програм, пов’язаних із візуальними аспектами розробки (наприклад, клієнт Azure DevOps: Team Explorer, який дає вам контроль над 3-рівневою архітектурою).

Підтримка різних мов програмування є важливою рисою якісного середовища розробки, це потрібно для того, щоб була присутня можливість написання модулів програми різними мовами з подальшою можливістю їхнього злагодження між собою. Кількість мов налічує 36 і серед них є наступні [5]: C++/CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML, CSS, Python, Ruby, Node.js. Також є можливість завантаження більш старих версій IDE, де можна знайти підтримку Java та J#.

Зазвичай достатнім набором інструментів навіть для комерційної розробки наділена базова версія середовища розробки під назвою Community [5]. Вигляд її можна побачити на рисунку 2.2.

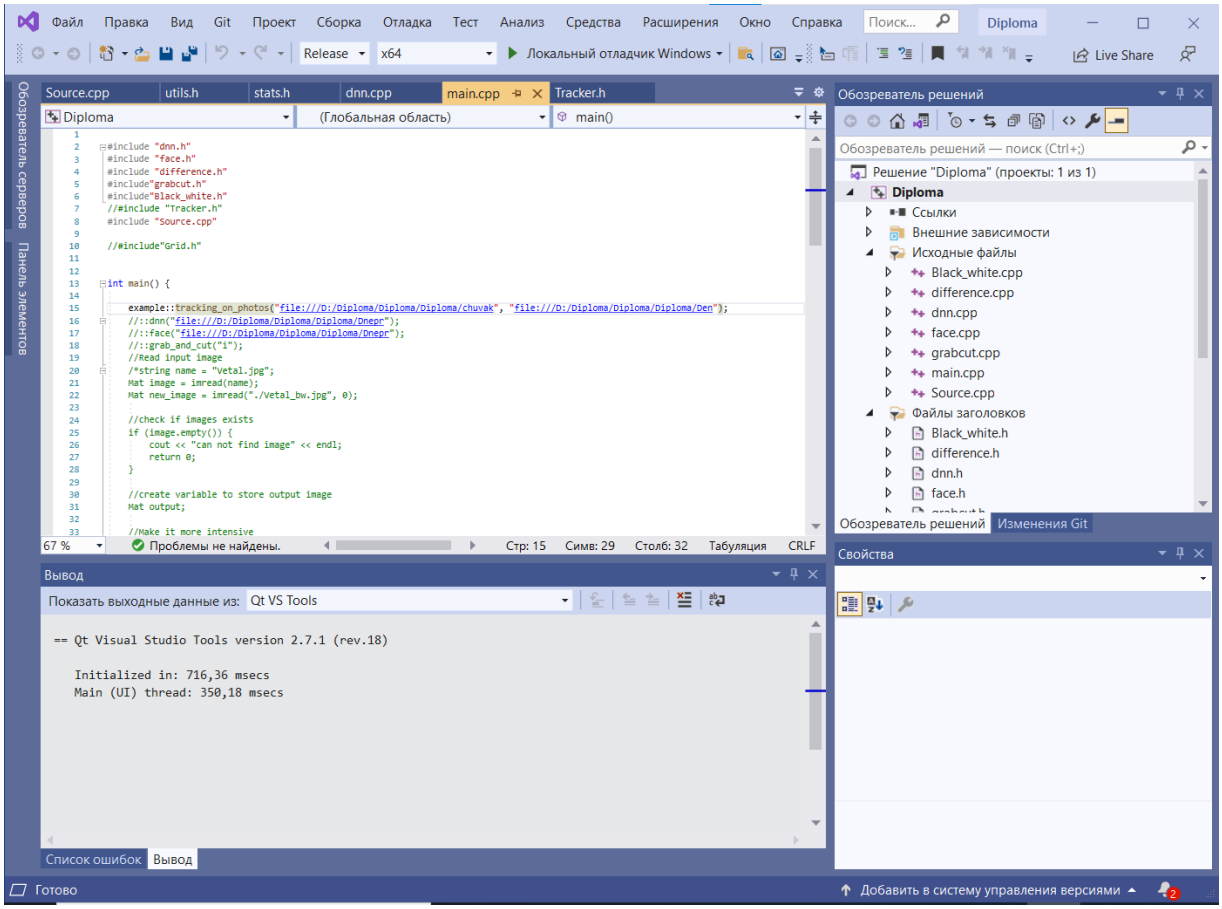

Рисунок 2.2 – Вигляд Visual Studio 2022

На даний момент, грудень 2022 року, стабільною версією [5] для розробки являється Visual Studio 2022, але є ще можливість використовувати більш старі версії, починаючи з 2015, бо вони ще входять до загальної девелеперської підтримки.

По раніше поясненим причинам будо обрано фреймворк QT, тому для використання візуального модуля і поєднання його з іншими було завантажено Qt VS Tools, що дає змогу повноцінно користатися даним фреймворком та його інструментами.

Інсталяція утиліти відбувається наступним чином:

1. Відкриваємо вкладку розширення (рисунок 2.3);

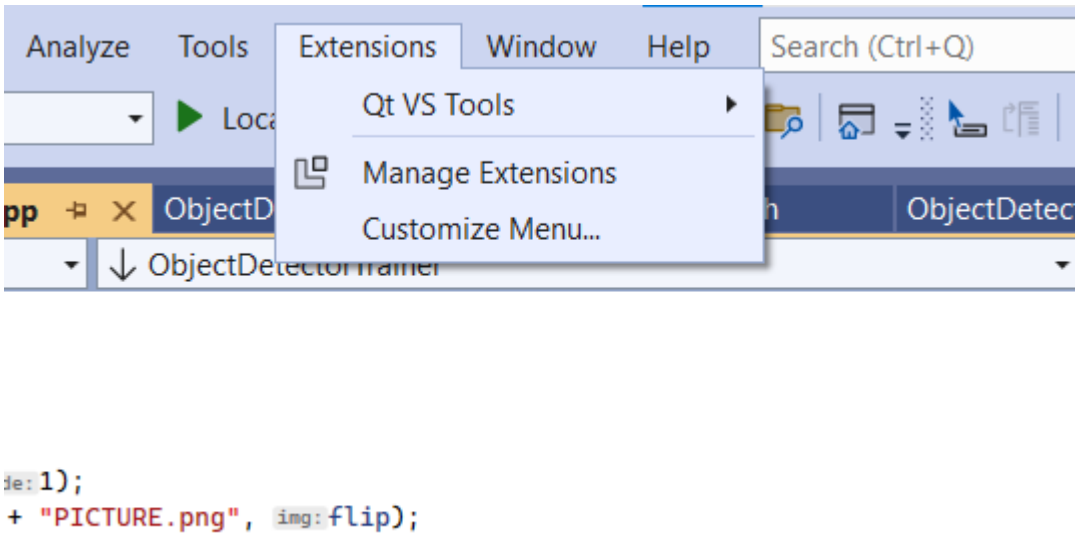

Рисунок 2.3 – Вкладка з утилітами

2. Відкриваємо вкладку управління розширеннями (рисунок 2.4);

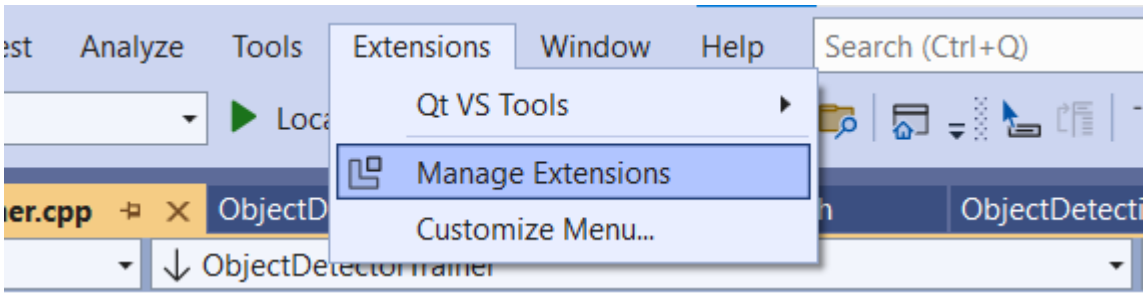

Рисунок 2.4 – Вкладка управління утилітами

3. Знаходимо потрібне утиліту і завантажуємо (рисунок 2.5).

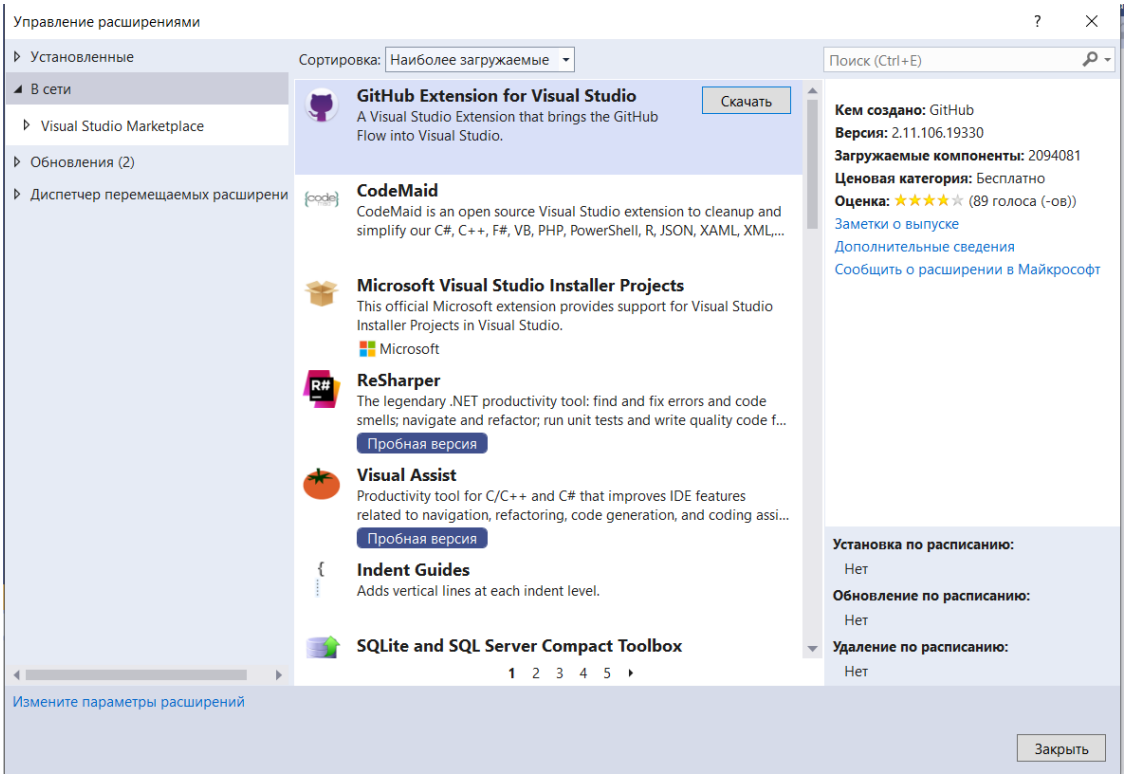

Рисунок 2.5 – Вкладка управління розширеннями

2.1.4. Менеджер пакетів vcpkg

Пакетний менеджер [5] є програмним забезпеченням, що контролює та управляє наступними процесами взаємодії з завантажуваною інформацією (програми, програмні модулі, програмне забезпечення для взаємодії з ОС, бібліотеки, фреймворки та інше): завантаження, оновлення, видалення і налаштування.

Vcpkg було обрано через наступні причини [5]:

1. Постійна підтримка з боку компанії Microsoft;
2. Зменшення об’ємів створюваних проектів за рахунок того, що файли лежать в іншій директорії і можуть бути використані в альтеративних рішеннях;

3. Легкість використання і керування командною рядка PowerShell (рисунок 2.6);

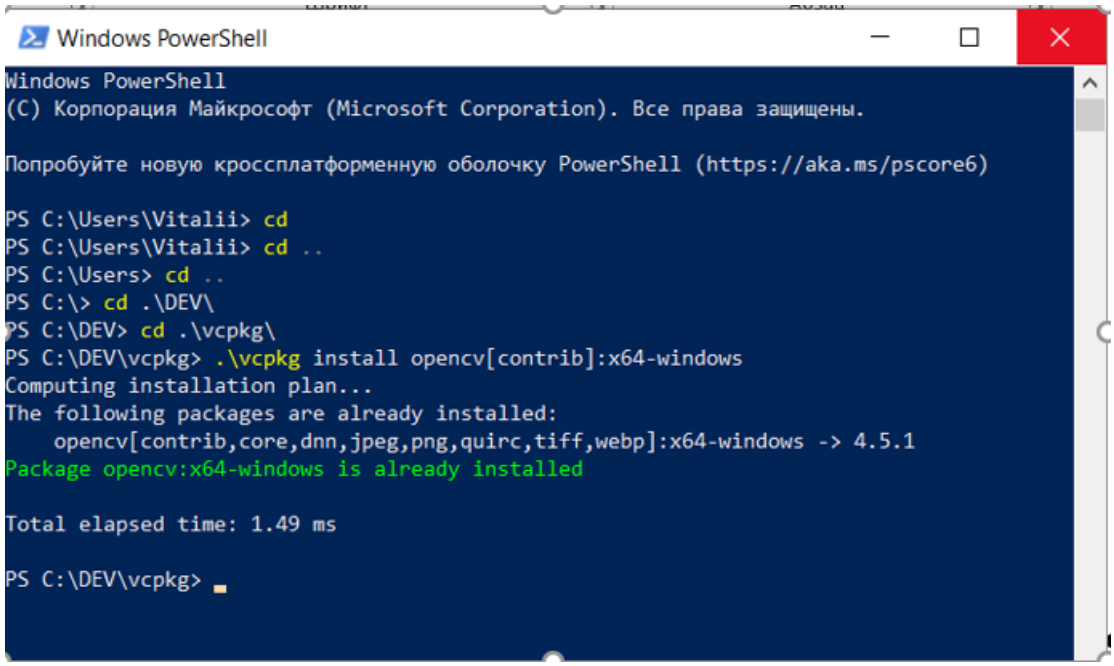

Рисунок 2.6 – Взаємодія з PowerShell

4. Прекрасна взаємодія з Visual Studio 2022;
5. Розташування завантажених ресурсів.

Остаточна причина [5] мого вибору полягає в тому, що всі необхідні файли, бібліотеки, фреймворки тощо, пов’язані з програмним забезпеченням, знаходяться в окремому каталозі, повністю відокремленому від поточного проекту, що розробляється, в будь-якому іншому місці. Варто також зазначити, що завдяки величезному списку файлів, бібліотек та фреймворків, які можна завантажити, цей менеджер задовольняє більшість запитів щодо необхідних інструментів розробки.

Цей менеджер пакетів [5] був потрібен для завантаження файлів бібліотеки OpenCV та dlib.

2.2. Вибір алгоритму розпізнавання обличчя

Haar Cascade Face [5] – це алгоритм для виявлення об’єктів, в нашому випадку обличчя людей, на основі машинного навчання (на англ. – machine learning), що був розроблений Майклом Джонсоном та Полом Віолою і представлений у роботі «Швидкісне виявлення об’єктів, використовуючи посилений каскад простих функцій» (на англ. – «Rapid object detection using boosted cascade of simple features» [9]) в 2001 році.

Машинне навчання [5] полягає в тому, що каскадна функція проходить тренування на великій кількості позитивних та негативних зображень, де позитивні – це ті зображення, на яких присутній об’єкт для пошуку, а негативні – це ті, де цього об’єкту немає.

Це навчання відбувається за наступним алгоритмом [5]:

- Даємо тестові зображення [5] $(x_1, y_1), \dots, (x_n, y_n)$, where $y_i = 0, 1$ для негативного або позитивного результату відповідно;
- Ініціалізуємо значення ваги [5] (1)

  $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ для $y_i = 0, 1$ відповідно, (1)

  де m та l – це кількість негативних та позитивних відповідно;
- Далі ми заходимо у цикл [5] $t = 1, \dots, T$:
  1. Нормалізуємо вагу, $w_{1,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{1,j}}$, (2)
  2. так що w_t – це розподіл ймовірностей;
  3. Для кожної ознаки j, треба «потренувати» класифікатор h_j, який обмежується використання однієї ознаки. Похибка оцінюється з урахуванням $w_t, \epsilon_{1,i} = \sum_i w_i \, |h_j(x_i) - y_i|$; (3)
  4. Обираємо класифікатор h_j з відповідним найменшим значенням помилки ϵ_t;

5. Перераховуємо значення ваги: $w_{t+1,i} = w_{t+1,i}\,\beta_t^{1-e_i}$, (4)

де $e_i = 0$, якщо x_i класифіковано правильно та $e_i = 1$, якщо класифікована неправильно та в цьому випадку $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

- Остаточний вирішальний класифікатор буде таким [5] (5):

$$h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0, & \text{якщо протилежне першому} \end{cases}, \text{ де } \alpha_t = log\frac{1}{\beta_t} \quad (5)$$

Як було сказано раніше навчання проводиться на [5] двох типах зображень і в нашому випадку позитивними є зображення обличь, а негативними – зображення без граней, це нам дає змогу підготувати класифікатор. Вся основна робота [5] проходить за допомоги так званого «фільтра», використовуючи який вилучаються особливості із поточного зображення, також варто сказати, що ця система є подібною до згорткового ядра. Ці фільтри представлені на рисунку 2.7.

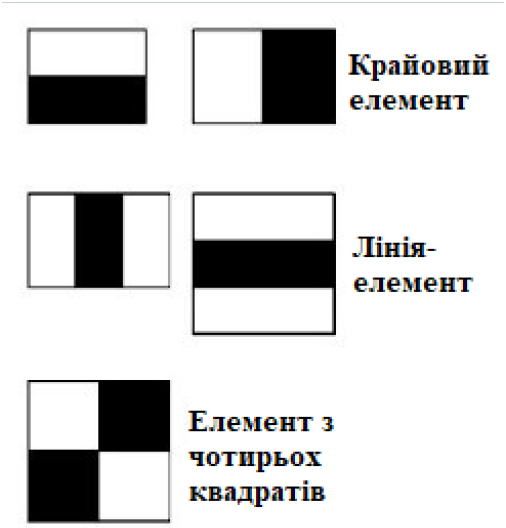

Рисунок 2.7 – Вигляд фільтрів Haar

Представлені вище фільтри [5] накладаються на зображення перевіряючи однин сектор (вікно) і далі переходять на інший. Далі [5] для кожного сектору іде обрахунок інтенсивності кожного пікселя за його відповідністю чорному та білому кольорам. Наступним кроком [8] отримується значення витягнутої ознаки,

що обраховується як різниця двох сум отриманих на минулому етапі. В найкращому випадку значення ознаки повинно показувати її релевантність. Також необхідно розуміти, що інтенсивність пікселів не буває однозначно рівною чорному чи білому кольорам, може виникати ситуація зображена на рисунку 2.8.

Рисунок 2.8 – Різна інтенсивність пікселів

Але від цього не змінюється стан речей, чим вищим є результат різниці, тим більшою є ймовірність сектора бути потрібною ознакою.

Далі ми стикаємося [5] з великою кількістю повернутих обрахунків з секторів, які треба оптимізувати. І тут на допомогу приходить концепція Integral Image – це особлива структура даних та алгоритм підрахунку значень у прямокутній підмножині сітки, який спровокує зменшення кількості обчислень для отримання потрібного результату.

На наступному кроці [5] оптимізації ми повертаємося до класифікаторів і завдяки технології Adaboost створюється так званий «сильний» класифікатор з декількох «слабких», а об’єднання проводиться за спільними найкращими характеристиками.

Останнім кроком [5] оптимізації виступає наша каскадність, яка існує для правильного групування ознак за різними стадіями класифікаторів і подальшого їх застосування по черзі у відповідних секторах. Це дає змогу відкинути відповідні

сектори і застосування означень для них, а отже це створює прецедент більш економічного використання пам’яті.

Після всієї обробки [5] зображення, використовуючи програмні та графічні можливості, «малюється» відповідний прямокутник чи овал, який показує нам знайдене обличчя.

2.3. Метод опорних векторів

SVM [6] – це алгоритм навчання з учителем. Головна мета SVM як класифікатора - знайти рівняння роздільної гіперплощини (6)

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_0 = 0 \quad (6)$$

у просторі R^n, яка розділила б два класи якимось оптимальним чином [5]. Загальний вид [8] перетворення F об'єкта x у мітку класу Y(7):

$$F(x) = \text{sign}(w^Tx - b). \quad (7)$$

Пам'ятатимемо, що ми позначили (8)

$$w = (w_1, w_2, \dots, w_n), b = -w_0. \quad (8)$$

Після налаштування [6] ваг алгоритму w і b (навчання), всі об'єкти, які потрапляють по одну сторону від побудованої гіперплощини, передбачатимуть як перший клас, а об'єкти, що потрапляють по інший бік — другий клас.

Усередині функції [6] sign() стоїть лінійна комбінація ознак об'єкта з вагами алгоритму, саме тому SVM відноситься до лінійних алгоритмів. Розділяючу гіперплощину можна [8] побудувати різними способами, але в SVM [10] ваги w і b налаштовуються таким чином, щоб об'єкти класів лежали якнайдалі від роздільної гіперплощини. Іншими словами, алгоритм максимізує відступ (англ. margin) між

гіперплощиною та об'єктами класів, які розташовані найближче до неї. Такі об'єкти називають опорними векторами (рисунок 2.7). Звідси й назва алгоритму.

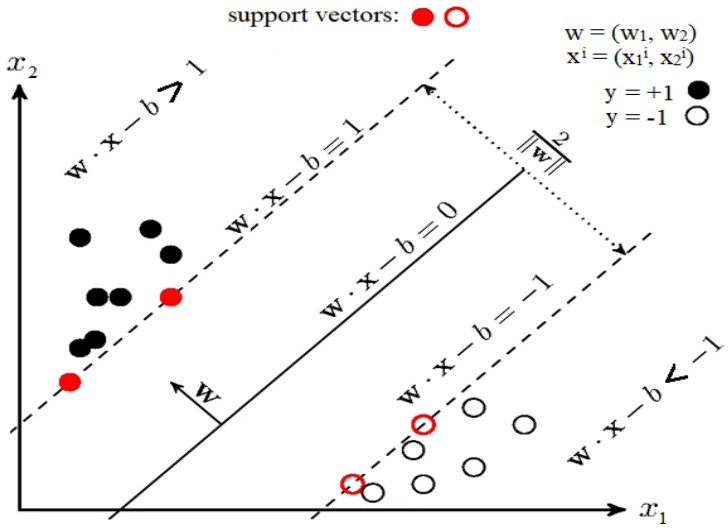

Рисунок 2.7 - Візуалізація методу опорних векторів

Алгоритм нової стратегії для вибірок навчання складається з наступних кроків:

1. Визначаємо значення відстані від кожної до всіх інших, вважаючи що відстань до самої себе уявним максимум (нескінченність);
2. Пошук найближчого за відстанню сусіда;
3. Визначаємо знак сумісності класу [8] з його найближчими сусідами з віднесенням до двох протилежних категорій(-1; 1);
4. Видалення знайденого гетерогенного вектору [8], який є несумісним зі своїм найближчим сусідом.

Після застосування цього алгоритму на виході ми отримуємо модернізовані навчальні вибірки.

2.4. Модифікація алгоритму Haar

До етапу навчання з SVM [10] зображення проходить три стадії обробки: розбирання за RGB простором, розбирання за YCbCr простором, розбирання за HLS простором. Потім вони всі накладаються для отримання результативного зображення, що пройде через обробку Haar фільтрів.

Далі обираються негативні фотографії, що не містять жодних обличчя в цілому та позитивні з нормальними зображеннями, далі все проводиться за алгоритмом, описаним вище, і ми отримуємо так званий потужний класифікатор. Далі до справи доєднається метод опорних векторів, який робить свій вклад у алгоритмізацію машинного навчання для вирішення проблем класифікації, що часто виникають у таких випадках. В нашому випадку це буде використовуватись у вигляді додавання нових прикладів до даних навчання, класифікуючи одержані екземпляри зображень у різні категорії. Тому тут використовується так зване навчання без контролю, де буде відбуватися пошук кластеризації у групах, а потім відображатися у вигляди даних для визначених підгруп. Структура роботи системи можна побачити на рисунку 2.8 нижче.

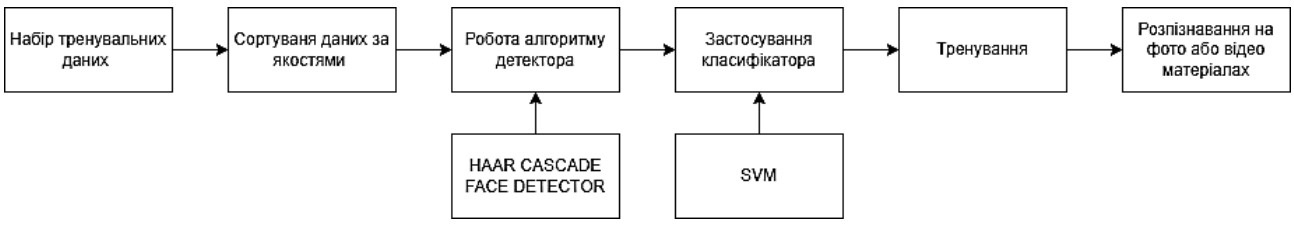

Рисунок 2.8 - Архітектура запропонованої системи

Запропонований алгоритм визначає обличчя за допомогою вичерпного сканування зображення з візерунками, схожими на обличчя, у багатьох можливих масштабах, поділ вихідного зображення на часткові зображення, що

перекриваються, і їх класифікація використання SVM для визначення відповідного класу. Обробляються кілька масштабів досліджуваного вікна, взяті з масштабованих версій оригіналу. Перед зберіганням зображення відбуваються деякі етапи попередньої обробки, такі як маскування, освітлення та складання гістограми вирівнювання. У процесі маскування візерунки непотрібного фонового шуму зменшуються у регіоні об’єктів зацікавленості. Далі у гістограмі використовується вирівнювання, яке керує розподілом кольорів у зображеннях. Зображення обличчя класу та зворотних класів використовуються для навчання SVM з використанням значення ядра та верхньої межі поля. Як тільки поверхня рішення була отримана під час навчання, система виконання визначає зображення, які не містять обличчя, і ці неправильні класифікації зберігає, щоб можна було їх використовувати у вигляді негативних прикладів на наступних етапах навчання. Щоб підвищити точність виявлення обличчя, ми можемо використовувати негативні приклади для навчання класу неправильної класифікації. Доступно багато зображень без обличчя, які можна навчити, використовуючи SVM. Негативні зображення обличчя багатші та ширші, ніж зображення з обличчям.

2.5. Алгоритм HOG та його застосування

Дескриптор функції [7] — це представлення зображення або фрагмента зображення, який спрощує зображення шляхом вилучення корисної інформації та викидання зайвої інформації.

Як правило, дескриптор функції [7] перетворює зображення розміром ширина x висота x 3 (канали) у вектор ознаки/масив довжиною n. У випадку дескриптора ознак HOG вхідне зображення має розмір 64 x 128 x 3, а вихідний вектор ознак має довжину 3780.

Майте на увазі, що дескриптор HOG [7] можна розрахувати для інших розмірів, але в цій публікації я дотримуюся цифр, наведених в оригінальній статті, щоб ви могли легко зрозуміти концепцію на одному конкретному прикладі.

Все це звучить добре [7], але що «корисно», а що «зайве»? Щоб визначити [7] «корисний», нам потрібно знати, для чого він «корисний»? Очевидно, що вектор ознак не корисний для перегляду зображення. Але це дуже корисно для таких завдань, як розпізнавання зображень і виявлення об’єктів. Вектор ознак [7], створений цими алгоритмами, коли він подається в алгоритми класифікації зображень, такі як Support Vector Machine (SVM), дає хороші результати.

Але які типи «функцій» [7] корисні для класифікаційних завдань? Давайте обговоримо це питання на прикладі. Припустимо, ми хочемо побудувати детектор об’єктів, який виявляє ґудзики сорочок і пальто.

Ґудзик має круглу форму [7] (може виглядати еліптичною на зображенні) і зазвичай має кілька отворів для пришивання. Ви можете запустити детектор країв на зображенні кнопки та легко визначити, чи це кнопка, просто подивившись лише на зображення краю. У цьому випадку інформація про краї є «корисною», а інформація про колір – ні. Крім того, функції також повинні мати розрізнювальну силу. Наприклад, хороші елементи, витягнуті із зображення, повинні відрізняти кнопки від інших круглих об’єктів, таких як монети та автомобільні шини.

У дескрипторі ознак HOG [7] розподіл (гістограми) напрямків градієнтів (орієнтованих градієнтів) використовуються як ознаки. Градієнти (похідні x і y) зображення є корисними, оскільки величина градієнтів велика навколо країв і кутів (областей різкої зміни інтенсивності), і ми знаємо, що краї та кути містять набагато більше інформації про форму об’єкта, ніж плоскі області.

Висновки до другого розділу

Представлено модифікацію обраного алгоритму розпізнавання обличчя, що складається з наступних кроків:

- Аналіз фотозображення за трьома різними кольоровими моделями (RGB, YCbCr та HLS) і поєднання обробок у єдиний результат;
- Заміна поточного алгоритму класифікації на SVM зі зміненою стратегією вибірок для навчання, що аналізує найближчі до кордону точки (перевірка на гетерогенність);
- Застосування методу генерації негативних фотозображень з наявних позитивних.

Визначено алгоритм HOG оптимальним варіантом для прямої ідентифікації обличчя. Показано обрані засоби програмної розробки та розписано всі переваги використаних інструментів.

3. ОПИС РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ ІДЕНТИФІКАЦІЇ ОБЛИЧЧЯ

3.1 Архітектура розробленої системи

Перед написанням програмних засобів розпізнавання та ідентифікації обличчя визначено з яких частин складатиметься розроблювана система і їх перелік наступний:

- Візуальний модуль (UI interface), що призначений для користувацького використання, так званого адміністрування системи;
- «Транзішн» модуль, він відповідає за взаємодію візуального модуля з іншими програмними модулями, що є частиною Backend (сукупність програмних рішень, що не видимі користувачі, але виконують всю основну роботу системи);
- Модуль розпізнавання обличчя;
- Модуль ідентифікації обличчя;
- Модуль взаємодії з базою даних, в деякій літературі такі компоненти називають серверною частиною;
- База даних.

Головною задачею при створенні стабільно працюючої системи є правильний розподіл програмних ресурсів та жорстка структура проекту, що виключає можливості непередбачуваної поведінки.

В сучасних комерційних програмних продуктах найпопулярнішим патерном архітектури є Model-View-Controller (MVC), який себе зарекомендував як один з найкращих парадигм поділення систем на два основних робочих модуля (Frontend та Backend) з модулем комунікації між першими двома (рисунок 3.1).

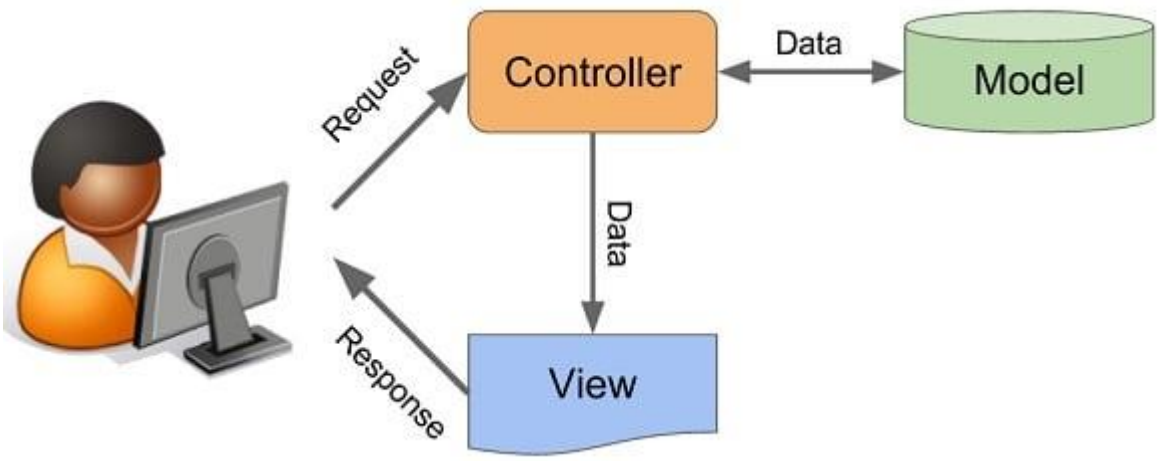

Рисунок 3.1 – Архітектура Model-View-Controller

У програмній реалізації роль View виконує візуальний модуль – сукупність файлів написаних мовою QML, які дають змогу користувачу взаємодіяти з продуктом і отримувати інформативний зворотній зв’язок, бо будь-яка інформація краще сприймається візуально.

Роль Model на себе беруть одразу декілька різних модулів: модуль розпізнавання обличчя, модуль ідентифікації обличчя, модуль взаємодії з базою даних та в деякій літературі сюди також відносять базу даних. Всі вищезгадані компоненти займаються найскладнішими та найвизначнішими задачами програмного продукту, тобто виступають його серцем та мозком, бо життєдіяльність системи без цих частин не можлива в принципі. Також варто зазначити, що розподіл задач на компоненти в середині модулю Backend дозволяє легко робити налагодження системи та спрощує тестування системи, як-от функціональне (англійською functional) та нефункціональне тестування (англійською non-functional).

І останню роль Controller на себе бере «Транзішин» модуль, що передає інформації між першими двома глобальними модулями, спрощуючи комунікацію завдяки зручним інструментам QT Framework.

З огляду представленої інформації вище структура проекту у Visual Studio виглядає наступним чином (рисунок 3.2).

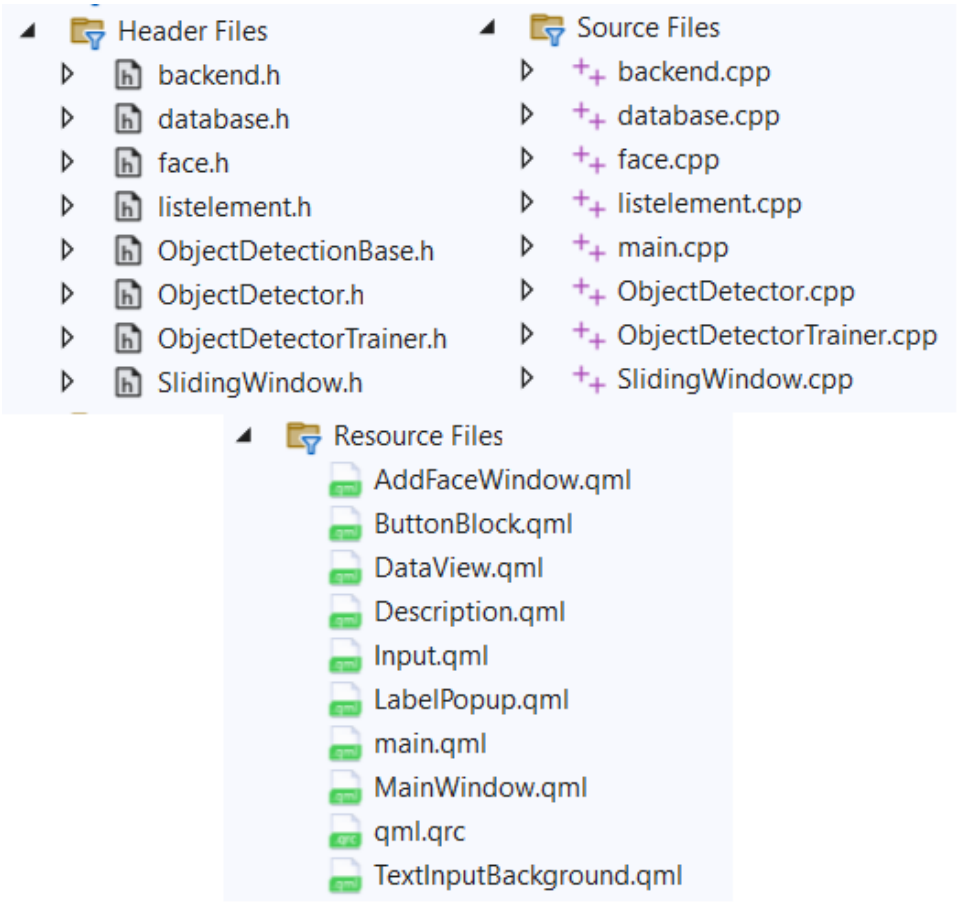

Рисунок 3.2 – Вигляд структура проекту

3.2 Сутність та структура розпізнавального модуля

Розпізнавальний модуль спирається на модифікований алгоритм Haar та завчасно створену модель ідентифікаторів, реалізація модуля складається з таких файлів:

- face.h (хедер);

- face.cpp (copc).

У «хедері» знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);
- Прототипи функції, реалізація яких описана у «copc» файлі.

У copc файлі знаходиться функція на ім’я face_detection (string image).

Ця функція приймає на вхід рядок типу string зі шляхом до фотозображення, виконується обробка фото для подальшого пошуку обличчя, далі відбувається сам пошук з використанням заданої моделі, знайдене обличчя позначається округлими фігурами та весь цей результат повертається з функції для подальшого використання іншими частинами програми. Приклад визначення обличчя на фотозображенні можна побачити на рисунку 3.3.

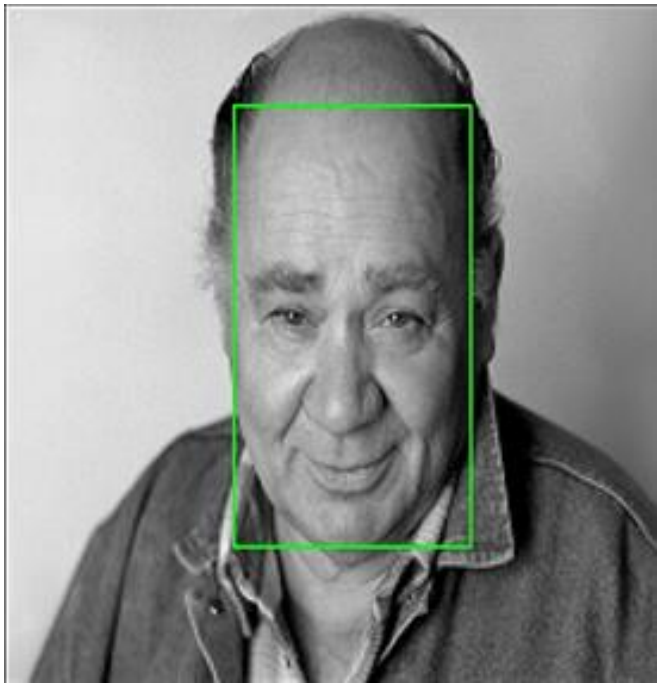

Рисунок 3.3 – Приклад роботи модуля розпізнавання

3.3 Сутність та структура ідентифікаційного модуля

Ідентифікаційний модуль спирається на алгоритм HOG та моделі ідентифікаторів обличь, що додаються до бази даних поступово під час роботи програми, реалізація модуля складається з таких файлів:

- HOGDetector.h (хедер);
- HOGDetector.cpp;
- HOGDetectorBase.h (хедер);
- HOGDetectorTrainer.h (хедер);
- HOGDetectorTrainer.cpp.

У HOGDetector.h знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);
- Метод start_identification(std::vector<std::pair<int, std::string>> detectors);
- Функція identification (const std::vector<std::pair<int, std::string>>& detectors, std::vector<cv::Mat> images, int start, int end, std::vector<int>& result).

У HOGDetectorTrainer.h знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);
- Метод set_pathes(struct HOGDetectorBase::Pathes pathes);
- Метод set_options(struct HOGDetectorBase::TrainerOptions options);
- Метод create_train_dataset();
- Метод training(string path);
- Метод generate_backgrounds(Size size).

HOGDetectorBase.h є єдиним «хедером», який не має ніяких «сорс» реалізацій, бо по факту є базовою класом, від якого відбувається подальше наслідування.

У HOGDetector.cpp знаходяться реалізації метода та функції з HOGDetector.h, а в HOGDetectorTrainer.cpp – всі методи з HOGDetectorTrainer.h.

Метод start_identification отримає вхідні дані пар ідентифікаторів для перевірки, створює сцену, позначає зону в якій має відбуватись ідентифікація і очікує на зміну ситуації з боку UI та інших девайсів, що контролюють взаємодію з користувачем. При отриманні відповідних сигналів всередині спрацьовуються відповідні слоти і застосовується функція identification, завдяки їй знаходяться відповідні індивідуальні номери в системи для подальшого запиту в дата базу щодо ідентифікованих користувачів.

Функція identification займається безпосереднім порівнянням даних користувача, що в даних конкретний момент намагається ідентифікуватись у системі, з тими ідентифікатори, які поступово дістаються з бази даних, поступовість зумовлена намаганнями не перенавантажувати системні ресурси комп’ютера. Дані користувача, які отримуються на вході проходять обробку до спрощених ідентифікаторів для порівняння.

Метод set_pathes отримаю на вхід значення шляхів, що мають відношення до ідентифікатор, і відповідно ініціалізує їх у новоствореному об’єкті класу тренування ідентифікації.

Метод set_options отримаю на вхід значення шляхів, що мають відношення до ідентифікатор, і відповідно ініціалізує їх у новоствореному об’єкті класу тренування ідентифікації.

Метод create_train_dataset займається видозміною отриманих користувацьких даних для так званого їх поглиблення та урізноманітнення, аби збільшити ймовірність позитивного ідентифікування. При користувацькому додаванні суб’єкта робиться 5 фотографій для отриманні прототипів різної якості, що потім будуть вдосконалюватись додаванням різних згенерованих фонів. Приклад такої роботи наведений на рисунку 3.4 нижче.

Рисунок 3.4 – Приклад роботи метода create_train_dataset

Метод generate_backgrounds займається генерацією фонів, що далі використовуються для створення «датасетів». Фактично під час генерації створюється до 120 фігур різної форми та кольору, які в довільному порядку накладаються одна на одну, також на задньому плані згенерованої картинки можна побачити чорний базовий фон. Існують інші варіанти таких генераторів, які замість однокольорових фігур використовують малюнки ландшафтів різної роздільної здатності. Приклад згенерованих картинок з фігурками можна побачити на рисунку 3.5.

Рисунок 3.4 – Приклад роботи метода generate_backgrounds

Метод training займається тренуванням для створення ідентифікаційної моделі, використовуючи раніше перелічені методи, що реалізовані в одному файлі. На виході роботи цього методу ми отримуємо готову потреновану модель, що зберігається у вигляді байтів у базі даних. Зберігаються такі файли у окремій директорії з розширенням файлі svm і це виглядає наступним чином (рисунок 3.5).

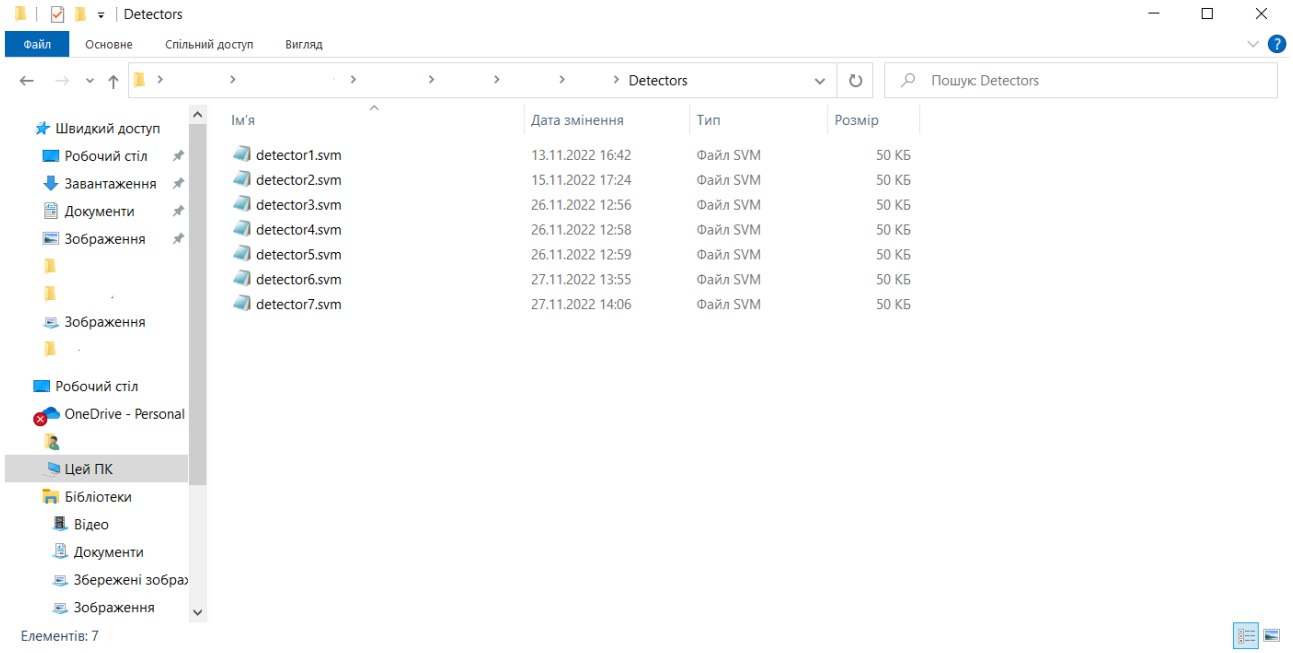

Рисунок 3.5 – Директорія з моделями для ідентифікації

Створення цих моделей відбувається ітераційно і залежить від того, яку точність виставляє розробник. Кількість ітерацій формується автоматично під час обробки зображень, тобто робота над зображенням буде продовжуватись до тих пір, поки не буде досягнуто конкретного значення можливої похибки. Це значення похибки напряму впливає на час використання системних ресурсів та на їхню кількість, останнє впливає на стабільність роботи системи в цілому. Тому одним з можливих варіантів покращення роботи програми є розподілення роботи окремих алгоритмів на потоки, аби паралельно виконувати подібну роботу. Номінально робота була розподілена 8 потоків, але фактичним є число 7, бо перший потік іде на адміністрування. Приклад «логування» ітерацій навчання можна побачити на рисунку 3.6.

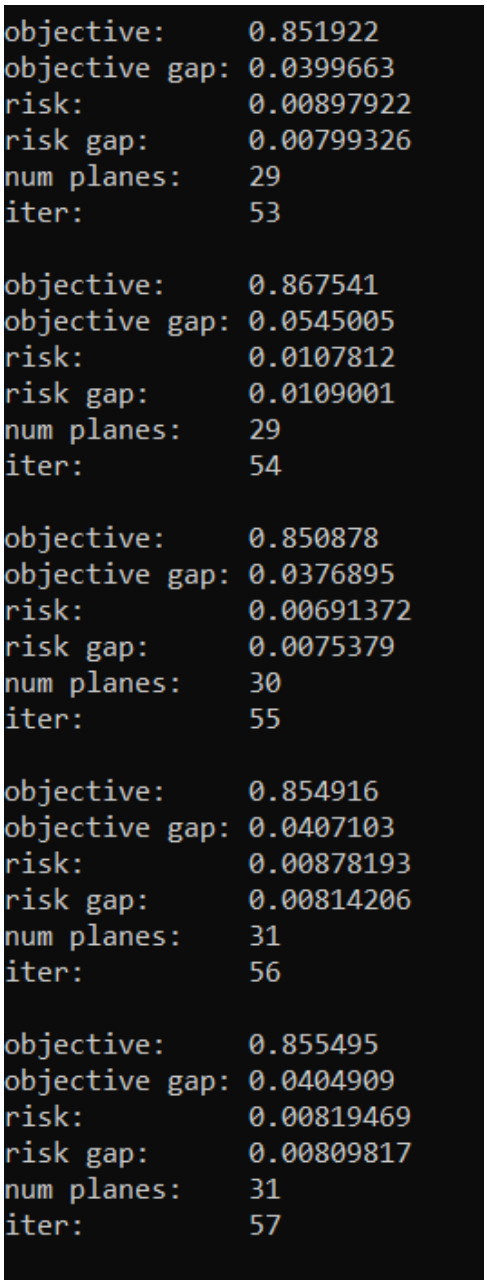

Рисунок 3.6 - «Логування» ітерацій навчання, що показують точність можливої похибки ідентифікації

3.4 Сутність та структура модуля комунікації з базою даних

Модуль взаємодії з базою даних виконує важливу роль контролю комунікації всередині системи, бо точність роботи цього модуля стоїть на 2 місці шкали

пріоритетів після модулів розпізнавання та ідентифікації, реалізація цього компонента складається з таких файлів:

- database.h (хедер);
- database.cpp;
- list_element.h (хедер);
- list_element.cpp.

У database.h знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);
- database_is_ruined() const;
- create_database();
- get_index_number(Table table) const;
- get_image_path(const int identificator);
- delete_from_database_by_id(const int identificatory, Table table);
- add_list_element(QString name, QString description, QString detectorPath, QString imagePath, QString place, int count);
- edit_list_element(int identificator, QString name, QString description, QString imagePath, QString place, int count);
- generate_detect_id() const;
- search_by_text(QString line, boo parameter);
- search_by_detector (const vector<int>& identificators);
- get_detector_path().

У list_element.h знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);

- ListElement (int id, QString name, QString imagePath, QString place, int count, QString description, QObject *parent);
- get_name() const;
- get_image_path() const;
- get_description() const;
- get_count() const;
- get_id() const;
- get_path() const.

У database.cpp знаходяться реалізації всіх методів з database.h, а в list_element.cpp – всі методи з list_element.h.

Метод database_is_ruined перевіряє стан роботи бази даних, якщо при перевірці виникають питання з приводу готовності окремих компонентів, то це метод поверне значення true, якщо навпаки – false.

Метод create_database створює SQL-запит на створення таблиці елементів з наступними полями:

- id int, name;
- varchar(1000);
- desc varchar(1000000);
- place varchar(1000);
- count int;
- imageId int;
- detectorId int.

Запит можна побачити на рисунку 3.7, його визначено інструментами QT Framework.

```
QSqlQuery query;
for (const auto& const QString & iter : this->tables) {
    if (iter == "Description") {
        continue;
    }
    if (!query.exec(query:"CREATE TABLE " + iter + "(id int, path varchar(1000))")) {
        qDebug() << query.lastError();
        return false;
    }
}
if (!query.exec(query:"CREATE TABLE Description (id int, name varchar(1000), desc varchar(1000000), place varchar(1000), count int, imageId int, detectorId int)")) {
    qDebug() << query.lastError();
    return false;
}
```

Рисунок 3.7 – Запит на створення таблиці

Метод get_index_number створює SQL-запит для визначення максимального значення індексу, що присутній у поточній таблиці, у випадку пустоти повернеться значення 0 (рисунок 3.8).

```
QSqlQuery query;
QString command = "SELECT max(id) FROM %1";
if (!query.exec(query:command.arg(tables[static_cast<int>(table)]))) {
    qDebug() << query.lastError();
    return -1;
}
query.next();
```

Рисунок 3.8 – Запит для визначення максимального значення індексу

Метод get_image_path створює SQL-запит для отримання значення шляху картинки, що лежить в елементі таблиці з визначеним ідентифікатором, що отримано як вхідні дані(рисунок 3.9).

```
QSqlQuery query;
QString command = "SELECT path FROM %1 WHERE id=:id";
query.prepare(query:command.arg(tables[static_cast<int>(Table::Images)]));
query.bindValue(placeholder:":id", val:id);
if (!query.exec()) {
    qDebug() << query.lastError();
    return "";
}
query.next();
```

Рисунок 3.9 – Запит для отримання значення шляху картинки

Метод delete_from_database_by_id створює SQL-запит для видалення елемента з визначеним ідентифікатором та вибраної таблиці, до якої він належить (рисунок 3.10).

```
QSqlQuery query;
QString command;
if (table == Table::Images || table == Table::Detectors) {
    command = "SELECT path FROM %1 WHERE id=:id";
    query.prepare(query: command.arg(tables[static_cast<int>(table)]));
    query.bindValue(placeholder: ":id", val: id);
    if (!query.exec()) {
        qDebug() << query.lastError();
    }
    else if (query.next()) {
        QString file = query.value(i: 0).toString();
        if (file.contains(s: "file:")) {
            file.remove(s: "file:");
        }
        QFile::remove(fileName: file);
    }
    query.clear();
}

command = "DELETE FROM %1 WHERE id=:id";
query.prepare(query: command.arg(tables[static_cast<int>(table)]));
query.bindValue(placeholder: ":id", val: id);
if (!query.exec()) {
    qDebug() << query.lastError();
}
```

Рисунок 3.10 – Запит для видалення елемента

Метод add_list_element створює SQL-запит для додавання нового елементу у визначену таблицю з полями, що можна побачити на рисунку 3.11, вид запиту визначено інструментами QT Framework.

```
command = "INSERT INTO %1 values (:id, :name, :desc, :place, :count, :imageId, :detectorId)";
query.prepare(query: command.arg(tables[static_cast<int>(Table::Description)]));
query.bindValue(placeholder: ":id", val: elementId);
query.bindValue(placeholder: ":name", val: name);
query.bindValue(placeholder: ":desc", val: description);
query.bindValue(placeholder: ":place", val: place);
query.bindValue(placeholder: ":count", val: count);
query.bindValue(placeholder: ":imageId", val: imageId);
query.bindValue(placeholder: ":detectorId", val: detectorId);
if (!query.exec()) {
    qDebug() << query.lastError();
    return false;
}
```

Рисунок 3.11 – Поля запиту на додавання елементу до таблиці

Метод edit_list_element створює SQL-запит для зміни полів елементу визначеного вхідним ідентифікатором, поля що будуть змінені визначаються користувачем застосунку. Запит можна побачити на рисунку 3.12, його визначено інструментами QT Framework.

```
command = "UPDATE %1 SET name = :name, desc = :desc, place = :place, count = :count WHERE id=:id";
query.prepare(query: command.arg(tables[static_cast<int>(Table::Description)]));
query.bindValue(placeholder: ":id", val: id);
query.bindValue(placeholder: ":name", val: name);
query.bindValue(placeholder: ":desc", val: description);
query.bindValue(placeholder: ":place", val: place);
query.bindValue(placeholder: ":count", val: count);
if (!query.exec()) {
    qDebug() << query.lastError();
    return false;
}
```

Рисунок 3.12 – Зміна полів елементу у таблиці

Метод search_by_text створює SQL-запит для пошуку елементів таблиці, що мають в назві або в описанні суб’єкту текст, що було отримано від користувача під час введення на клавіатурі(рисунок 3.13).

```
QSqlQuery query;
if (search.isEmpty()) {
    QString command = "SELECT id, name, imageId, place, count, desc FROM %1";
    query.prepare(query:command.arg(tables[static_cast<int>(Table::Description)]));
}
else  if (searchByDescription) {
    QString command = "SELECT id, name, imageId, place, count, desc FROM %1 WHERE name LIKE :search OR  desc LIKE :search";
    query.prepare(query:command.arg(tables[static_cast<int>(Table::Description)]));
    query.bindValue(placeholder:":search", val:QString("%%1%").arg(search));
}
else {
    QString command = "SELECT id, name, imageId, place, count, desc FROM %1 WHERE name LIKE :search";
    query.prepare(query:command.arg(tables[static_cast<int>(Table::Description)]));
    query.bindValue(placeholder:":search", val:QString("%%1%").arg(search));
}
```

Рисунок 3.13 – Запит для пошуку елементів таблиці, що мають в назві або в описанні суб’єкту текст

Метод search_by_detector створює SQL-запит для пошуку елементів таблиці, що мають такий самий ідентифікатор, який було подано у вигляді вхідних даних(рисунок 3.14).

```
QSqlQuery query;
QList<ListElement*> result;
QString command = QString("SELECT id, name, imageId, place, count, desc FROM %1 WHERE detectorId=:search").arg(tables[static_cast<int>(Table::Description)]);
for (const auto& const int & iter : ids) {
    query.prepare(query:command);
    query.bindValue(placeholder:":search", val:iter);
    if (!query.exec()) {
        qDebug() << query.lastError();
        return result;
    }
    while (query.next()) {
        result.push_back(new ListElement(query.value(i:0).toInt(), query.value(i:1).toString(), imagePath(id:query.value(i:2).toInt()), query.value(i:3).toSt
    }
    query.clear();
}
```

Рисунок 3.14 – Запит для пошуку елементів таблиці, що мають такий самий ідентифікатор

Метод get_detector_path створює SQL-запит для отримання всіх ідентифікаторів елементів таблиці(рисунок 3.15).

```
QSqlQuery query;
std::vector<std::pair<int, std::string>> result;
QString command = "SELECT * FROM %1";
if (!query.exec( query: command.arg(tables[static_cast<int>(Table::Detectors)]))) {
    qDebug() << query.lastError();
    return result;
}
```

Рисунок 3.15 – Запит для отримання всіх ідентифікаторів

Конструктор ListElement створює елемент C++ структури з наступними полями:

- int id;
- QString name;
- QString imagePath;
- QString place;
- int count;
- QString description;
- QObject *parent.

Метод get_name повертає значення поля name поточного об’єкта, що витягнули з контейнера.

Метод get_image_path повертає значення поля imagePath поточного об’єкта, що витягнули з контейнера.

Метод get_description повертає значення поля description поточного об’єкта, що витягнули з контейнера.

Метод get_count повертає значення поля count поточного об’єкта, що витягнули з контейнера.

Метод get_id повертає значення поля id поточного об’єкта, що витягнули з контейнера.

Метод get_path повертає значення поля path поточного об’єкта, що витягнули з контейнера.

3.5 Сутність та структура модуля Controller

Цей модуль займає особливе положення в структурі проекту, бо виконує дуже важливу комунікативну роль між «бекендом» та «фронтендом». Більшість дій модуля у цьому проекті складається з передачі сигналу до «бекенду» про те, що треба розпочати якійсь ємнісний процес, наприклад, обробку фотозображення для подальшої ідентифікації, а коли подібні операції з боку «бекенду» завершуються, то Controller передає зворотній сигнал та дані, що були отримані в процесі роботи модулів, до «фронтенту» і той у свою чергу показує зміни.

Реалізація модуля складається з таких файлів:

- backend.h (хедер);
- backend.cpp (сорс).

У «хедері» знаходяться такі компоненти:

- Бібліотеки двох видів: стандартні C++ та інші сторонні (включно зі своїми);
- Namespaces (простори імен);
- Метод search_by_name(QString search, bool option);
- Метод add_face(QString name, QString description, QString place, int count, QString image);
- Метод identificate();
- Метод edit_element(int id, QString name, QString description, QString place, int count, QString image);
- Метод delete_element(const int identificatory);

- QT-сигнал database_changed();
- ().

У «сорс» файлі знаходиться реалізації методів та сигналів, що оголошені у «хедер».

Метод search_by_name отримує значення вхідного рядка для пошуку користувачів у дата базі, викликає метод пошуку за текстом та передає туди завчасно отримане значення. Також цей метод визначає час роботи викликаного алгоритму пошуку.

Метод add_face отримує на вхід значення опису особистості, що додається в систему, потім всередині викликаються «бекенд» методи у наступному порядку:

1. Метод, що визначає шлях зберігання, set_path;
2. Метод, що визначає опції тренувальника, set_options;
3. Метод, що створює тренувальні зображення з отриманих раніше згенерованих фонів, create_train_dataset;
4. Метод тренування, що створює потреновану модуль за відповідну кількість ітерацій.

Метод identificate викликає метод детектування, що повертає ідентифікатор користувача, що намагається зайти у систему, а потім відбувається порівняння цього ідентифікатора з наявним списком шаблонів програмованого продукту.

Метод edit_element отримує на вхід значення полів, що мають бути змінені у елемента з відповідним ідентифікатором, відбувається пошук даного елемента у загальному списку і при позитивному результаті відбувається виклик метода, що займається безпосередньою зміною поточних значень.

Метод delete_element викликає «бекенд» метод, що має видалити за отриманим ідентифікатором знайдений елемент з контейнера, де зберігаються такі самі шаблони.

QT-сигнал database_changed передає сигнал на «фронтент» про те, що наповнення бази даних змінилось.

QT-сигнал model_changed передає сигнал на «фронтент» про те, що наповнення зображуваної моделі змінилось.

3.6 Сутність та структура візуального модуля

Візуальний модуль є головною складовою програми для користувача, бо він контактує тільки з цією частиною системи, саме тому UI (User Interface) має бути максимально приємним та інтуїтивно зрозумілим у використанні. Часто UI пишуть на кроссплатформенних інструментах розробки задля збільшення охоплення, як регіонального так і девайсного. не варто забувати що самому розробнику інструмент повинен подобатись за набором складових та зручність написання коду. Всі ці вище перераховані пункти вплинули на вибір мови написання візуального модуля і нею стала QML (Qt Meta Language або Qt Modeling Language).

Візуальний модуль складається з наступних компонентів з розширенням «.qml»:

- AddFaceWindow.qml;
- ButtonBlock.qml;
- DataView.qml;
- Description.qml;
- Input.qml;
- LabelPopup.qml;
- Window.qml;
- TextInput.qml;
- main.qml;

Останнім файлом, що зберігає в собі наявність всіх файлів наведених вище та їх конфігурацію, є qml.qrc (рисунок 3.16). Зазвичай проекти створені в QT Creator «парсяться», використовуючи файл з розширенням «.pro», який у свою чергу спирається на цей самий файл qml.qrc.

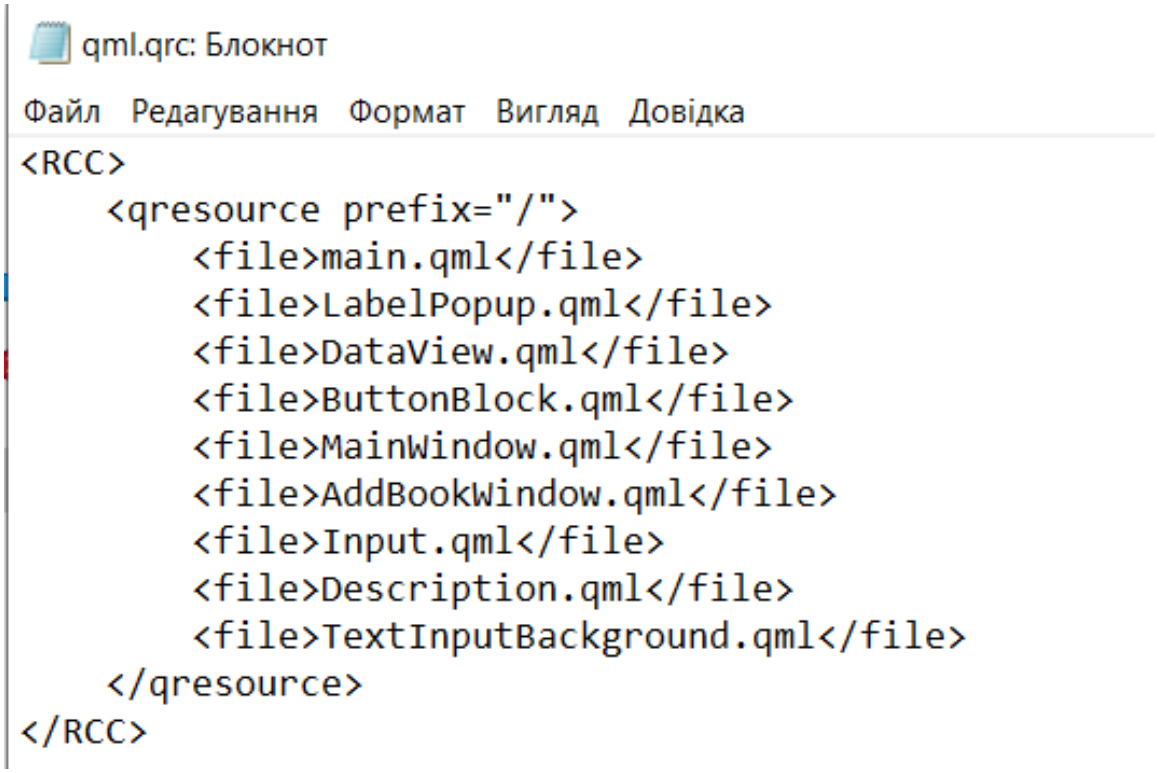

Рисунок 3.16 – Наповнення файлу qml.qrc

Файл main.qml є головною графічною компонентною, що об’єдную у собі всі інші другорядні графічні рішення, створюючи їх систематичність та впорядкованість.

Файл AddFaceWindow.qml є компонентною графічного вікна, в якому відбувається конфігурація даних користувача, що додадуть в систему, і перехід до подальшого вікна сканування обличчя.

Файл ButtonBlock.qml містить в собі блок кнопок, що відображені на першому вікні, і вони дають змогу переключатися в інші ключові режими програми, такі як ідентифікування чи додавання обличчя.

Файл DataView.qml містить в собі графічне відображення листа елементів суб’єктів користування системи та інші компоненти, що впиваються на відображення його наповнення.

Файл Description.qml – це графічна компонента, що відображається при натисканні на елемент списку і відображує його фотозображення, коротку інформацію та кнопки, які запускають зміну поточного делегата у базі даних чи навіть спричиняють його видалення.

Файл Input.qml являє собою змінним рядком, який призначений для клавіатурного вводу адміністратора.

Файл LabelPopup.qml являє з себе випливаюче вікно, яке позначає що відбулась якась помилка.

Файл Window.qml це базовий компонент графічного вікна, що визначає розміри додатку, який бачить перед собою користувач, і показує в собі компоненти перелічені вище в залежності від умов закладених в алгоритм стека програми.

Файл TextInput.qml являє собою забарвленням заднього фону текстового рядка.

Основні графічні компоненти, що використані при написанні графічних конструктів та їх вигляд:

- Rectangle;

Це простого рівня графічний компонент, що представляє з себе квадрат, який може мати різну кольору гама, виступати фоновим або сукупним елементом для інших більш менших. Приклад такого елементу в коді показано на рисунку 3.17.

```
Rectangle {
    id: root
    radius: 5
    border.width: 2
    border.color: "black"
    readonly property alias searchByDescriptionChecked: searchByDescription.checked
    property alias searchText: searchField.text
    property alias viewModel: dataView.model
    signal itemClicked(var item)
```

Рисунок 3.17 – Приклад графічного елементу Rectangle

- Item;

Графічний елемент базового рівня, який виступає таким собі контейнером для більш специфічних компонентів. Приклад такого елементу в коді показано на рисунку 3.18.

```
Item {
    id: root
    property var backend: null
    signal close()
    readonly property alias nameField: name.text
    readonly property alias descriptionField: description.text
    readonly property alias countField: count.text
    readonly property alias placeField: place.text
    property var item: null
```

Рисунок 3.18 – Приклад графічного елементу Item

- Button;

Шаблонна графічна компонента, яка використовується тільки для позначення кнопки у вікні, має слот onClicked, який реагує на натискання і виконує програмний код написаний всередині його тіла. Приклад такого елементу в коді показано на рисунку 3.19.

```
Button {
    id: close
    text: "Close"
    anchors.left: root.left
    anchors.right: edit.left
    anchors.bottom: root.bottom
    anchors.top: description.bottom
    onClicked: root.close()
}
```

Рисунок 3.19 – Приклад графічного елементу Button

- Image;

Шаблонна графічна компонента, яка використовується тільки виведення малюнку у вікні з можливістю змінювати основні параметри зображення. Приклад такого елементу в коді показано на рисунку 3.20.

```
Image {
    id: image
    anchors.left: root.left
    anchors.right: root.right
    anchors.top: root.top
    height: 200

    source: Boolean(root.item) ? item.imagePath : ""
    fillMode: Image.PreserveAspectFit
    cache: false
}
```

Рисунок 3.20 – Приклад графічного елементу Image

- Input;

Шаблонна графічна компонента, яка використовується тільки для введення змінного рядка, що далі буде використовуватись для пошуку як ключова фраза або

інших подібних маніпуляцій. Приклад такого елементу в коді показано на рисунку 3.21.

```
Input {
    id: searchField
    anchors.left: dataView.left
    anchors.right: searchByDescription.left
    height: 50
    inputVerticalAlignment: TextArea.AlignVCenter
}
```

Рисунок 3.21 – Приклад графічного елементу Input

- MouseArea.

Невидимий елемент, який зазвичай використовується в поєднанні з видимим елементом, щоб забезпечити обробку миші для цього елемента. Приклад такого елементу в коді показано на рисунку 3.22.

```
MouseArea {
    anchors.top: element.top
    anchors.right: element.right
    anchors.left: element.left
    anchors.bottom: element.isCollapsed ? element.bottom : description.top
    onClicked: {
        if (!element.isCollapsed) {
            dataView.currentIndex = null;
        } else {
            dataView.currentIndex = modelData.id;
        }
    }
}
```

Рисунок 3.22 – Приклад графічного елементу MouseArea

Висновки до третього розділу

Створено чітку структуру проекту за правилами архітектури Model-Controller-View та рівномірно розподілено задачі між модулями (частинами) програмного продукту.

Код написано у стандартизованому вигляді, з огляду проектного мислення щодо делегування обов’язків розробки або повного передання підтримки та модифікування готового продукту в інші руки. Це проявляється у наступних пунктах:

- Коментування важливих речей серед коду, зазвичай маленькі пояснення за що відповідає параметр або функція;
- Використання стандартних архітектурних рішень фреймворку Qt для поєднання модулів;
- Створення інструментів для роботи з базою даних, що можуть підлаштовувати під інші типи баз або легко замінюватись на аналогічні.

4. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Тестування візуального модуля

Відкриваємо головне вікно програми та натискаємо кнопку «Add». Потім заповнюємо всі потрібні стрічки та починаємо розпізнавання (рисунок 4.1).

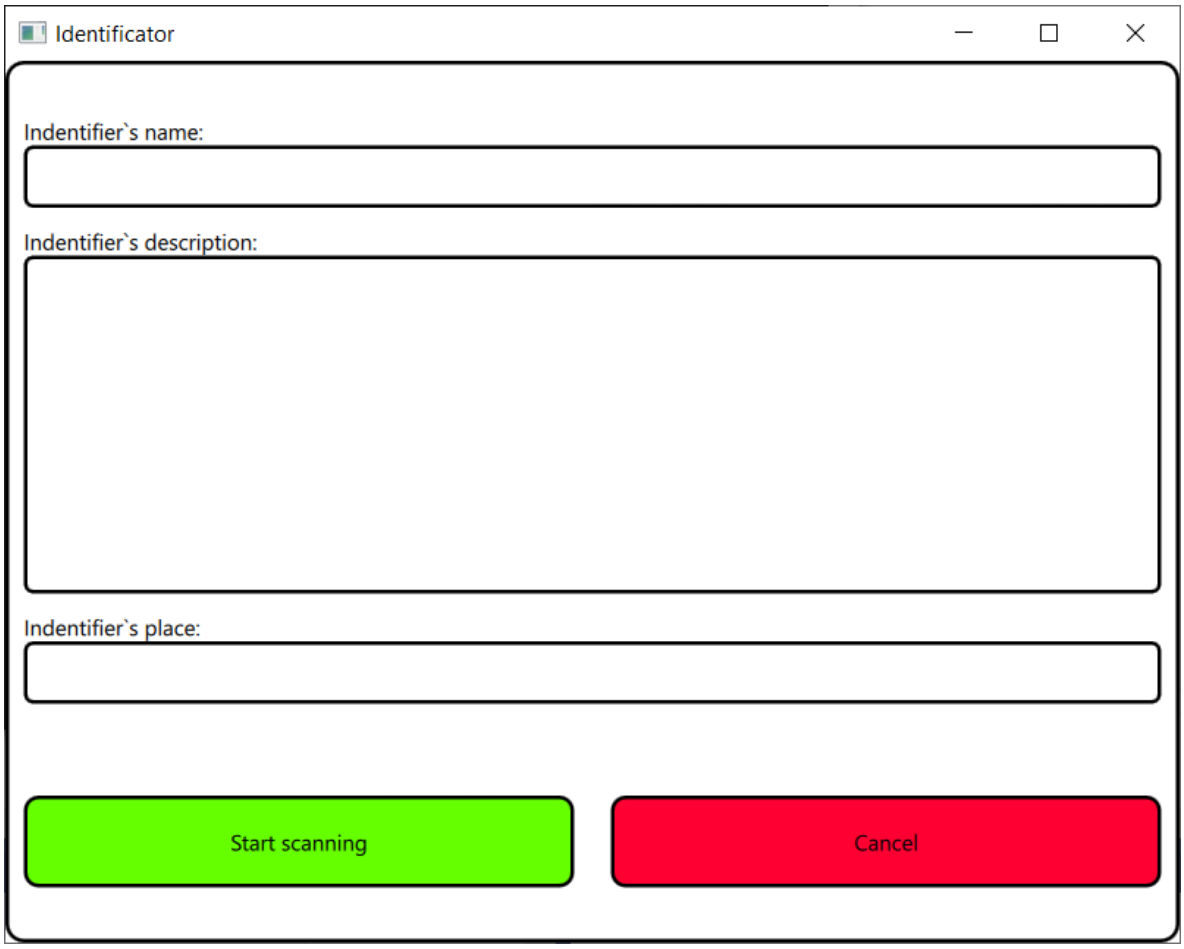

Рисунок 4.1 – Описання ідентифікатора

Розташовуємо всі головні класифікаторскі органи у обраному квадраті і для додавання поточного обличчя на вікні натискаємо на кнопку «Enter» (рисунок 4.2).

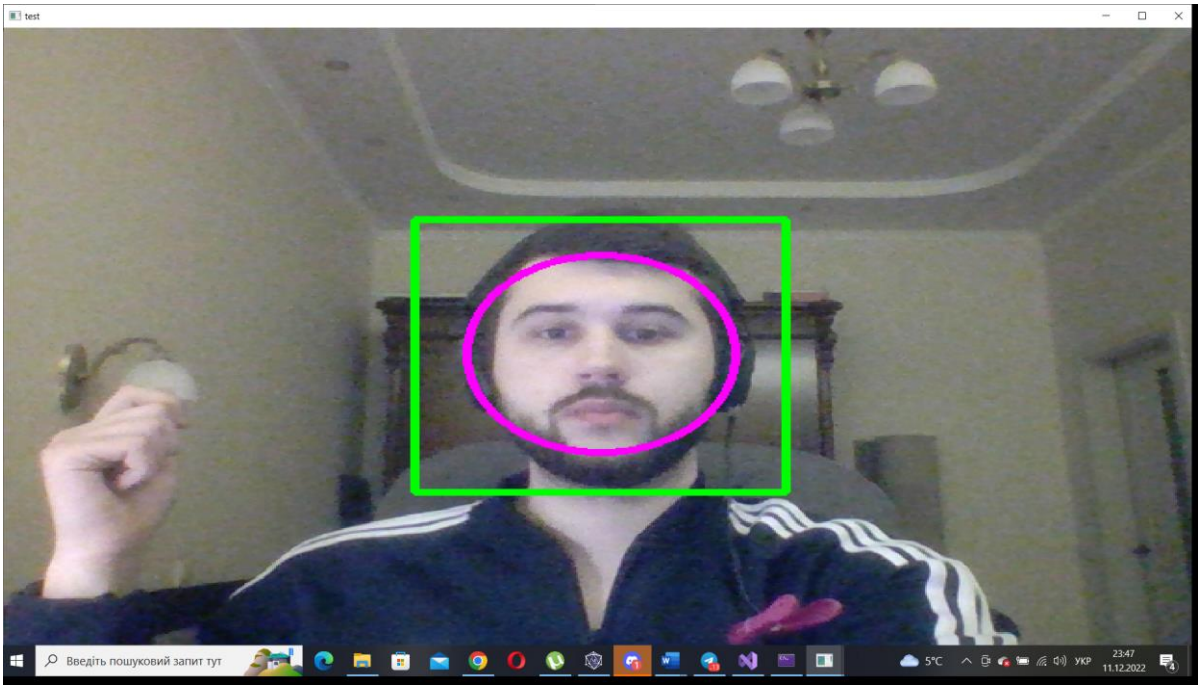

Рисунок 4.2– Додавання обличчя

Вводимо назву доданого ідентифікатора і знаходимо його в базі даних (рисунок 4.3). Також присутня можливість змінити його дані або видалити з системи.

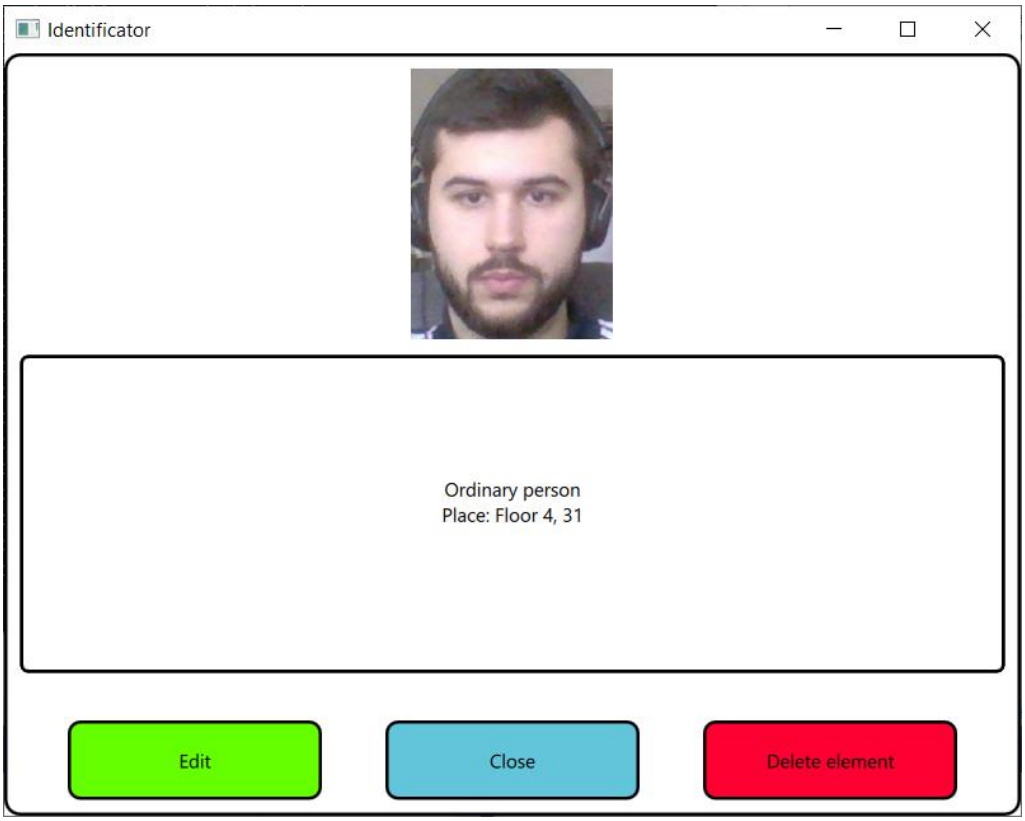

Рисунок 4.3– Вікно з можливими опціями керування та зміни елемента бази даних

4.2 Тестування роботи розпізнаваного модуля та порівняння з немодифікованою версією

Системи оцінювання за двома параметрами продуктивності:

- Коефіцієнт вдалого розпізнання або ККД розпізнавання.

Визначається як відношення вдалих розпізнавань до загальної кількості спроб.

- F-міра.

Вона розраховується на основі точності та запам’ятовування тесту, де точність – це кількість справді позитивних результатів, поділена на кількість усіх позитивних результатів, включаючи ті, які не були визначені правильно, а

відкликання – це кількість справді позитивних результатів, поділена на кількість кількість усіх проб, які мали бути визначені як позитивні. (6).

$$F_{\text{міра}} = 2 * \frac{\text{влучність} * \text{повнота}}{\text{влучність} + \text{повнота}} \quad (9)$$

Далі наведена таблиці з результатами визначень обличчя.

Таблиця 1 – Порівняння алгоритмів з першим набором фотозображень

	Кількість обличь	Кількість визначених обличь	Кількість хибно визначених обличь	Кількість розпізнавань	ККД	F-міра
Алгоритм Haar	30	26	6	30	86,6	83,8
Модифікований алгоритм	30	27	5	30	90	87
Алгоритм Eigenface	30	26	7	30	86,6	82,4
Алгоритм Fisherface	30	25	4	30	83,3	84,7
Алгоритм CNN	30	26	6	30	86,6	83,8
Алгоритм LBP	30	26	7	30	86,6	82,4

Таблиця 2 – Порівняння алгоритмів з другим набором фотозображень

	Кількість обличь	Кількість визначених обличь	Кількість хибно визначених обличь	Кількість розпізнавань	ККД	F-міра
Алгоритм Haar	100	88	21	100	88	84,1
Модифікований алгоритм	100	93	17	100	93	88,5
Алгоритм Eigenface	100	88	20	100	88	84,5
Алгоритм Fisherface	100	87	25	100	87	82
Алгоритм CNN	100	92	18	100	92	87,5
Алгоритм LBP	100	89	20	100	89	85,1

Таблиця 3 – Порівняння алгоритмів з третім набором фотозображень

	Кількість обличь	Кількість визначених обличь	Кількість хибно визначених обличь	Кількість розпізнавань	ККД	F-міра
Алгоритм Haar	300	268	62	300	89,3	85
Модифікований алгоритм	300	284	48	300	94,6	89,7
Алгоритм Eigenface	300	269	64	300	89,6	84,9
Алгоритм Fisherface	300	263	69	300	87,6	84
Алгоритм CNN	300	278	45	300	92,6	89,1
Алгоритм LBP	300	265	68	300	88,3	83,6

Таблиця 4 – Порівняння алгоритмів з четвертим набором фотозображень

	Кількість обличь	Кількість визначених обличь	Кількість хибно визначених обличь	Кількість розпізнавань	ККД	F-міра
Алгоритм Haar	900	805	187	900	89,4	85
Модифікований алгоритм	900	854	139	900	94,8	90,1
Алгоритм Eigenface	900	805	195	900	89,4	84,7
Алгоритм Fisherface	900	794	209	900	88,2	83,4
Алгоритм CNN	900	835	133	900	92,7	89,3
Алгоритм LBP	900	792	202	900	88	83,5

Графік зміни коефіцієнту вдалого розпізнавання за кількістю розпізнавань наведено нижче (рисунок 4.4).

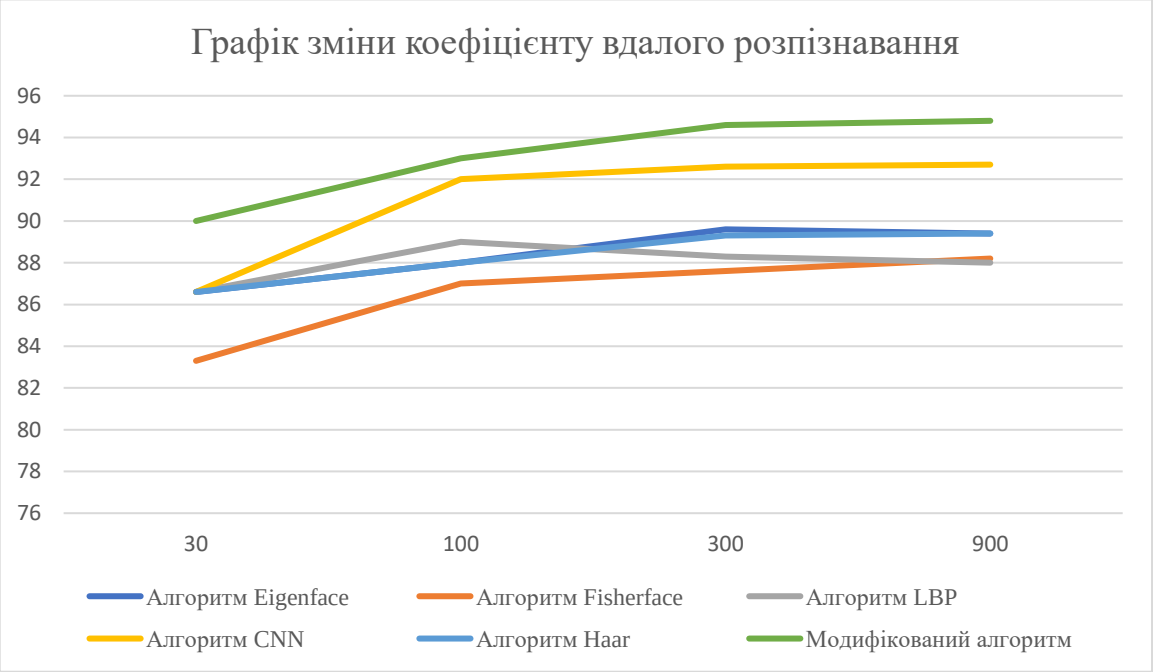

Рисунок 4.4 – Графік зміни коефіцієнту вдалого розпізнавання

Графік зміни F-міри за кількістю розпізнавань наведено нижче (рисунок 4.5).

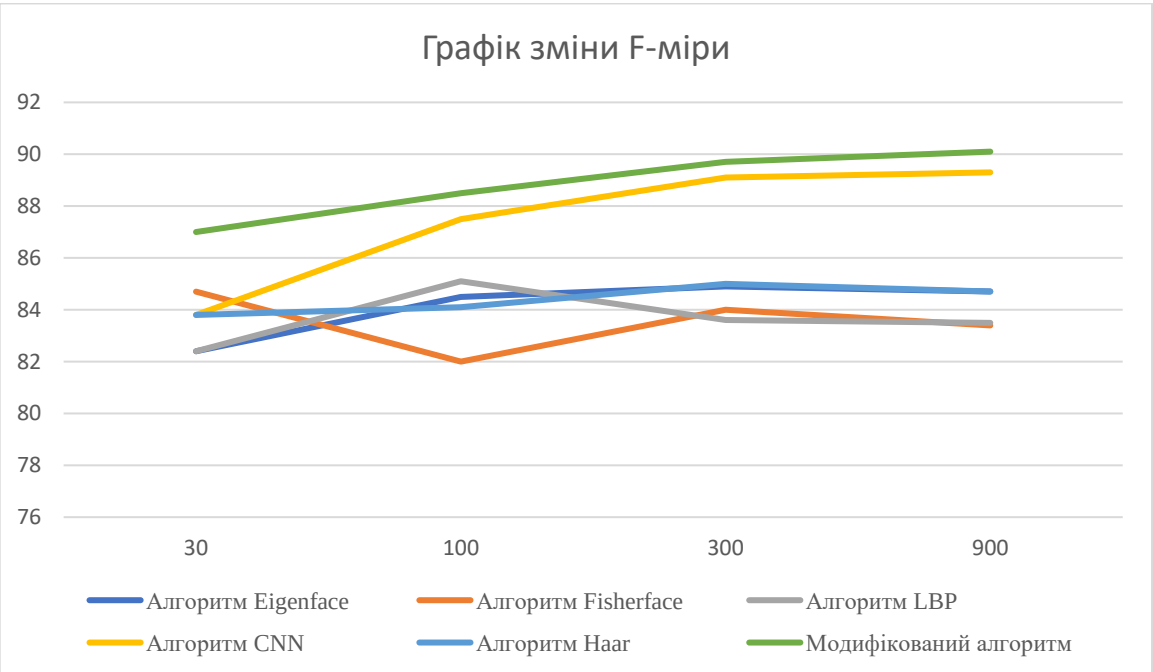

Рисунок 4.5 – Графік зміни F-міри

На обох графіках видно тенденцію приросту на невеликих значеннях кількості використаних фотозображень (до 100 фотозображень), далі для двох алгоритмів (Модифікований та CNN) ми бачимо поступове зростання і майже досягнення значення їхнього екстремуму, інші алгоритми не відзначаються такою стабільністю і не виходять на так зване плато, а мають більш рвану зміну відсоткового значення у ту чи іншу сторони.

4.3 Тестування ідентифікації

Перевірка починається з відкриття UI-інтерфейсу, в якому представлені наявні в системі обличчя (шаблони ідентифікаторів), потім натискаємо на кнопку «Identificate», яка переводить нас в наступне вікно, де відбувається безпосередня ідентифікація (рисунок 4.6).

Рисунок 4.6 – Список наявних особистостей

У вікні ідентифікації підставляємо обличчя в зону, де найкраще проводити знімок, очікуємо на розпізнання обличчя для більшої точності і натискаємо на клавіатурі кнопку «Enter» (рисунок 4.7).

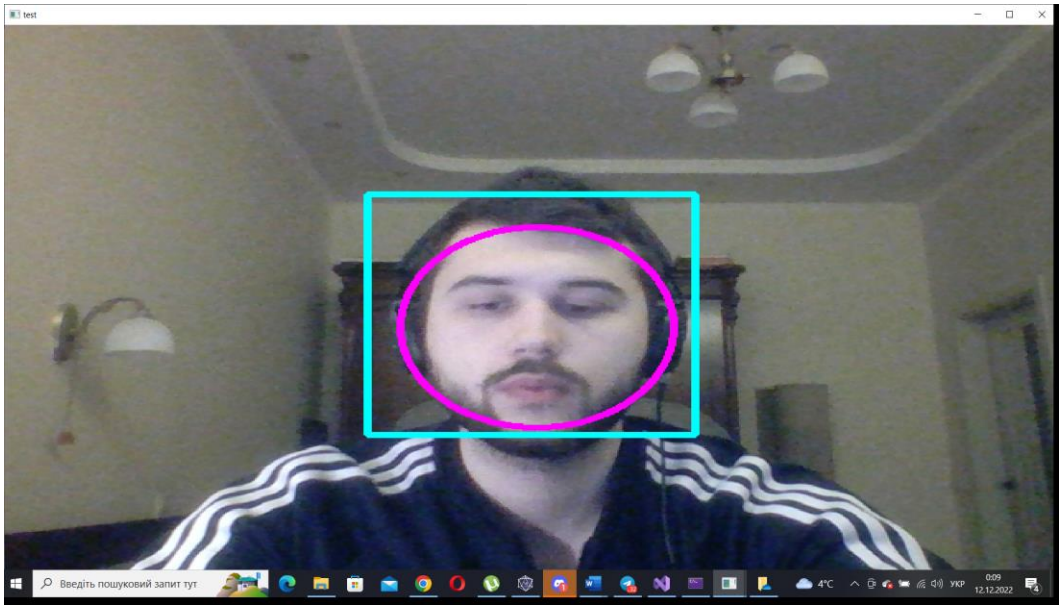

Рисунок 4.7 – Ідентифікація обличчя

Очікуємо деякий час та отримуємо лист з подібними користувачами. Якщо їх більше одного, то адміністратор обирає правильний варіант і візує вхід (рисунок 4.8).

Рисунок 4.8 – Лист подібних користувачів

4.4 Навантаження процесора на різних етапах роботи програми

Заміри навантаження процесора проводилось у трьох станах роботи системи:

- Взаємодія з UI-інтерфейсом (рисунок 4.9);

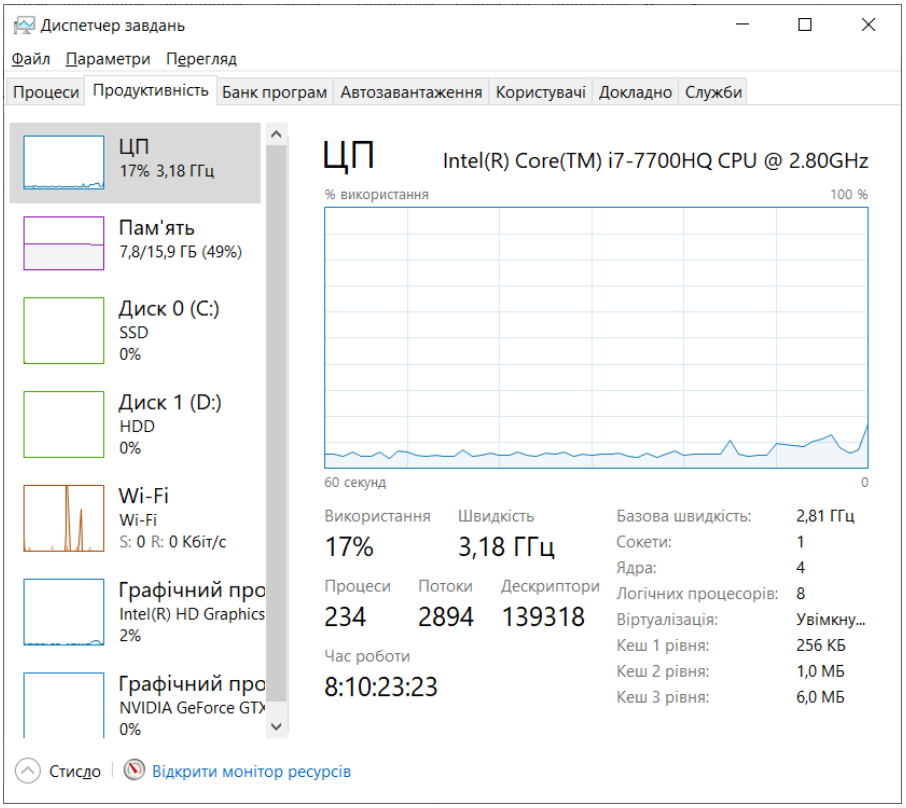

Рисунок 4.9 – Заміри для першого стану

- Додавання обличчя в базу даних (рисунок 4.10);

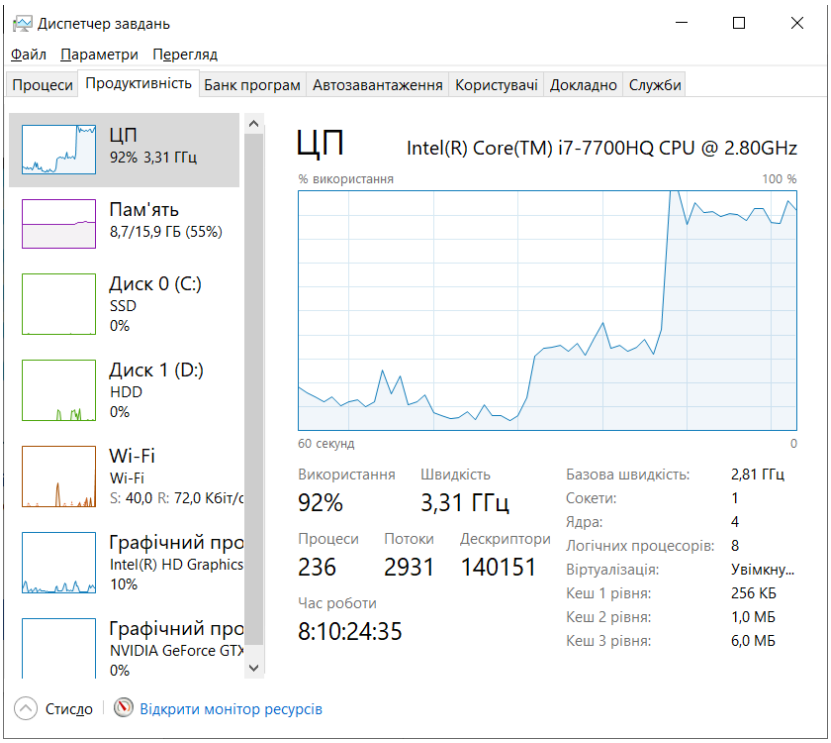

Рисунок 4.10 – Заміри для другого стану

- Ідентифікація обличчя (рисунок 4.11).

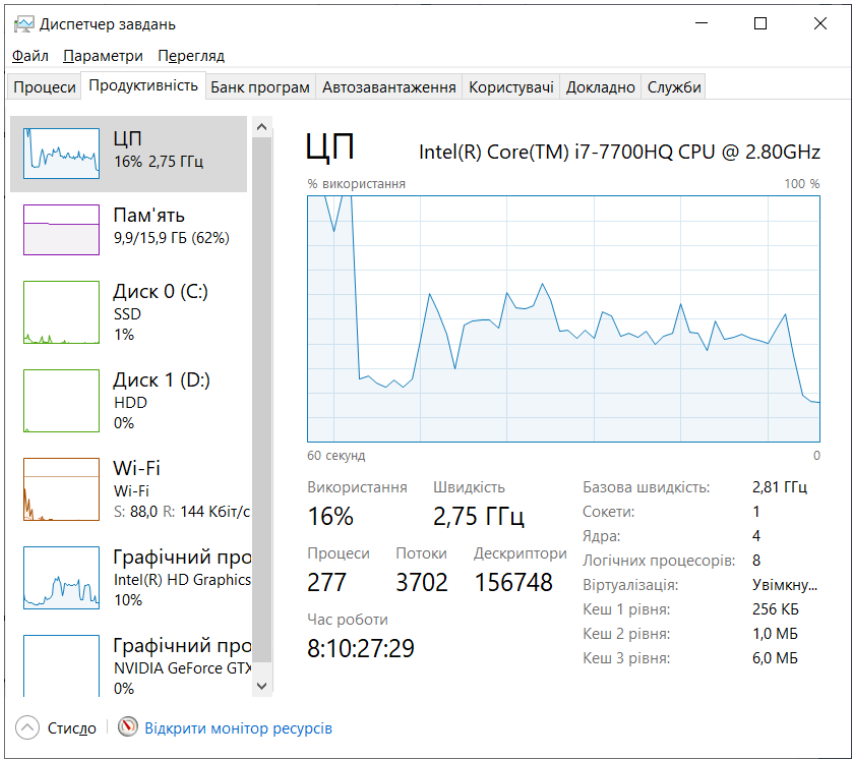

Рисунок 4.11 – Заміри для третього стану

З наявних вище графіків продуктивності можна виділити такі речі:

- навантаження на ЦП зі сторони UI-інтерфейсу є невисоким і в середньому тримається планки у 10%;
- навантаження на ЦП при ідентифікації досягає в пікові моменти 60%, що означає достатність наявних ресурсів комп’ютера (невеликих за сьогоднішніми мірками);
- навантаження на ЦП при створенні ідентифікатора досягає місцями 100%, що означає правильне розподілення наявних ресурсів комп’ютера для максимальної швидкодії.

Висновки до четвертого розділу

Перевірено роботу візуального модуля на правильність спрацювання кожного окремого графічного компоненту. Помилок, у вигляді «рандомних» активацій чи спрацювань виявлено не було. Визначено середній приріст на 5% в якості визначення обличчя модифікованим алгоритмом у порівнянні з його початковою версією. В таблицях та графіках представлено зміну якості визначення обличчя за обома обраними показниками (коефіцієнт вдалого розпізнання та F-міра). Показано навантаження ЦП комп’ютерної системи з наступними характеристиками:

- Процесор Intel Core i7-7700HQ 2.8 GHZ;
- Невідома камера з роздільною здатністю 640x480.
- Оперативна пам’ять DDR4 16 gb;
- Графічний процесор Intel HD Graphics.

ВИСНОВКИ

В магістерській дисертації розроблено модифікований алгоритму Haar та програмний застосунок ідентифікації обличчя.

Проаналізовано існуючі методи розпізнавання та ідентифікації обличчя, визначено параметри привабливості біометрії як надійного ідентифікаційного ключа та розглянуто різні програмні підходи реалізації систем контролю та управління доступом.

Представлена модифікація алгоритму Haar складається з наступних пунктів:

- Аналіз фотозображення за трьома різними кольоровими моделями (RGB, YCbCr та HLS) і поєднання обробок у єдиний результат;
- Заміна поточного алгоритму класифікації на SVM зі зміненою стратегією вибірок для навчання, що аналізує найближчі до кордону точки (перевірка на гетерогенність);
- Застосування методу генерації негативних фотозображень з наявних позитивних.

Створення модифіковано методу розпізнавання обличчя на базі Haar та імплементування його до застосунку дозволило вирішити наступні проблеми:

- Локалізація об’єкту ідентифікування в полі роботи камери;
- Чіткість розпізнавальних частин обличчя;
- Підвищення точності роботи розпізнавального алгоритму в залежності від кількості зображень; найбільший приріст склав 5,1% за більш повним параметром оцінки.

У розділі тестування представлено результати покращення алгоритму та приросту якості його розпізнавання в залежності від параметра вимірювання (значення надані у відсотках):

- Інтервал значень для коефіцієнту вдалого розпізнавання: 3,4 – 5,4;
- Інтервал значень для F-міри: 3,2 – 5,1.

Таблиці та графіки відмічають тенденцію зменшення кількості хибних визначень та планомірне зростання значення якості розпізнавання за обома критеріями оцінки з виходом на максимальне плато. Заміри роботи процесора показують правильний підхід до розподілу навантаження на наявні системні ресурси.

Дисертацію розроблено з використанням засобів, що спираються на мову C++ для забезпечення швидкодії та зменшення кількості інструментів написаних різними програмними мова, що робить розробку більш комплексною та складною.

Структуру програмного застосунку спроектовано таким чином, щоб була можливість для модифікації та вдосконалення наявних компонентів. Стандартизація програмних модулів дозволить стороннім розробником достатньо швидко зрозуміти всю нюанси створеної програми.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Відомості про ідентифікацію [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B0);
2. Відомості про біометрію [Електронний ресурс] – Режим доступу: https://esu.com.ua/search_articles.php?id=35327;
3. Відомості про розпізнавання обличчя [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchenterpriseai/definition/face-detection#:~:text=Face%20detection%20%2D%2D%20also%20called,human%20faces%20in%20digital%20images;
4. Відомості про наявні СКУД [Електронний ресурс] – Режим доступу: https://www.spiceworks.com/it-security/identity-access-management/articles/facial-recognition-software/;
5. Дадиверін В. В. Засоби обробки та аналізу фотозображень на базі Computer Vision: дипломний проект бакалавра. – Київ, 2021. – 92 с;
6. Воронцов К. В., «Лекції методу опорних векторів». 2007. – 16 с;
7. Відомості про HOG [Електронний ресурс] – Режим доступу: https://waksoft.susu.ru/2021/11/01/histogram-of-oriented-gradients/;
8. Дадиверін В. В., Фещенко І.О., Потапова К.Р., Тарасенко-Клятченко О.В. «СИСТЕМА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ З ВИКОРИСТАННЯМ ПОЄДНАННЯ МЕТОДІВ HAAR ТА SVM». – Науковий журнал «Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки», Херсон, 2022. – 12 с;
9. Viola P., Jones M. «Rapid object detection using boosted cascade of simple features» – New Jersey, USA, 2001. – 9 pages;

10. Kai-Bo D., S. Sathiya K. «Which Is the Best Multiclass SVM Method? An Empirical Study» – Multiple Classifier Systems, LNCS. pp.278–285.