НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки

Кафедра Автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ Ролік О. І.
(підпис)(ініціали, прізвище)
«___»_____20__р.

Дипломний проект
на здобуття ступеня бакалавра
зі спеціальності 6.050201 Системна інженерія
на тему: Система оцінки якості відбитків пальців

Виконав: студент 4 курсу, групи ІА-з51
(шифр групи)

Бєляєв Кирило Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник роботи: к.т.н., доцент Репнікова Н.Б. _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) (підпис)

Консультант: _____ _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) (підпис)

Рецензент: доцент каф. АСОіУ, к.т.н., доцент Жданов О.Г. _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент_____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6. 050201 «Системна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.І. Ролік

«___»___2019 р.

ЗАВДАННЯ

на дипломний проект студенту

Бєляєву Кирилу Андрійовичу

1. Тема проекту «Система оцінки якості відбитків пальців», керівник проекту Репнікова Наталія Борисівна, к. т. н., доцент, затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом проекту 14 червня 2019 р.

3. Вихідні дані до проекту

Якість відбитків пальців 5 балів, для зіставлення використання алгоритма Minutia Cylinder-Code

4. Зміст пояснювальної записки

Аналіз існуючих рішень, вибір алгоритмів, опис реалізації розробленої системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Блок-схема повного цикла програми, UML-діаграма прецендентів, діаграма класів, структурна схема системи.

6. Дата видачі завдання 5 квітня 2019 р.

Календарний план

№	Назва етапів виконання дипломного проекту	Термін виконанн етапів проекту	Приміт
1	Порівняльний аналіз існуючих рішен	22.01.18 – 2.02.18	
2	Визначення основних вимог до систе	8.02.18 – 1.03.17	
3	Проектування архітектури системи	1.03.17 – 10.04.17	
4	Розробка програмного коду системи	11.04.17 – 15.05.17	
5	Опис розробленої системи	16.05.17 – 26.05.17	
6	Оформлення пояснювальної записки	1.06.18 – 13.06.18	
7	Подача проекту на перевірку	16.06.18 – 19.06.18	

Студент К. А. Бєляєв

Керівник проекту Н. Б. Репнікова

АНОТАЦІЯ

Бєляєв К. А. «Система оцінки якості відбитків пальців». КПІ ім. Ігоря Сікорського, Київ, 2019.

Дипломний проект присвячений розробці системи, яка використовується для покращення якості вхідних відбитків пальців за допомогою алгоритмів NFIQ (для аналізу карт якості) та Minutia Cylinder-Code (для зіставлення та покращення).

Текстова документація містить інформацію про призначення, проектування, розробку, тестування та особливості роботи із програмною системою. Велика увага була приділена вибору технічних засобів для реалізації та проектуванню, оскільки розроблювана система має зв’язок з технічною частиною, а саме сканерами відбитків пальців.

Ключові слова: архітектура програмного забезпечення, C#, Windows Forms, дактилоскопія, похибка.

Розмір пояснювальної записки – 63 аркуша, містить 27 ілюстрації, 0 таблиць.

SUMMARY

Belyaev K. A. "Fingerprint Quality Score System” Igor Sikorsky’s KPI, Kyiv, 2019.

The graduation project is devoted to the development of a system used to improve the quality of input fingerprints using NFIQ algorithms (for analysis of quality maps) and Minutia Cylinder-Code (for comparison and improvement).

The text documentation contains information on designation, design, development, testing and features of work with the software system. Much attention was paid to the choice of technical means for implementation and design, as the developed system has a connection with the technical part, namely, fingerprint scanners.

Keywords: software architecture, C #, Windows Forms, fingerprinting, error

The size of the explanatory note is 63 sheets, contains 27 illustrations, 0 tables.

Ном	Фо	Позначення	Найменування	Кіль	Ном	При
			1.1.1.1.2 Документація			
			Знову розроблена			
		ЗІА51.020БАК.002.ПЗ	Пояснювальна			
		ЗІА51.020БАК.003.Э1	Структурна схема			
		ЗІА51.020БАК.004.Д1	Діаграма класів			
		ЗІА51.020БАК.005.Д2	Діаграма прецендентів			
		ЗІА51.020БАК.006.Д3	Діаграма циклу			

		№	І		1.1.1.1.1 ЗІА51.020БАК.001.ТП		
		Бєляєв			Система оцінки якості відбитків пальців	Л	Лист
		Репнік					6
		Н.				КПІ ім. Ігоря	
					Відомість технічного		

Пояснювальна записка

до дипломного проекту

на тему: «Система оцінки якості відбитків пальців»

Київ – 2019 рік

ЗМІСТ

ВСТУП

В наш час ідентифікація за допомогою відбитків пальців є найбільш поширеним біометричним способом визначення особистості. З плином часу відбитки пальців все частіше використовуються в криміналістиці, на кордонах держав, на пропускних терміналах аеропортів, вокзалів тощо. Завдяки своїй унікальності і незмінності, такий підхід привернув багато уваги серед спеціалістів криптографії, дактилоскопії та контр–розвідувальних структур.

Однак завжди існує можливість отримати неякісне зображення відбитка. Причин досить багато: надто жирна або суха шкіра рук, великі пори або випадковий рух пальця при знятті відбитка.

Рисунок 1 – Приклади різних відбитків пальців

Таким чином, після розпізнавання двох однакових відбитків однієї людини, зіставлення може виявитися, що вони не однакові. Або ж навпаки такий відбиток може збігтися з відбитком іншої людини. Також використання неякісних зображень відбитків знижує прохідну здатність біометричних систем.

Для того, щоб заздалегідь уникнути подібних проблем, були розроблені алгоритми визначення якості зображення відбитку пальця. Перед тим, як використовувати відбиток для ідентифікації особистості, перевіряється його якість.

Наприклад, в алгоритмі NFIQ (NIST Fingerprint Image Quality) кожне зображення відбитка пальця відповідає значенням від

1 (відмінну якість) до 5 (погана якість). Також існує більш поглибленний алгоритм LFIQ (Latent Fingerprint Image Quality)[1], який спеціалізується на визначенні якості відбитків, знайдених на місці скоєння злочину.

На данний час існують такі підходи до розпізнавання відбитків пальців: на основі глобальних ознак і по локальним (місцевим) ознакам.

Глобальні ознаки - це зовнішній вигляд відбитка, орієнтація зображення, кривизна і поле напрямків, що описує загальний стан формування папілярних ліній відбитків пальців.

Локальні ознаки або Мінуції (від англ. Minutia) – це місцеві особливості папілярних ліній, унікальні для кожного відбитку точки по площі всього зображення. Всього існує 155 різновидів Мінуцій, однак найбільш відомими вважаються два види Мінуцій, в яких обриваються або роздвоюються папілярні лінії (рисунок 2). Надалі в роботі буде розглядатися метод локальних ознак, так як він є найбільш поширеним в наш час і є більш надійнішим, ніж розробка моделей глобальних ознак.

Рисунок 2 – Обривання та роздвоєння папілярних ліній.

Для порівняння декількох відбитків по Мінуціям в роботі буде використаний алгоритм MCC (Minutia Cylinder-Code) [1], як найбільш точний і швидкий з практичної точки зору. Для кожної Мінуції створюється циліндр, в якому враховуються напрямок і розташування найближчих сусідів мінуції.

Щоб оцінити стійкість дактилоскопічних систем, зазвичай використовують ймовірності помилок 1-го і 2-го порядку. В даному випадку це помилковий збіг і помилкова розбіжність відбитків пальців.

Для них визначено ймовірність помилкового збігу (False Match Rate, FMR) і ймовірність помилкової розбіжності відбитків (False Non-Match Rate, FNMR). У загальному випадку алгоритм, який використовується для порівняння, приймає рішення на підставі деякого порогового значення t. У зв'язку з цим важливо пам'ятати, що помилки FMR і FNMR залежать прямо пропорційно один від одного, та при підвищенні чутливості системи обробки, тобто при збільшенні порога(відповідно, при зниженні FMR), також підвищується FNMR. Для результативного уявлення даних ймовірностей використовують ROC-криві (Receiver Operating Characteristic - операційну характеристику пристрою одержувача) або DET-криві (Detection-Error Tradeoff, компроміс помилок виявлення). ROC-криві являють собою залежність FMR (t) від 1-FNMR(t) при постійной плаваючій точці порогового значення, а DET-криві - залежність FMR (t) від FNMR (t).

Також іноді використовують коефіцієнт порівняння ймовірності помилок 1-го і 2-го роду (Equal Error Rate, EER, рівень рівної помилки), які позначають точку зіставлення помилок FMR і FNMR на одній площині. Чим нижче рівень EER, тим біометрична система вважається стійкішою. Рівна частота помилок може також згадуватися як частота кросовера (CER, частота помилок кросовера).

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Розглянемо автоматизовану технологію в дактилоскопічних сенсорах – мультиспектральну [8]. Новітня система коригування якості за допомогою апаратної частини аналізатора.

Ринковий сегмент дактилоскопічних сканерів, що застосовуються в біометричних сканерах для СКУД, відомий, швидше за все, тільки вузькому колу фахівців.

В області систем біометричної ідентифікації в системах контролю і управління доступом (СКУД) найбільшого поширення мають дактилоскопічні сканери. Цьому є ряд пояснень - популярність самого методу ідентифікації особистості по папілярних лініям пальців ще з часу виникнення класичної криміналістики, простота отримання даних, надійність, стабільність даного біометричної ознаки протягом усього життя людини.

Найважливішим елементом будь-якого дактилоскопічного сканера, що використовує в якості біометричної ознаки відбиток пальця, вважається спеціалізований елемент зчитування. Значною мірою саме від сканера залежить стабільність ідентифікації, здатність розпізнати підробку (муляж), можливість роботи в несприятливих зовнішніх умовах.

Оптичні контактні сканери. Існуючі в даний час дактилоскопічні сканери використовують різні методи отримання зображення відбитка пальця, які можна розділити на декілька основних типів: оптичний; ємнісний; радіочастотний; ультразвукової; теплової.

Оптичні контактні сканери. Один з найбільш поширених типів, які забезпечують сканування відбитків пальців мініатюрними камерами на ПЗС, або КМОП-чіпи. При скануванні палець повинен бути прикладений до скла, під яким розташовуються джерело світла і, власне кажучи, сама реєструє камера. До оптичних також відносяться оптоволоконні і електрооптичні сканери, але для вітчизняного ринку це швидше екзотика.

Кремнієві ємнісні сканери. Другий за поширеністю тип. Фактично це масив конденсаторів, що змінюють свою ємність при контакті пальця з сенсором. Ємність конденсатора в точці дотику папілярної лінії і ємність між двома лініями буде різна. Цей метод дозволяє оцифрувати відбиток.

Решта типів сканерів за обсягом ринку набагато відстають від розглянутих вище, тому ми не будемо на них зупинятися, а звернемося до цікавих рішень в найбільшому сегменті оптичних сканерів

Незважаючи на те що різні методи сканування відбитку пальця мають різну чутливість, для більшості характерний загальний недолік - часте виникнення низькоякісних зображень. Що пов'язано або з фізичними характеристиками самого пальця (нечіткі або стерті папілярні лінії, мікропошкодження, суха шкіра, мокрий палець), або з зовнішніми умовами (поганий контакт пальця з сенсором, засвітка) та ін.

FTIR-сканери. У більшості сучасних оптичних дактилоскопічних сенсорів використовується FTIR-ефект (Frustrated Total Internal Reflection), або ефект "порушення повного внутрішнього відображення". Палець прикладається до скла (призми) і підсвічується джерелом обрахунку з одного боку; з іншого боку розташовується камера, яка реєструє відбите світло. Різний характер відбиття світла в місці прикладання пальця дозволяє сформувати зображення відбитка пальця, зареєстроване камерою у вигляді контрастних світлих і темних ліній.

Считыватель SUPREMA с FTIR-сенсором

Считыватель SUPREMA
с двумя FTIR-сенсорами

Рисунок 3 – Новітні прилади з FTIR-сенсором [8]

Принцип роботи. Відбитий від скла з доданим пальцем світло реєструється камерою. У тому місці, де скла торкнувся гребінь папілярної лінії, відбувається порушення ефекту повного внутрішнього відображення (TIR), і світло частково розсіюється. В камеру надходить менший потік світла, і вона реєструє це як темну область.

При потраплянні світла на "впадину" розсіювання не відбувається, оскільки FTIR-ефект зберігається, і камера реєструє світлу область.

В результаті гребені на знімку виглядають як темні лінії, а западини - як світлі. В таких сканерах зазвичай використовують квазімонохроматичне джерело світла.

Проблеми данної технології: Якісний відбиток, отриманий в оптимальних зовнішніх умовах (природне зволоження, відсутність засвічення), буде максимально контрастним, і комп'ютеру не доведеться обробляти його додатково. Однак реальні умови на об'єкті далекі від ідеальних, що призводить до появи низькоконтрастних відбитків, що мають сірі області. Надалі буде необхідно інтерпретувати ці сірі області як чорні або білі, що знижує швидкість роботи біометричного сканера і підвищує ймовірність помилки при порівнянні отриманого відбитка з раніше введеним. В результаті це спричиняє складність реєстрації стертих, пошкоджених

пальців (промислові підприємства), мокрих пальців і впливу засвічення (зовнішня установка сканера).

Розпізнавання підробок. Успішна біометрична система повинна надійно ідентифікувати відбитки живих людських пальців і відхилити всі інші. Якісний сканер забезпечує "живе виявлення", тобто може відрізнити відбиток пальця, вироблений живим людським пальцем, від відбитка, виробленого будь-яким іншим матеріалом (муляжем).

Існує безліч методів підробки - від найпростішого дихання на сенсор, щоб проявився раніше зроблений відбиток від використання графіту і липкої стрічки, застосування латексних, силіконових або желатинових емуляторів до таких зловісних методів, як використання мертвого пальця.

Считыватель L1 с MSI-сенсором

Рисунок 4 – Сканер, захищений від підробок різної складності. [8]

Звичайні сенсори відбитка пальця дозволяють отримувати його образи (темплейти), засновані на відмінності між повітрям (западини) і матеріалом в контакті з датчиком (гребені папілярних ліній). Незважаючи на те, що різні технології розрізняються за методом отримання образу, кожна використовує зазвичай тільки єдину характеристику матеріалу, що знаходиться в контакті з датчиком.

Оптичні сенсори відбитка пальця використовують відмінність в коефіцієнті заломлення, кремнієві сенсори покладаються на відмінність в імпедансі, а теплові датчики - на відмінність в теплової провідності Будь штучний матеріал, який має ту ж саму характеристику, як у реального зареєстрованого пальця, може застосовуватися для виготовлення муляжу. Наприклад, оптичний сенсор може прийняти зображення від тривимірного відбитка пальця, зробленого з пружного матеріалу типу латексу, силікону або желатину.

В цьому і полягає головна слабкість звичайних сенсорів - їх орієнтація на єдину особливість (характеристику) поверхні матеріалу, що містить відбиток пальця. Як тільки будь-який матеріал, який копіює таку поверхневу особливість, виявлений, це можна використовувати для виготовлення муляжу. Більш надійний метод використовує не одну, а кілька вимірюваних характеристик поверхневого і внутрішнього шару шкіри.

Мультиспектральна технологія: Для підвищення стійкості процесу сканування та захисту від обману, на ринку СКУД була представлена мультиспектральна технологія. Завдяки їй в нових дактилоскопічних сенсорах велике значення має не тільки зовнішній, а й внутрішній шар шкіри.

Щоб зрозуміти значення внутрішнього (підшарового) шару шкіри для мультиспектральної технології, треба розуміти, як створений відбиток пальця. Гребені папілярних ліній відбитка, які ми бачимо на поверхні шкіри, мають приховану основу, у вигляді судин і інших підшкірних структур. Фактично видимі папілярні лінії на кінчиках наших пальців - це просто "відлуння" фундаментального "внутрішнього відбитка пальця". На відміну від поверхневих особливостей відбитка пальця, які можуть бути змінені вологістю, брудом або частково стерті, "внутрішній відбиток пальця" стабільніший і незмінний. Об'єднання цих двох характеристик забезпечує новому методу високу надійність і стійкість.

Дактилоскопічні сенсори на основі мультиспектральної технології формування зображення MSI (Multispectral Imaging) здатні отримувати інформацію не тільки при поверхневому, а й при підповерхневому шару шкіри. Сенсори MSI забезпечують отримання ряду знімків пальця при різних умовах освітлення, що включають в себе різні довжини хвиль, положення джерела світла, умови поляризації.

Різні довжини хвилі видимого світла взаємодіють зі шкірою по-різному, дозволяючи значно збільшити обсяг даних В результаті отримані знімки містять інформацію не тільки про поверхневі, а й про внутрішні (підповерхневі) особливості шкіри.

Мультиспектральна технологія використовує зв'язок між зовнішніми і внутрішніми образами відбитка пальця; дані внутрішнього образу, зібрані за допомогою MSI-сенсора, доповнюють дані зовнішнього відбитка пальця, дозволяючи отримати образ (темплейт) відбитка пальця екстра-класу.

Метод проявлення зображення: Мультиспектральний сенсор має кілька джерел світла (LED-світлодіоди) і використовує для отримання остаточного деталізованого відбитка пальця одночасно поляризоване і неполяризоване світло. На рис.5 темні стрілки ілюструють освітлення пальця прямим поляризованим світлом.

Світло від першого LED-світлодіода проходить через лінійний поляризаційний фільтр і потрапляє на палець, прикладений до поверхні сенсора. Після взаємодії з пальцем відбите світло надходить до блоку формування зображення через другий поляризаційний фільтр. Оптична вісь другого фільтра ортогональна фільтру джерела світла, що зменшує вплив світла, відбитого від поверхні шкіри, і підкреслює світло, який зазнавав розсіюванню в більш глибоких шарах.

Другий LED-світлодіод (його на рис.5 не відображали) висвітлює палець безпосередньо, без використання проміжного поляризованного фільтра. В цьому випадку відбите світло від поверхні і від внутрішніх

структур пальця світло потрапляє на блок реєстрації зображення одночасно. Отримане зображення несе набагато більше інформації саме про поверхневому шарі.

Важливо, що обидва джерела освітлення (і поляризоване, і неполяризоване) і блок реєстрації і зображення влаштовані таким чином, щоб уникнути будь-яких критичних кутових відображень в точці дотику пальця на склі сенсора. Таким чином, кожен LED-світлодіод висвітлює палець, а блок реєстрації зображень отримує стабільне зображення незалежно від того, суха чи шкіра, брудна і навіть наскільки якісний контакт пальця з поверхнею сенсора. Ця особливість - ключовий аспект надійності технології MSI сенсора.

Для підвищення якості отриманого темплейта і розширення обсягу даних при створенні дактилоскопічних сенсорів сучасні розробники використовують обидві технології - і FTIR, і MSI

Отримання остаточного темплейта. Темплейт, фактично отримується в дактилоскопічних сенсорі на базі технологій MSI і FTIR, є результатом "збірки" дев'яти зображень, вісім з яких отримані з використанням прямого поляризованого і неполяризованого освітлення, а одне - на основі FTIR-технології.

Вбудований процесор зчитувача збирає з дев'яти базових зображень одне якісне - остаточне. Якість підсумкового зображення MSI-сенсора набагато вище одержуваного в класичних FTIR-сканерах.

Захист від підроблених пальців. Мультиспектральні сенсори дозволяють захиститися від муляжу, визначаючи справжність пред'явленого пальця. Вони також мають можливість відрізнити - живий чи мертвий палець використаний. Базові образи, отримані при різній поляризації і кутах освітлення, дозволяють сенсора MSI реєструвати багато властивостей.

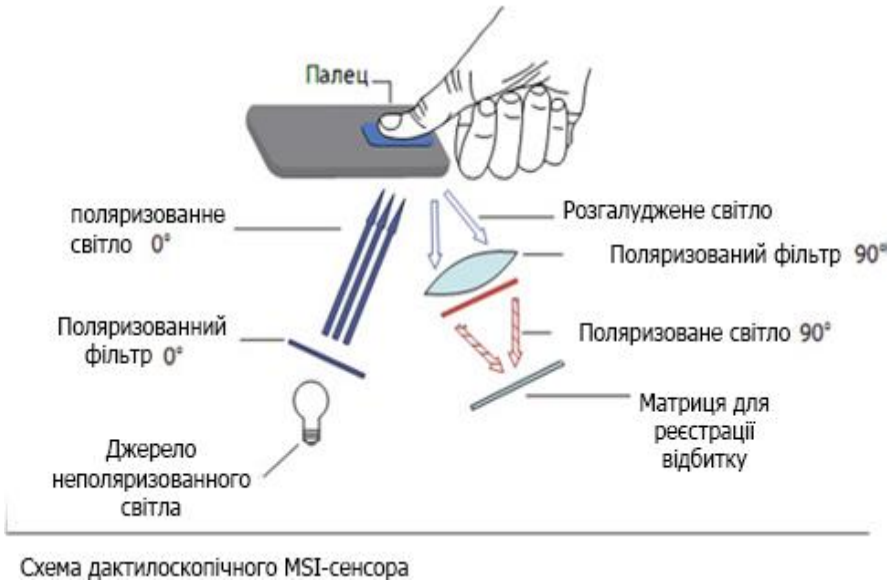


Рисунок 5 – Схема мультиспектрального MSI-сенсора. [8]

Спектральні особливості - це унікальні параметри кольору поверхні матеріалу, розміщеного на сенсорі. Діапазон кольору шкіри живого пальця досить вузький і сильно відрізняється від більшості матеріалів, які можуть застосовуватися для муляжів. Мультиспектральний сенсор використовує ці особливості для визначення підробки. Враховуються і більш складні спектральні характеристики.

Просторові характеристики також надзвичайно важливі для виявлення підробки. Оскільки MSI-сенсор використовує не тільки інформацію про поверхневий шар шкіри, але і про внутрішній, він здатний детектувати наявність муляжу, зіставляючи ці дані між собою. Так як зовнішній і внутрішній шари повинні відповідати один одному, поява відмінностей говорить про спробу емулювати палець за допомогою будь-якого органічного або неорганічного матеріалу.

Робота в надскладних умовах: Якщо класичні FTIR-сенсори і дактилоскопічні зчитувачі на їх основі обмежені діапазоном фізичних умов експлуатації та категорією користувачів (вимога природного зволоження пальця, стертість відбитків пальців при інтенсивному фізичній праці), то мультиспектральні сенсори таких обмежень не мають.

Поганий контакт пальця з сенсором або потрапляння на палець або на сенсор води призводять до неможливості зняти відбиток, використовуючи FTIR-технологію. Для FTIR-сенсора буде складним прочитати палець людини, яка прийшла з морозу, - палець занадто сухий. Інша проблема класичних сенсорів - засвітлення прямим сонячним світлом. Для MSI-сенсора такі проблеми – це нічого.

Перераховані проблеми призводять до того, що класичні FTIR-сенсори небажано застосовувати в дактилоскопічних зчитувачах СКУД для зовнішньої установки. Вони оптимальні за критерієм ціна / якість при використанні в приміщеннях на невиробничих об'єктах. Мультиспектральні сенсори, навпаки, відповідають найжорсткішим вимогам до дактилоскопічний технологіям, які можуть застосовуватися в СКУД. Вони ефективні, надійні і можуть експлуатуватися в широкому діапазоні зовнішніх умов, в тому числі і поза приміщеннями.

Тому, згідно з представленними рішеннями, об’єктом використання було взято мультиспектральний MSI-фільтр бо він має найстійкіші характеристики сканування та легко програмується на покращення якості відбитків.

2 ОГЛЯД ТЕХНОЛОГІЇ

В данному проекті для системи якісної оцінки відбитків пальців була використана мова програмування C#. Було реалізовано 4 підпроекта та підключено бібліотеку візуального оформлення – Nvidia CUDA, технологія – CUDA graphics for fingerptints. До складу підпроектів входить: NFIQ-аналізатор, MCC-компаратор[3], бібліотеки графічного забезпечення та користувацька форма взаємодії між іншими проектами.

2.1 Постанова задачі

Метою даної роботи є підвищення якості відбитків пальців та забезпечення стійкості якісних областей відбитків на основі локальних ознак. Для досягнення даної мети були задекларовані наступні сутності:

– отримання карт якості відбитків за алгоритмом NFIQ;

– вибір критеріїв визначення якісних областей відбитків;

– проведення ряду зіставлень відбитків пальців для різних вибірок мінуцій згідно їх якості;

– обчислення типових помилок FMR, FNMR, EER і порівняння результатів зіставлень.

Також безпосередньо використані методології, які описують актуальність технології, зручність використання користувачем та адміністратором, рентабельність фізичного обладання з використанням данної технології, відсутність помилок кода,грамотні use-case та декларування за допомогою UML.

2.2 Огляд існуючих алгоритмів

Відбитки пальців, як унікальні і незмінні характеристики людини, широко використовуються для ідентифікації біометричної особи. Виходячи з факту недосконалості сканерів, для алгоритмів розпізнавання і зіставлення відбитків важлива якість зображень, наприклад, відсутність шуму. Під шумом розуміються жирові області відбитка, області без вираженого спрямування папілярних ліній тощо, за допомогою яких неможливо отримати характеристичну інформацію про відбиток.

Причини отримання неякісних зображень відбитків можуть бути різні: надто жирна або суха шкіра рук, великі пори або випадкове рух пальця при знятті відбитка. Неякісні зображення відбитків пальців як правило

характеризуються відсутністю виражених напрямів папілярних ліній і низькою контрастністю або зовсім відсутністю таких.

Серед розмаїття існуючих підходів для розпізнавання відбитків пальців можна виділити декілька, які найбільш часто застосовуються:

- кореляційне порівняння;
- порівняння по особливих точках;
- порівняння по візерунку;
- зіставлення по шаблону;
- порівняння на основі графів.

Суть методу кореляційного порівняння полягає в тому, що отриманий відбиток пальця накладається на кожен еталон з бази даних по черзі, після чого по пікселях здійснюється розрахунок різниці між ними. Правда процес порівняння повинен включати в себе безліч ітерацій, на кожній з яких зображення повертається під невеликим кутом або ледь-ледь зміщується. Таким чином, даний метод є надзвичайно повільним і потребує високої обчислювальної потужності.

При порівнянні по особливих точках формується шаблон, на якому виділені кінцеві точки та точки розгалуження. На відсканованому зображенні відбитка також виділяються особливі точки, які порівнюються з шаблонними. Головною перевагою даного алгоритму є швидкість його роботи і простота реалізації. До недоліків слід віднести високі вимоги до якості зображення і розмірів чутливого датчика.

В алгоритмі порівняння по візерунку використовується особливості будови папілярного узору. Отримане зображення, розбивається на безліч дрібних осередків, в кожному з яких розташування ліній описується параметрами синусоїдальної хвилі. Отриманий для порівняння відбиток вирівнюється і приводиться до того ж виду, що й шаблон. Головними плюсами розглянутого алгоритму є досить висока швидкість і низькі вимоги до якості зображення. Проте, даний метод дуже складний для реалізації.

В алгоритмі зіставлення по шаблону до уваги беруться не тільки окремо взяті точки, але і загальні характеристики відбитка пальця, такі як товщина смуг, їх кривизна або щільність. Достоїнствами даного методу є те, що він може працювати з відбитком гіршої якості. Проте, даний метод не пристосований для безлічі пошуків в базі даних.

В алгоритмі порівняння на основі графів вихідне зображення відбитка перетворюється в зображення поля орієнтації папілярних ліній, на якому відмічено області з однаковою орієнтацією ліній. Потім визначаються центри цих областей і виходить граф. Подальші дії аналогічні методу порівняння в особливих точках.

На даний момент існують два основних алгоритму визначення якості відбитків: NFIQ і LFIQ. Останній є досить вузькоспеціалізованим, так як використовується для відбитків, знайдених на місці скоєння злочину. Тому в даному проекті буде розглянуто алгоритм NFIQ.

2.3 Алгоритми покращення якості зображення

Поліпшення якості зображення відбитку пальця за допомогою використання фільтра Габора [6].

Фільтр Габора — лінійний фільтр, імпульсна перехідна характеристика якого представляється у вигляді добутку функції Гаусса на гармонійну функцію:

$$g(x,y) = s(x,y) \cdot w_r(x,y),$$

де — s(x,y) комплексна синусоїда, а — $w_r(x,y)$ огинає Гаусса для двовимірного простору. Застосування фільтру Габора для зображень здійснюється за п’ять етапів.

Етап 1. Нормалізація зображення.

Нормалізація зображення необхідна для того, щоб поставити попередні середні значення і відхилення. Нормалізоване зображення G визначається як

зображення, де G (i,j) — значення нормалізованої яскравості пікселя з координатами (i,j). Нормалізоване зображення розраховується виходячи із середнього і середньоквадратичного відхилення вихідного зображення:

$$G(i,j)=\begin{cases} M_0+\sqrt{\dfrac{VAR_0\cdot(I(i,j)-M)^2}{VAR}}, если I(i,j)>M \\ M_0-\sqrt{\dfrac{VAR_0\cdot(I(i,j)-M)^2}{VAR}}, иначе \end{cases}$$

де M_0 і VAR_0 — задані значення середнього і середньоквадратичного відхилення відповідно, і — вихідні значення середнього і середньоквадратичного відхилення, обчислюються за формулами:

$$M=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}I(i,j)\,,\quad VAR=\frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(I(i,j)-M)^2\,.$$

На рисунку 6 наведено приклад вхідного і нормалізованого зображення.

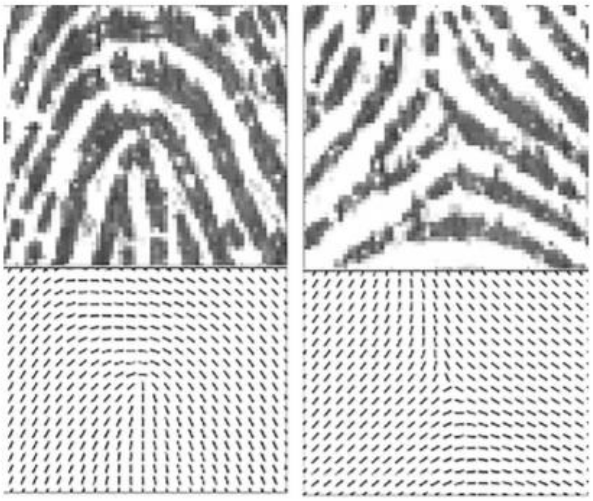
(а) (б)

Рисунок 6 – Приклад вхідного (а) та нормалізованого (б) відбитків одного пальця.

Етап 2. Розрахунок орієнтаційного зображення.

Орієнтаційне зображення O представляє собою матрицю N×N , в якій кожна компонента O(i, j) показує локальну орієнтацію (кут нахилу в даній точці) лінії з координатами (i, j).

$$O(i,j)=\frac{1}{2}\arctan\left(\frac{d_x^2(i,j)d_y^2(i,j)}{2d_x(i,j)d_y(i,j)}\right)$$



а) *б)*

Рисунок 7 – Орієнтаційні зображення. На рис. а) зображений фрагмент центру, на рис. б) дельти.

Етап 3. Розрахунок частотного зображення.

Частотне зображення являє собою матрицю розміру N×N, в якій кожна компонента F(i,j) показує локальну частоту ліній в даній точці, яка визначається як частота гребенів, спрямованих уздовж орієнтації виступу. У разі, якщо в якійсь точці неможливо визначити чітку синусоїдально-окреслену хвилю (наприклад, через наявність особливих точок в цих координатах), частота визначається як середня величина частоти в сусідніх блоках.

Рисунок 8 - Блоки, в яких неможливо визначити чітку синусоидально-окреслену хвилю.

Рисунок 9 - Хвильове представлення ліній в осередку

Розрахунок частоти в точці з координатою (i, j) розраховується наступним чином: якщо λ — кількість пікселів між двома сусідніми гребенями в блоці розмірністю W×W, і центр блоку — піксель з координатами (i,j) , то частота в даній точці буде розраховуватися як F(i, j) = $\frac{1}{\lambda}$

Етап 4. Бінаризація зображення.

Зображення B визначається як зображення, якщо кожен піксель приймає одне з двох можливих значень — нуля одиниці. Одиниця відповідає гребеню відбитка, нуль — западині:

$$R(i,j)=\begin{cases}1, & \textit{если } G(i,j) > B_0\\ 0, & \textit{иначе}\end{cases}$$

де B_0 — поріг маскування, а G(i, j) — інтенсивність пікселя зображення.

Етап 5. Застосування до бінарного зображення фільтрів Габора. Фільтр налаштовується на локальну орієнтацію виступів, застосовується до пікселів виступів і западин зображення.

$$H(i,j:\phi,f)=\exp\left\{-\frac{1}{2}\left[\frac{x_\phi^2}{\delta_x^2}+\frac{y_\phi^2}{\delta_y^2}\right]\right\}\cos(2\pi f x_\phi),$$

Де $x_\varphi^2 = i\cdot\cos\varphi + j\cdot\sin\varphi$; $y_\varphi^2 = -i\cdot\cos\varphi + j\cdot\sin\varphi$; де φ — Орієнтація фільтра Габора, f — частота, а δ_x й δ_y — просторові константи обвідної Гаусса уздовж осей x і y відповідно. Для використання фільтра Габора нам необхідно знати значення наступних величин:

- направлення фільтра;
- частоту синусоїдальної площинний хвилі;
- δ_x і δ_y — середньоквадратичні відхилення обвідної Габора.

Частотна характеристика фільтра визначається з локальної частоти f виступів, напрямок визначається локальної орієнтацією. Значення δ_x і δ_y можна задати при реалізації алгоритму. Чим більше будуть ці значення, тим більше фільтр буде стійкий до шумів, але, в той же час, буде вносити більше спотворень, створюючи неіснуючі виступи і западини. Якщо вибрати значення δ_x і δ_y низькими, фільтр не вноситиме спотворень, але його здатність фільтрувати значно знизиться, що призведе до неефективного усунення шумів. Тому при підборі значень δ_x і δ_y намагаються знайти компроміс між ефективністю фільтра і відсутністю внесених фільтром спотворень. Як правило ці параметри підбираються емпіричним шляхом.

Рисунок 10 – Приклад нормалізації за критерієм Габора.

Вибір параметрів алгоритму, математична постановка завдання

Завдання: підбір параметрів для оптимальної роботи алгоритму з умови забезпечення якості фільтрації і збереження ключової інформації. M = { m_1 ,..., m_k } — множина ключових точок в зображенні; E = { e_1 ,..., e_k } — множина помилок в зображенні. Необхідно вибрати оптимальний набір параметрів для фільтра зображення, заснованого на перетворенні Габора. Буде проводиться підбір параметра, середньоквадратичної обвідної Гаусса, використовуваної в фільтрі. Кожному значенню δ ставляться у відповідність значення функцій $P_1(\delta)$ і $P_2(\delta)$.

$P_1(\delta)$ — Функція ймовірності помилки першого роду, то є ймовірність внесення фільтром спотворень в ключову інформацію $P_2(\delta)$ — Функція ймовірності помилки другого роду, тобто ймовірність ігнорування, тобто не видаленні шумів у зображенні.

Якщо прийняти f за якусь функцію, яка є лінійною комбінацією 2-х параметрів: $f(\delta) = a_1P_1(\delta) + a_2P_1(\delta)$,

де a_1 і a_2 — вагові коефіцієнти, то завдання можна сформулювати наступним чином — необхідно знайти таке значення параметра δ , що f (δ) = min; δ є R

Рішення поставленого математичного завдання

Для вирішення поставленого в роботі математичної задачі необхідно визначити діапазон значень, в якому буде проводиться підбір параметру δ і кількість кроків підбору N. Нехай $[\delta_{min},\delta_{max}]$ — обраний діапазон значень, тоді: $\Delta = \frac{\delta min - \delta max}{N}$ — величина кроку. Значення δ на кроці і визначається за формулою: $\delta_i = \delta_{min} + \frac{i}{N}(\delta_{max} - \delta_{min}),\ i = \overline{1,N}$.

Необхідно знайти таке i, при якому $f(\delta) = a_1P_1(\delta) + a_2P_1(\delta)$--→min

Приймемо $[\delta_{min},\delta_{max}] = [0.3,1.5]$,

N=24 величина кроку

$\Delta = \frac{\delta min - \delta max}{N} = 0.05$

Результати експерименту показали, що значення функції

$f(\delta) = a_1P_1(\delta) + a_2P_1(\delta)$ мінімально при $\delta = 1.05$. На Рис. 11 проілюстрована робота алгоритму при різних значеннях параметра δ.

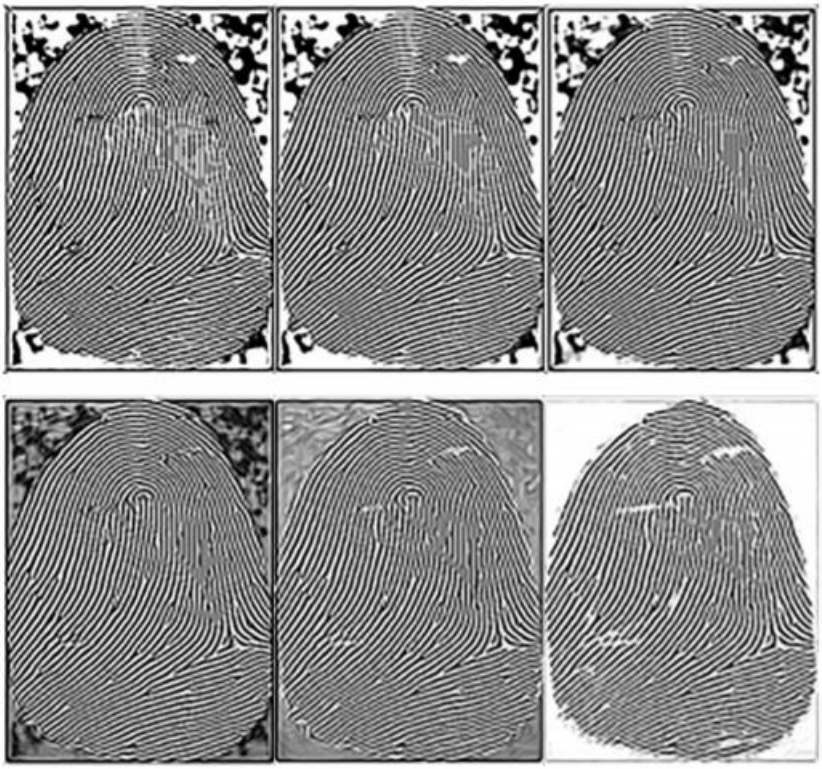


Рисунок 11 - Результати обробки відбитка фільтром Габора при різних значеннях.

2.4 Типові помилки FMR, FNMR, EER

При зіставленні двох відбитків можуть бути висловлені 2 гіпотези:

• дані відбитки належать різним людям (розбіжність) і

• дані відбитки належать одній людині (збіг).

Результатом зіставлення є коефіцієнт схожості, як було сказано раніше. Залежно від обраної границі це значення інтерпретується як збіг або розбіжність відбитків, тобто підтверджується одна з пред'явлених гіпотез.

Для таких гіпотез існує два типи помилок:

- помилка 1-го роду: помилковий збіг відбитків і
- помилка 2-го роду: помилкова розбіжність відбитків.

Відповідно, визначена ймовірність помилкового збігу (False Match Rate, FMR) і ймовірність помилкової розбіжності відбитків (False Non-Match Rate, FNMR).

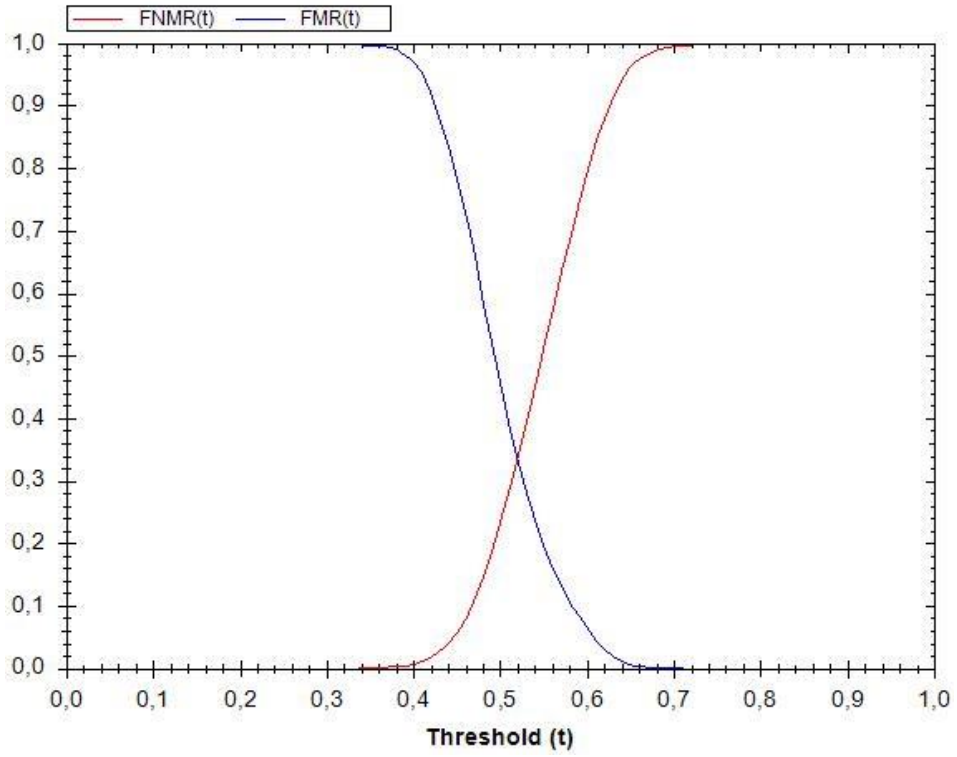


Рисунок 12 – Приклад залежності FMR от FNMR.

Як було сказано раніше дані помилки пов'язані. Якщо зменшити поріг, заради отримання більш стійкої системи по відношенню до вхідних даних, то FMR зросте. З іншого боку, якщо збільшити поріг з метою поліпшення надійності системи, збільшиться FNMR.

Наприклад, в криміналістиці потрібно забезпечити мінімальне значення FNMR. Висока ймовірність помилкової ідентифікації при цьому не є критичною, тому що метою є пошук злочинця, навіть якщо буде потрібно організувати додаткову експертизу для деякого числа подібних відбитків. З іншого боку, для систем безпеки з високим рівнем захисту важливо, щоб значення FMR було якомога менше. В такому випадку = 10. Високе значення FNMR є допустимим. Як правило, більшість біометричних систем вимагають деякого компромісу між значеннями FMR і FNMR.

Для визначення оптимальної границі використовують ROC-криві або DET-криві (рисунок 13). Координата збіжності помилок FMR і FNMR називається коефіцієнтом рівної ймовірності помилок 1-го і 2-го роду (Equal Error Rate, EER). Чим нижче рівень EER, тим система вважається стійкішою. У realtime системах значення EER близько 0,05.

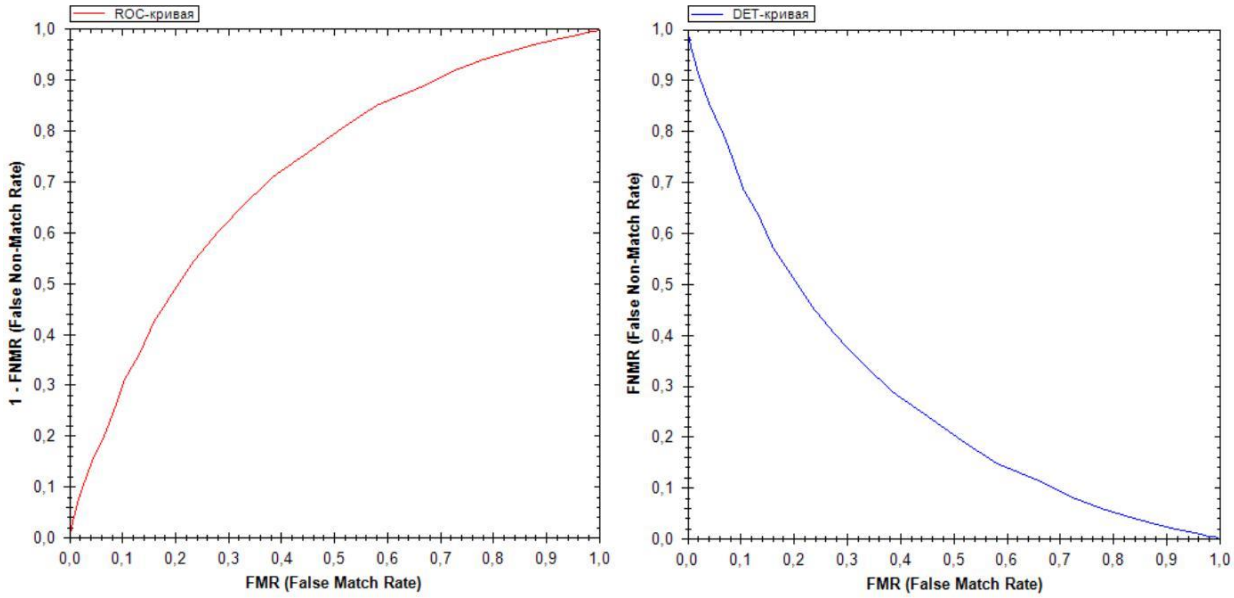


Рисунок 13 – а) ROC-крива, б) DET-крива

3 РЕАЛІЗАЦІЯ ПОШУКУ ЯКІСНИХ ОБЛАСТЕЙ

3.1 Карти якості зображення

Використовуючи NFIQ, якість відбитка описується в залежності від контрастності зображення, визначеності напрямків папілярних ліній й областей збільшеної кривизни. Так для кожного відбитка знаходимо:

- карта пониженного контраста;
- карта лінійних напрямків;
- карта мінімальних напрямків;
- карта збільшеної кривизни

Найрезультативнішою є карта лінійних напрямків бо розташування на сканері пальця, який ми порівнюєму з іншим, не завжди рівно сходится

(поворот пальця, його розміщення тощо) по відношенню іншого відбитка. Об'єднуючи всі описані вище результати, обчислюється унікальна карта якості відбитка. Кожному блоку зображення присвоюється значення від 5 (погана якість) до 1 (відмінну якість). На рисунку 14 показаний приклад карти якості. Області якості 1 позначені білим кольором, а погані 5 - чорним.

Рисунок 14 – Зліва оригінал відбитку, справа його карта якості

3.2 Карта лінійних напрямків

Мета створення цієї карти - показати області з достатньою кількістю ребер та виявити їх загальний напрямок.

Спочатку зображення поділяється на непересічні квадратні блоки зі стороною M = 8 (рисунок 15). Для визначення загального напрямку блоку потрібно розглянути деякий блок - вікно зі стороною L = 24 зміщення блоку щодо вікна N = 8. Якщо розмір зображення не кратний розміру блоку, вікно може зайти за межі зображення. У цьому випадку зображення доповнюється середніми значеннями сірого - 128.

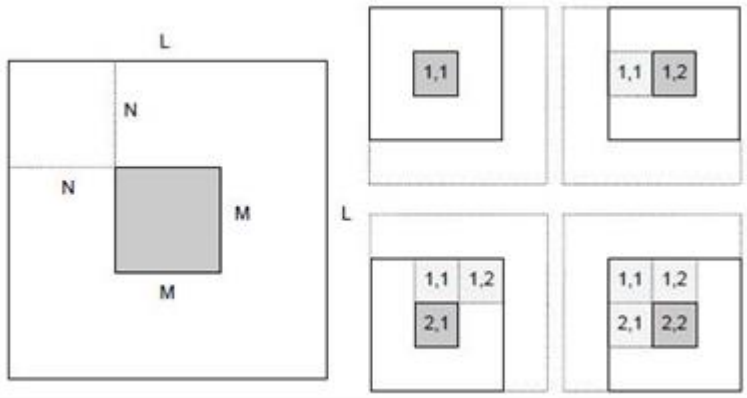


Рисунок 15 – Блоки зображення та вікна

Для кожного блоку зображення ми інкрементально повертаємо його вікно відповідно до кожного з 16 напрямків (кут між напрямками дорівнює 11.25°) і проводимо DFT (дискретне перетворення Фур'є) для кожного напрямку, тобто повороту вікна. Так кожному напрямку відповідає число від 0 до 15, а блокам без певного напрямку відповідає -1.

Приклад напрямків представлений на рисунку 16, а на рисунку 17 - карта лінійних напрямків.

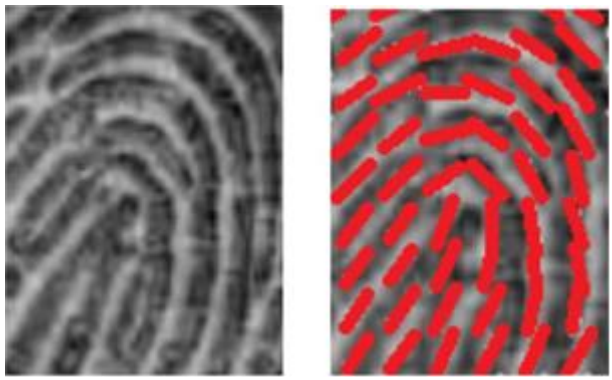


Рисунок 16 – Зліва фрагмент відбитка, справа карта з виділеними напрямками.

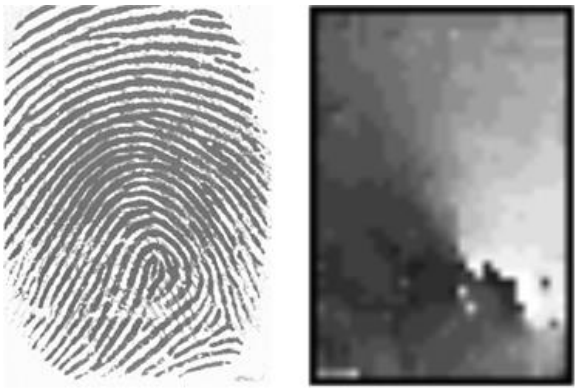


Рисунок 17 – Зліва повне зображення відбитку, справа його карта напрямків.

3.3 Критерії знаходження стійких фрагментів відбитків

Обчислення моделей за допомогою алгоритму NFIQ визначало якість всього зображення відбитка. Однак, відбиток має свої якісні фрагменти, за якими відбиток може бути ідентифікований. Якщо коєфіцієнт таких областей замалий, то зображення буде оцінено як неякісне, навіть якщо в ньому міститься достатньо правдивої інформації.

Для того, щоб була можливість використовувати зображення відбитків в таких ситуаціях, було прийнято рішення використовувати для зіставлень тільки якісні області відбитків, тобто блоки якості

1, 2, 3. Таким чином будуть отримані достовірні результати навіть для тих відбитків, які визнані неякісними.

4 АРХІТЕКТУРА РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Структурна модель програмного забезпечення

Даний проект характеризується структурною схемою (рисунок 18) на якій зображено модулі послідовної обробки зображень відбитків пальців. Вхідними даними варто вважати файл зображення відбитку (tiff, bmp), вихідними параметрами будуть відредактовані відбитки за алгоритмом покращення якості та данні похибок для створення графіків.

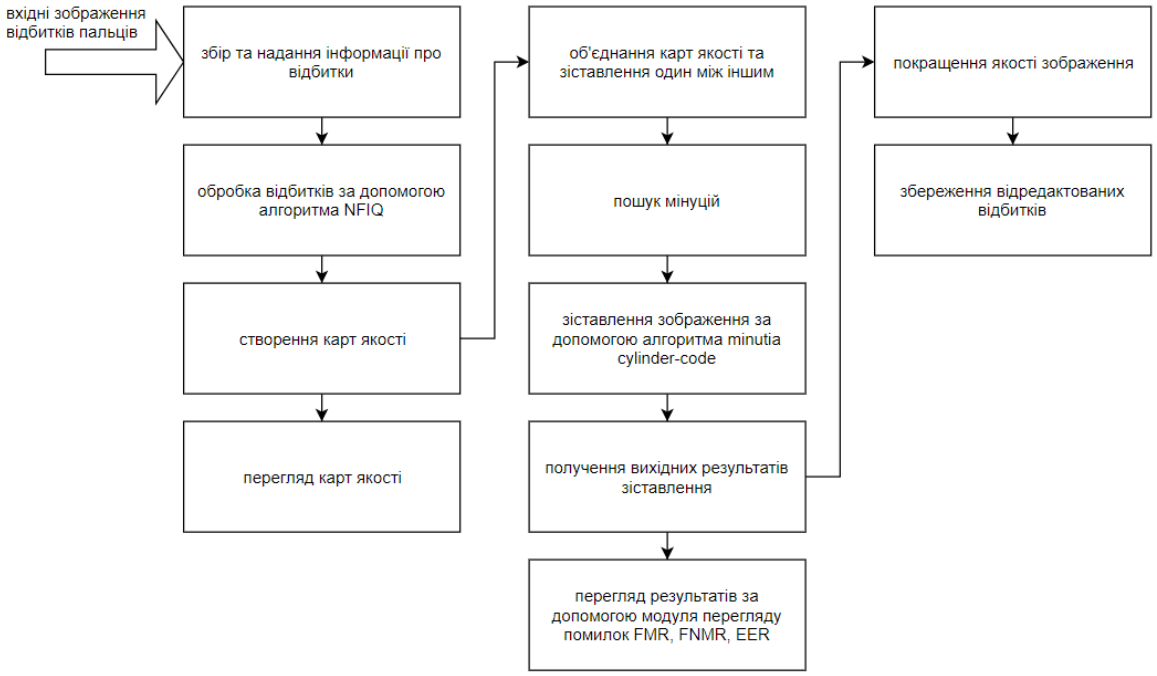


Рисунок 18 – Структурна модель програмного забезпечення

4.2 Модель розробки програмного забезпечення

В сучасному цифровому світі існує декілька моделей розробки програмного забезпечення, але кожна з них особлива, саме тому використовується в різних варіаціях. Для покращення потужності та стабільності роботи програмного забезпечення використовують метод комбінування моделей розробки, оскільки досить проблемно продумати всі

можливі варіанти рішеннь, задля знаходження специфікації окремої моделі. Саме через функціональні модулі добряче розрізняються між собою, більшість розробників приходять до рішення комбінувати різні моделі і таким чином знаходити найбільш оптимізований та стабільний варіант, що задовольняє більшість потреб програмного забезпечення.

Найпопулярніші моделі сучасноті:

- структурна;
- об’єктна;
- компонентна.

Структурна модель – це модель розробки програмного забезпечення, яка заснована по принципу функціональної декомпозиції, структура якої являє собою систему що описана за допомогою термінів ієрархії та її функціональних особливостей та передачі даних між окремими елементами. Можно вважати що використання даної моделі - це автоматизоване розбиття на підсистеми, що поділяються на підфункції, що, в свою чергу, розбиваються на підзадачі – і це розбиття триває аж до конкретних процедур. Інформаційна система зберігає цілісність інформації, де всі складові є взаємопов’язаними.

Об’єктна модель – це модель, основою для об’єктно-орієнтованого аналізу та для проектування програмного забезпечення. Основними принципами даної моделі є: абстрагування, інкапсуляція, наслідування та поліморфізм, також використовуються поняття: об'єкту, класу, атрибута, методу, потоку тощо. Об’єктна модель в основному використовується при описі структури у дефініціях об’єктів і зв’язків між ними, а вже поведінка системи описується в термінології обміну повідомлень між об’єктами.

Компонентна модель – це модель, основою даної моделі є визначення компонентів програмного забезпечення, що в свою чергу є простими структурними елементами, що можна повторно використовувати при побудові програмних систем. Ці компоненти відповідають за реалізацію

прикладних функцій інформаційної системи, також їх використовують для семантичної характеристики прикладного чи технічного характеру. Вони можуть модифікуватися в процесі розробки програмного забезпечення на рівнях двійкових кодів Компоненти існують та функціонують лише безпосередньо всередині контейнерів. Контейнери, в свою чергу, відтворюють загальну картину взаємодії між окремими компонентами прикладних завдань, також компонентам вкладеним в інші компоненти надається стандартний доступ до можливостей середовища виконання. За основу даної моделі взаємодії взятий механізм публікації та підписки, що надає можливість встановлювати зв'язки між компонентами.

Для успішного керування проектами розробки програмного забезпечення потрібні дві важливих якості. Перше – строгість – необхідно для постійного контролю стану проекту. Друге – гнучкість – потрібно для успішної адаптації до неминучих несподіванок. Більша частина цього розділу присвячена моделі процесу розробки додатків MSF(MSF Process Model for Application Development), або, коротше, моделі процесу розробки. Модель процесу MSF – це не детальна інструкція, а структурний каркас, на базі якого організації можуть створювати конкретні способи рішення своїх завдань. Модель процесу розробки – складова частина цієї структури, що описує життєвий цикл проекту розробки програмного забезпечення. Вона дозволяє проектній групі створювати продукт у постійному контакті із замовником і адаптувати процес відповідно до його побажань. Крім того, цей метод здатний забезпечувати найшвидшу реалізацію ключових складових проекту. Модель процесу розробки додатків MSF – гнучкий компонент загальної моделі процесу MSF. с успіхом застосовуваний в індустрії розробки програмного забезпечення для підвищення керованості проектів, мінімізації ризиків, підвищення якості продукції й прискорення розробки.

Для розробки програмного забезпечення дипломної роботи було обрано і використано об'єктну модель, оскільки в основі об'єктної моделі

лежить опис необхідної поведінки програмного забезпечення, що є дуже важливим при повному циклі розробки нового функціоналу.

Під час моделювання було розроблено діаграму класів (ЗІА51.020БАК.004.Д1), на якій відображено відносини сутностей.

4.3 Архітектура програмного забезпечення

Основою побудови архітектури системи є процес визначення цілей системи, вхідних та вихідних даних, методу декомпозиції системи та підсистем. Процес створення документації архітектури програмного забезпечення є основою для спрощення процесу комунікації між розробниками, замовником та командою бізнес-аналітиків і менеджменту у майбутньому. Створення та тестування функціонуючих макетів дизайну майбутнього програмного забезпечення надає можливість, визначити основні функціональні дефекти та зручність інтерфейсу для користувача, ще на етапі проектування та моделювання.

На сьогодні існує декілька видів архітектури розробки програмного забезпечення, які в свою чергу поділяються в залежності від завдань, які має виконувати програмне забезпечення. Саме після визначення основних функціональних та нефункціональних модулів починається сам процес вибору архітектури.

Основними властивостями, що впливають на вибір необхідної архітектури для подальшої розробки вважають: час, витрачений на розробку програмного забезпечення, задана вартість всього проекту, можливість покращити існуючі чи додати нові модулі програмного забезпечення без ризику руйнування цілостності проєкта.

На даний момент найбільш дієвими є такі архітектурні рішення:

- багаторівневе;
- розподілене;

- модульне;
- сервісно-орієнтована архітектура
- подійно-орієнтована архітектура.

Багаторівневу архітектуру ми можемо співвіднести до модульного типу уявлень, а вже взаємодію між модулями можна описати як «один є частиною іншого» і «один успадковує від іншого». Кожен рівень, по суті, має стати об’єднанням відповідного коду і діяти як незалежна віртуальна машина. В даному типі архітектурного рішення зв’язки між рівнями мають ієрархічну структуру, а саме: рівні більш вищого значення передають управління рівнями нижчого значення. Елемент на одному рівні повинен відправляти інформацію елементу в наступному рівні, і на цьому його дія має бути завершена, оскільки він не очікує повернення інформації.

Розподілена архітектура – це архітектурне рішення, що передбачає розташування компонентів програмного забезпечення на мережевих комп’ютерах, а передача та прийом інформації, а елементи координації відбуваються шляхом передачі повідомлень. Компоненти даної архітектури повинні взаємодіяти одне з одним заради досягнення спільної мети. Для розподілених систем, зазвичай, визначають три основні характеристики: паралелізм компонентів, відсутність глобального часу, і незалежний вихід з роботи компонентів.

Модульна архітектура – це вид архітектури, що є поділом функціональних частин програмного забезпечення на незалежні модулі, такого формату, що кожен з них може містити все необхідне, аби виконати тільки один вид необхідної функціональності. Модулі, як правило, реалізовують у програму за допомогою інтерфейсів.

Подійно-орієнтована архітектура – це своєрідне архітектурне рішення, за якого майже будь-яка подія визначається як зміна стану. Даний архітектурний підхід застосовується переважно при проектуванні і розробці

додатків і систем, що передають події між компонентами із нестабільно пов’язаними модулями програмного забезпечення.

Сервісно-орієнтована архітектура — це архітектурний шаблон програмного забезпечення, що включає в себе розбиття програмного забезпечення на модулі до безпосередньої розробки проєкту. Даний підхід бере за основу використання розподілених, слабко пов'язаних компонентів, які потрібно замінити. Замінні компоненти повинні бути оснащенні стандартизованими інтерфейсами для швидкої та коректної взаємодії за уніфікованими протоколами.

Під час розробки програмного забезпечення досить важко дотримуватись лише одного конкретизованого архітектурного рішення, оскільки досить часто специфікація функціональних частин додатку є досить широкою.

Важливим фактором архітектури є якість, опис якої поділяється на: погану (загниваючу), стабільну та покращену

Закритість (rigid) — система відчайдушно чинить опір змінам, неможливо сказати, скільки займе реалізація тієї або іншої функціональності, тому що зміни, швидше за все, торкнуться багатьох компонентів системи. Через це вносити зміни стає занадто проблематично, тому що вони вимагають багато часу на рефакторинг.

Нестійкість, крихкість (fragile) — система ламається в непередбачених місцях, хоча зміни були проведені до цього, але “зламані” компоненти не змінювали.

Нерухомість або монолітність (not reusable) — система побудована таким чином та характер залежностей такий, що використовувати будь-які компоненти окремо від інших не можливо.

В'язкість (high viscosity) — код проєкту такий, що зробити що-небудь неправильно набагато простіше, ніж зробити це правильно.

Невиправдані повторення (high code duplication) — розмір проєкту набагато більший, ніж в тому випадку, якби використовували абстракції замість фізичних обьектів.

Надмірна складність (overcomplicated design) — проєкт містить рішення, користь від яких неочевидна, вони приховують реальну суть системи, ускладнюючи її розуміння і розвиток.

Принцип відкриття-закриття (Open Close Principle або OCP) Програмні сутності такі як класи, модулі та функції повинні бути відкриті для розширення, але закриті для змін. Подібний же принцип застосовується для модулів, пакетів і бібліотек. Якщо у проекті є бібліотека, що складається з множини класів, буде доречним розширення функціоналу замість зміни коду, який вже написаний (заради зворотної сумісності, повернення до попереднього тестування і т.д.). Це причина, по якій розробник повинен розуміти, що модулі слідують Принципу відкриття - закриття. По відношенню до класів Принцип відкриття-закриття може бути корисний за рахунок використання Абстрактних Класів і конкретних класів для реалізації їх поведінки.

Принцип Заміщення Ліскоу (Liskov's Substitution Principle) Похідні типи повинні бути здатні повністю заміщатися їх базовими типами. Цей принцип є всього лише розширенням Принципу відкриття-закриття в умовах поведінки, що означає, що розробник повинен бути впевнений, що нові похідні класи є розширенням базових класів без зміни їх поведінки. Нові похідні класи повинні мати здатність замінювати базові класи без будь-яких змін у коді.

Принцип Єдиної Відповідальності (Single Responsibility Principle) Клас повинен мати тільки одну причину для зміни. У цьому контексті відповідальність розглядається як єдина причина для зміни. Цей принцип стверджує про актуальність розподілу функціонального обьекту на два класи, якщо причини задовольняють правило. Кожен клас повинен мати тільки одну

відповідальність, і в майбутньому, якщо потрібно зробити одну зміну, то це буде реалізоване в класі, який утримує цю одну відповідальність. Коли розробнику потрібно робити зміни в класі, який має більше відповідальностей, то заміна може вплинути на іншу функціональність класів.

Принцип Відділення Інтерфейсу (Interface Segregation Principle) Клієнти не повинні бути залежними від інтерфейсів, які вони не використовують. Цей принцип описує правильність інтерфейсів. При реалізації інтерфейсу, треба подбати про додавання тільки тих методів, які там повинні бути. Якщо при додаванні методів, які не повинні знаходитися в даному контексті, класи, що реалізують інтерфейс будуть змушені реалізовувати зайві методи.

Принцип інверсії залежностей (Dependency Inversion Principle) Залежності всередині системи будуються на основі абстракцій. Модулі верхнього рівня не залежать від модулів нижнього рівня. Абстракції не залежать від подробиць. Даний принцип дуже важливий і гідний докладного розгляду. Якщо додаток є гарним прикладом вдалого дизайну архітектури, то він, в тій чи іншій мірі, дотримується принципу інверсії залежностей.

Ознаки покращенної архітектури:

Висока зв’язаність коду (High Cohesion).

Код, відповідальний за будь-яку одну функціональність, повинен бути зосереджений в одному місці.

Низька зв'язаність коду (Low Coupling).

Класи повинні мати мінімальну залежність від інших класів.

Вказуй, а не питай (Tell, Don't Ask).

Класи містять дані і методи для оперування цими даними. Класи не повинні коннектитися з даними з інших класів.

Не розмовляй з незнайомцями (Don't talk to strangers).

Класи повинні знати тільки про своїх безпосередніх сусідів. Чим менше знає клас про існування інших класів або інтерфейсів - тим більш стійкий код. Всі ці рекомендації спрямовані на те, щоб постаратися розвести класи по сторонам, зосередити сильні взаємозв'язки в одному місці і провести чіткі розмежувальні лінії в коді. Але ці принципи надто розпливчасті, тому з'явився набір більш чітких правил, якими слід керуватися при формуванні архітектури.

Принцип персональної відповідальності (Single Responsibility Principle). Клас володіє тільки однією відповідальністю, тому існує тільки одна причина яка приводить до його зміни.

Принцип відкриття-закриття (Open-Closed Principle)

Класи повинні бути відкриті для розширень, але закриті для модифікацій. Здається, що це неможливо, однак варто згадати шаблон проектування Strategy.

Принцип підстановки Ліскоу (Liskov Substitution Principle) Дочірні класи можна використовувати через інтерфейси базових класів без знання про дочірність класу. Інакше - дочірній клас не повинен заперечувати поведінку базовогу класу і повинен мати можливість використовувати дочірній клас скрізь, де використовувався базовий клас.

Принцип інверсії залежностей (Dependency Inversion Principle). Всередині системи взагалом використовують абстракцій. Модулі верхнього рівня не залежать від модулів нижнього рівня. Абстракції не залежать від подробиць.

Принцип відділення інтерфейсу (Interface Segregation Principle).

Клієнти не повинні потрапляти в залежність від методів, якими вони не користуються. Клієнти визначають, які інтерфейси їм потрібні.

4.4 Опис розроблених модулів системи

При проектуванні дипломного проекту було розроблено 4 модулі, а саме це : інтерфейс взаємодії користувача, NFIQ-аналізатор, MCC-компаратор, бібліотеки графічного забезпечення.

Описуючи модуль користувацького інтерфейсу слід зауважити, що основною проблемою реалізації цього модулю було некоректне відображення інформації або зовсім несумісність із системою на різних операційних системах, бо мова C# (окрім кроссплатформенного фреймворка Core) використовується только під ядро NTOSKERNEL32 (Windows OS). Але за допомогою методів сучасного програмування було створено об'єкти графічного відображення для розуміння аналізу та його виведення на екран.

Модуль, що відповідає за інтерфейс користувача містить у собі класи та методи, що забезпечують коректну взаємодію користувача із вбудованими алгоритмами обчислювальних проектів. Взаємодія між проектами зумовлена об'єктно-орієнтовною моделю програмування та модульної структурою, яка позволяє без перешкод використовувати класи та методи різніх під проектів в единому користувацькому ПЗ. В даному модулі був вказаний акцент на те, що основні дії користувача, під час взаємодії з додатком, повинні в першу чергу швидко відпрацьовуватися і мати адаптивно-зрозумілий дизайн та вико діюю взаємодії між використовуємими даними. Дії користувача в даному модулі пов'язані з методами інших модулів, що, в свою чергу, забезпечує вільну взаємодію без додаткових ПЗ. Адаптивно-зрозумілий дизайн не потребує спеціальної підготовки дактилоскопіста або далі Користувача для подальшої роботи.

Наступним модулем було реалізовано NFIQ-аналізатор. Особливість цього модулю полягає в обробці зображення відбитку пальця, на виході з якого, ми получаємо карту якості (як основний потрібний файл аналізу) та

інші карти характеристики зображення. До іншиї карт зображення входять такі карти:

– PrintMap(direction_map, @"..\..\..\Results\direction_map.txt");

Карта напрямків потрібна для аналізу напряму папілярних ліній згідно квадрату оцінювання, під який ми вирівнюємо відбиток

– PrintMap(low_contrast_map, @"..\..\..\Results\low_contrast_map.txt");

Карта низьких контрастів потрібна для розпізнавання контрастних частин відбитка для розуміння темних та світлих областей

– PrintMap(low_flow_map, @"..\..\..\Results\low_flow_map.txt");

Карта низьких розмитостей потрібна для розпізнавання нечітких частин відбитка для розуміння зхованих областей

– PrintMap(high_curve_map, @"..\..\..\Results\high_curve_map.txt");

Карта високої кривизни потрібна для аналізу угла поворота кривих.

Зчитування та відображення інформації проходить через інтерфейс користувача, але безпосередньо обробка протікає через математичні алгоритми проекту та середовища обробки спеціальних видів зображень. Вибір декількох даних за одну дію дещо спростило та пришвидшило роботу із даними, проте викликало деякі дефекти під час взаємодії цього модулю із модулем інтерфейсу користувача, а саме затримки в відображенні послідовних сторінок при створенні їх в контролах форми згідно с тільки що створенними даними якості зображення.

Алгоритм створення карти якості реалізован на основі додавання (апроксимації адитивних характеристик) карт напрямків, контраста, потоків, кривизни, а також параметрів розміру ширини и довжини блоку. Остаточна карта виглядає як матриця N*M, в основі якої цифри від 1 до 5.

За допомогою інтерфейсу користувача ми наглядно можемо побачити цю матрицю на формі, з підсвіченими значеннями від червоного(4) до світло-зеленого(1) та до голубого (0).

Далі реалізован модуль порівняння карт якості кожного з відбитків одного і того же пальця на основі алгоритму MCC (Minutia Cylinder-Code). Сам алгоритм був вбудований до проекту як бібліотека даних. Під мову C# мною використовувалась SDK від проекту Biolab, також на цьому сайті знаходяться зображення відбитків різної якості для тестування та проведення якісних експериментів не застосовуючи дактилоскопічні прибори для самостійного створення зображень або їх повірку. Цей модуль використовується на формі користувацького забезпечення як основний та послідовний у своїх діях для налаштування різних видів похибок.

Вхідним параметром є декілька зображень, які користувач додає через інтерфейс користувача, на виході через вікно збереження файлу користувач вказує розміщення файлу та його назву; в файлі знаходятся координати суммарної інформації та показчик кількості папілярних ліній. Однак я зіткнувся з однією проблемою – зображення відбитків подається вже унормованим та без похибок, цей фактор дає майже рівний графік. З одної сторони ми маємо еталон якості, але в разі пошуку відхилення ми зовсім не побачимо результативного висновку.

Модуль візуалізації від компанії NVIDIA – CUDA, безкоштовна бібліотека для розробників графічного ПЗ, основана на взаємодії графічних процесів відео карти представленних, як методи, за допомогою яких на форму інтерфейсу користувача можливо вивести графічний процес.

Наприклад, представлення якогось bitmap відбитку (як обьект CUDA.Fingerprintings::Dictionary) через графік (Graphics.Chart) у вигляді лінійної залежності однієї вибранної величини від іншої. В данному випадку папілярні лінії, які пройшли перевірку на FMR\FNMR.

Вивід результатів графіку виводится на окремій формі. Графік має площину та 2 відносини FMR та FNMR, апроксимуючою величиною это EER. Різні мінуції показані на графіку у вигляді DET-кривих.

В майбутньому можлива реалізація автоматичного створення графіку на першій формі UI.

Для кращого розуміння взаємодії між модулями було також створено діаграму взаємодії користувача та діаграму прецендентів (Дод. Б,В)

4.5 Детальний опис фрагментів та класів

Існує декілька методів оформлення та використання коду для забезпечення стабільної роботи програмного забезпечення. В нашому дипломному проекті під час розробки проектів мовою C# під ОС Windows було використані такі технології:

- BioLab.Biometrics.Mcc.Sdk
- CUDAFingerprinting.*
  *.Common;
  *.Common.Segmentation;
  *.Common.OrientationField;
  *.ImageEnhancement.LinearSymmetry;
  *.TemplateBuilding.Minutiae.BinarizationThinning;
- C#.Net (Linq, Collections)
- ZedGraph

В рамках даної платформи використовується стандартна система типів Common Type System (CTS), яка повністю описує всі типи даних, підтримувані середовищем виконання, визначає взаємодію типів даних та їх представлення в форматі метаданих .NET.

Набір правил, що визначають підмножину загальних типів даних, у відношенні яких гарантується, що вони небезпечні при використанні у всіх мовах .NET, описується в рамках специфікації Common Language Specification (CLS). Для того щоб класи, розроблені на різних мовах, можна

було спільно використовувати в рамках однієї програми, вони повинні відповідати певним обмеженням, що задається CLS.

Клас, що задовольняє CLS, називається CLS-сумісним. Він доступний для використання в інших мовах, класи яких можуть бути клієнтами або спадкоємцями сумісного класу.

Платформа .NET надає в розпорядження програміста бібліотеку базових класів, доступну з будь-якої мови програмування .NET. Оскільки число класів бібліотеки FCL досягає декількох тисяч, то в цілях структуризації функціонально близькі класи об'єднуються в групи, звані простором імен (Namespace).

Основним простором імен бібліотеки FCL є простір System, що містить як класи, так і інші вкладені простору імен. Наприклад, в просторі System.Collections знаходяться класи та інтерфейси, що підтримують роботу з колекціями об'єктів - списками, чергами, словниками. Простір System.Windows.Forms містить класи, використовувані при створенні windows-додатків.

The Compute Unified Device Architecture (CUDA) є однією з мов обчислювальної техніки GPU, яка підтримує мову «C» і «Fortran» для ефективного розгортання алгоритмів. Керуючись своєю унікальністю, незмінністю, прийнятністю відбитки пальців знаходяться на передньому плані між біометричними ознаками.

Нещодавно GPU розглядалася як перспективна паралельна технологія обробки завдяки високій продуктивності, товарності та доступності. Перевірка автентичності відбитків пальців продовжує зростати і включає розгортання багатьох алгоритмів обробки зображень і комп'ютерного зору. У цій роботі представлена локальна інваріантна система вилучення відбитків за допомогою двох домінуючих детекторів, а саме SIFT і SURF, які працюють на процесорі і GPU. Експериментальні результати показують, що реалізація GPU дає багатообіцяючу поведінку, як для SIFT, так і для SURF порівняно з

процесорною. Крім того, детектор функцій SURF забезпечує більш короткий час обробки порівняно з реалізаціями SIFT CPU і GPU.

Під час розробки було створено 3 проєкти, що забезпечують основну функціональність, а також 1 GUI – що являє собою життєвий цикл програмного забезпечення з візуальним адаптивно-зрозумілим інтерфейсом.

Описуючи головний проект Client.UI було визначено, що будуть задіяні такі методи:

– ROC.GetPositiveRates(List<string>, StreamWriter):void – метод, який використовує колекцію ссилок на зображення як перший параметр та обьект запису данних у файл. Метод використовує NFIQ алгоритм як метод для створення карти якості, яку в наступному кроці передає до MCC-компаратора. Вихідне значення – результат запису данних мінуцій різних порядків для подальшого використання у візуалізаторі.

– GetQualityMap(bool):string – цей метод визиває метод IntArraytoString(int[,]), за допомогою якого ми конвертуємо масив цілочисленних значень в строку у вигляді матриці. Параметр массиву приймає значення методу Nfiq.GetQualityMap(string), який приймає на вхідний параметр строку ссилки на зображення, повертаючи массив інтів.

– colortext_sintacsis_html(RichTextBox, string, Color):void – метод, який приймає на вхід екземпляр прив’язки контрола, підстроку, яку ми бажаємо виділити та колір. Метод шукає значення від 0 до 5 (якість) та підсвічує їх.

– btnSaveSummary_Click():void – метод тиснення на кнопку при якому виконується метод GetPositiveRates;

Наступним вказуємо опис ключових полей і методів класу FingerprintData.

Структура класу:

– List<FingerprintData> Items – статична колекція обьектів даного класу.

– FilePath – це властивість, яка описує посилання на зображення.

– FileName – поле, яке автоматично шукає назву файла, використовуючи властивість посилання

– GetQualityMap():string – перегруженний метод, який конвертує інти в строку, також має оригінальний метод GetQualityMap():int[,] -

– IntArraytoString(int[,]):string – метод конвертації масива інтів в строку;

Розглянемо проект NFIQ:

Ключовим класом вважается GetMin, розглянемо його структуру

– GetQualityMap(Bitmap):int[,] – метод, що має одну перегрузку, параметром якої можно вважати не посилання на обьект зображення, а строку на місце розташування файла зображення. Метод визиває калькулятор якості

– CalculateQualityMap(Bitmap):void – метод, що має одну перегрузку, параметром якої можно вважати не посилання на обьект зображення, а строку на місце розташування файла зображення. Метод визиває пошук мінуцій та генератор карти якості

Розглянемо проект ResultsExtraction

Структура класу ROC:

– GetPositiveRates(string, string):void – метод, що має одну перегрузку. Метод використовує папку із зображеннями та сканує її, добавляючи посилання в колекцію. Далі за допомогою MCC знаходяться схожі відбитки та записуються в 2 коллекції – FMR, FNMR. Далі всі данні передаются на метод, який записує сумарний результат у файл.

– SetMatchParameters():void – метод, що викликає пошук однакових відбитків за допомогою MCCSDK. А саме MccSdk.MccSdk.SetMccMatchParameter(string/resource)

– Match(object, object):double – метод, що викликає пошук однакових шаблонів за допомогою MCCSDK. А саме MccSdk.MccSdk.MatchMccTemplates(object, object);

– PrintResult(int[],int[], int, int, string):void – метод запису суммарної інформації у файл. Вхідними параметрами є fmr, fnmr, кол-во сходжень і розходжень.

– IsGoodMinutia(int):bool – шаблонний метод пошуку потрібних якісних мінуції

Розглянемо клас MinutieExtractor:

- ExtractCUDAFpMinutiae(double[,], int, int):List<Minutia> - метод бібліотеки CUDA, який нормалізує зображення для чіткості пошуку міну цій

Методи для покращення якості:

- SegmentImage – метод сегментації зображення
- BinarizeImage – метод бінарізації зображення
- ThinImage – метод тонкомпенсації зображення
- MakeOrientationField – метод створення поля для орієнтації зображення

FindMinutiae():List<Minutia> – метод пошуку мінуцій з додатком їх в лист

Розглянему структуру FingerprintStruct

- Finger:string – строкове поле описуюче назву відбитка
- Template:object – поле-обьект, який вказує на шаблон відбитків

Розглянему клас FingerprintsDictionary

- ListOfFingerprints: List<FingerprintStruct> - колекція обьектів
- DictionaryOfSame: Dictionary<string, List<FingerprintStruct>> – словник схожих строки та списка обьектів пальця
- FingerprintsDictionary – конструктор класу, який має одну перегрузку.
- GetShortName(string):string – метод для знаходження підстроки в строке. В данному контексті пошук назви з посилання
- GetNumberOfFinger(string):int – метод який знаходить номер по повній назві відбитка

- GetFingerprintStructure(string):FingerprintStruct – метод, який за допомогою сконвертованної бази мінуцій знайденної за допомогою вхідного параметра у вигляді посилання на зображення, створює обьект структури. Шаблон мінуції сгенерований за допомогою MCC.

- IsGoodMinutia(Minutia, int[,]):void – метод, який генерує карту мінуцій згідно вхідних параметрів.

- ConvertMinutiae(List<Minutia>):List<MccSdk.Minutia> - метод конвертації із листа класу Мінуція в ліст обьектів організованих мінуцій.

Розглянемо статичний клас Constants, який обумовленний константами:

- int SegmentationWindowSize = 12;
- double SegmentationWeight = 0.3;
- int SegmentationThreshold = 5;
- double BinarizationThreshold = 140.0d;
- int OrFieldWindowSize = 16;
- int OrFieldOverlap = 0;
- int Blocksize = 8;

4.6 Використання типових помилок для порівняння

Щоб задовольнитися правильностью роботи алгоритмів, було проведено 2 показових експеримента Для кожного експертного аналізу були визначені по-різному результати на рисунку 18,19.

Для експериментів було взято 2 архіви відбитків з відкритих баз даних FVC 2000 by Biolab. Алгоритми виставлення карти якості та пошуку мінуцій, реалізованих на C # для платформ .NET 4.5.

Рисунок 19 – Схема генерації шаблона відбитка

Рисунок 20 – Зіставлення шаблонів з помилками FMR, FNMR, EER

Показники біометричних помилок системи. FMR і FNMR для заданого порогу t відображаються над розподілом справжнього і самозванця; FMR - це відсоток пар, які не мають мат, чиї відповідні оцінки більші або дорівнюють t, а FNMR - це відсоток пар, чиї відповідні оцінки менше, ніж t. Вибір результатів різних робочих точок у різних FMR та FNMR. Крива, що відноситься до FMR до FNMR при різних порогових значеннях, називається робочими характеристиками приймача (ROC).

Типові робочі точки різних біометричних додатків відображаються на кривій ROC. Відсутність розуміння частоти помилок є основним джерелом плутанини в оцінці точності системи у спільнотах постачальників / користувачів.

4.7 Використання типових помилок для порівняння

Для зіставлень використовувалася .NET бібліотека MCC SDK. Можна виділити кілька етапів роботи:

створення шаблонів відбитків (MccSdk.CreateMccTemplate)

встановлення параметрів повірки (MccSdk.SetMccMatchParameters)

зіставлення шаблонів відбитків (MccSdk.MatchMccTemplates)

Шаблони створюються на основі списку Мінуцій відбитка. Таким чином для кожного випадку, вибранного із областей, бралися Мінуцій з цих областей. Як параметри зіставлення були взяті стандартні значення, надані бібліотекою, за винятком методу визначення локального схожості. Був використаний метод LSS, як найбільш швидкий. Як було сказано раніше, схожість відбитків виражається числом від 0 (відбитки належать різним людям) до 1 (відбитки належать одній людині). Підсумкове значення ідентифікації або неідентіфікаціі відбитків залежить від заданого порогу.

4.8 Помилки FMR, FNMR і EER і порівняння результатів

Для кожної вибірки Мінуцій були обчислені помилки FMR, FNMR в залежності від порогового значення. Також були обчислені EER, як точки збігу помилок FMR і FNMR, і були побудовані графіки DET-кривих (рисунок 21).

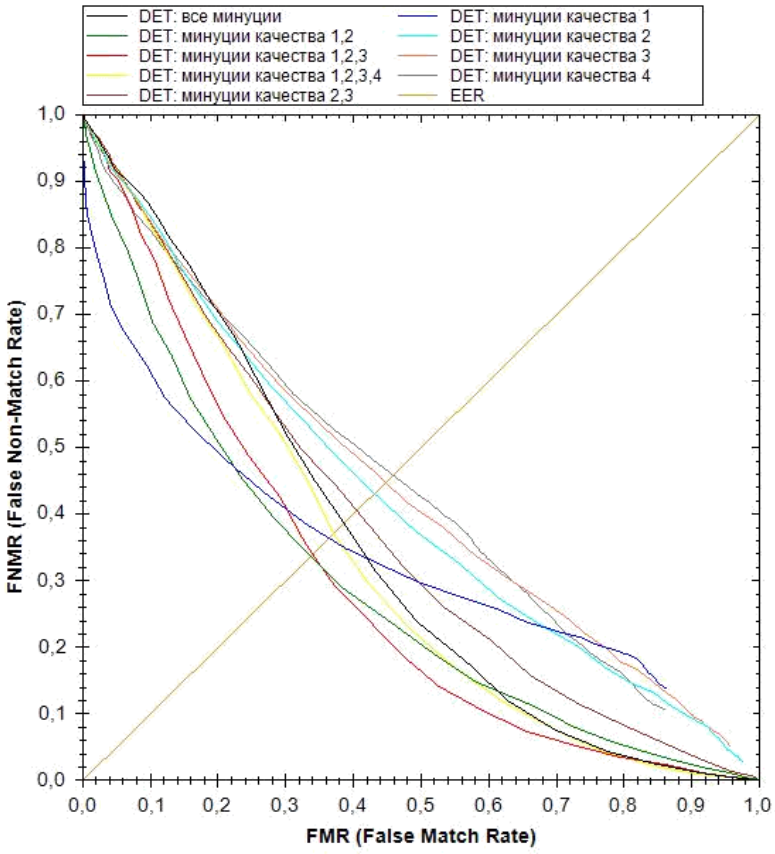


Рисунок 21 – графік порівняння

За результатами видно, що для невеликих вибірок Мінуцій з блоків одного якості 2 або 3, або 4, DET графік стає більше схожий на лінійну залежність y = x. Якщо дотримуватися такого вибору Мінуцій, рішення системи ідентифікації будуть схожі на випадкове ворожіння.

Також варто відзначити той факт, що у кожного з цих графіків максимальне значення ймовірності помилкового збігу FMR не дорівнює 1. Це означає, що яким би малим ні поріг, все одно деякі відбитки, що належать різним людям, не співпадуть.

Особливо варто відзначити графік для Мінуцій якості 1. Видно, що в даному випадку максимальне значення обох ймовірностей не досягає 1, тому до описаного вище варто додати, що яким би високим ні поріг, все одно деякі відбитки одну людину будуть розпізнані як однакові. Причина виникнення подібних особливостей прихована в недостатній кількості даних.

Розглядаючи інші випадки видно два найбільш підходящих графіка: при вибірці Мінуцій якості 1, 2 і при вибірці 1, 2, 3. Видно, що значення EER у них досить близькі, більш точні результати можна подивитися в Таблиці

Можна сказати, що вибір Мінуцій якості 1, 2, 3 застосовні там, де помилкова ідентифікація не так критична, як помилкова неідентіфікація.

Нижче ви побачите результати обчислення EER для всіх випадків в порядку убування. Відразу варто обмовитися, що значення трохи вище, ніж у реальних біометричних систем, через точності алгоритму виділення Мінуцій. Видно, що найменше значення EER досягається при виборі Мінуцій якості 1, 2, однак, так як різниця між ними досить мала (різниця дорівнює 0.00337), то її можна інтерпретувати як статистичний шум при даній кількості оброблюваних відбитків.

4.9 Інструкція користувача

Компоненти Програмного Забезпечення

Програму розроблено мовою програмування C# у середовищі розробки Visual Studio 2017 (Pro version) з використанням MCC SDK, CUDA fingerprint що дає можливість чітко визначати відбитки із зображень.

Для коректної роботи додатку не потрібні пристрої із потужними обчислювальними характеристиками, оскільки для відображення та обрахунку даних не використовується робота над складними математичними обрахунками. Але для правильного функціонування додатку потрібно мати CPU + GPU:

– операційна система: ОС Windows XP+ (SP3 with net framework 4.5+)/Win 7/ Win 10

– підключення до мережі інтернет;

– клавіатура та миша;

– кольоровий графічний дисплей з розподільною здатністю 800x480 і вище;

– процесор із частотою роботи 1.0 GHz і більше;

– оперативна та постійна пам’ять.

Детальна інструкція знаходження мети проекту:

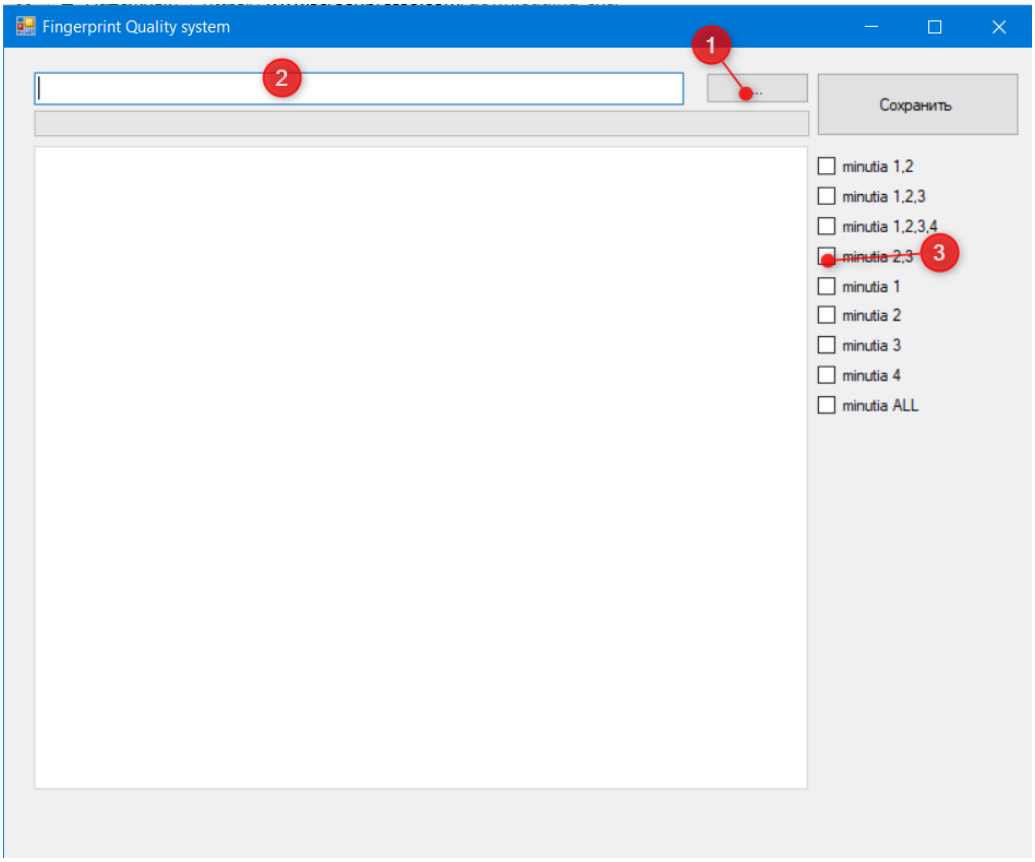


Рисунок 4.9.1 – Основне вікно користувача

На данному вікні користувацького інтерфейсу ми бачимо кнопку відкриття фалів (1), нажимая на неї ми бачимо вікно з файловим менеджером де ми можемо вибрати як один так и декілька файлів. В формі для текста (2) будуть відображені файлі через кому. Справа панель (3) дає вибір які типи мінуцій ми будемо використовувати. Чим їх більше, тим точніше буде вибірка. Далі через деякий час, після додавання файлі, з’являється вікно з картами які ми бачимо на рисунку 4.9.2

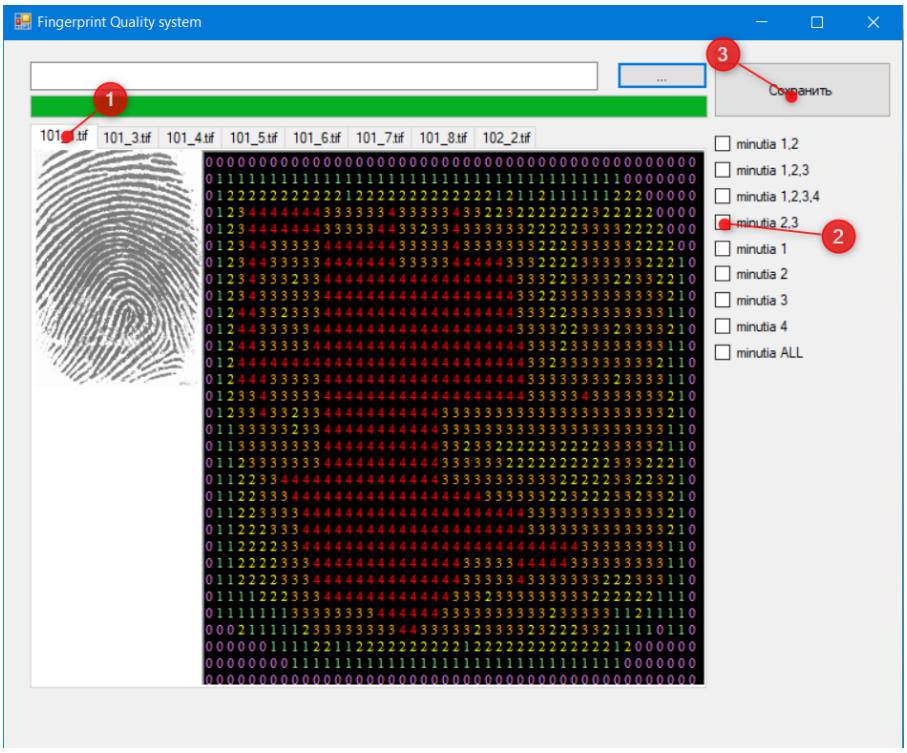


Рисунок 4.9.2 – Використання зображення та їх відображення

Далі для знаходження сумарної інформації по мінуціям різних рівнів, ми використовуємо вкладки для візуального розуміння (1) та панель з вибором. Користувачу треба вибрати одне із значеннь (2) та натиснути на кнопку (3) для збереження той чи іншої інформації. Алгоритм розуміє збереження цих файлів в корень проекту в папку /Results/ROC/**, де ** - 23,123,1234 – кількість мінуцій, які було вибрано. Так по одному зберігаються різні рівні мінуцій. На це треба декілька секунд часу щоб механізм зберіг. Важливо не виключати та перезавантажувать ПЗ під час обробки зображень.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Компоненти забезпечення користувача

Для виконання послідовних дій, які призведуть до ідентифікації, аналізу та зіставлення з подальшим покращенням якості, було створено додаток користувача під назвою Client.UI Розглянемо основні функції додатку на рисунку 22.

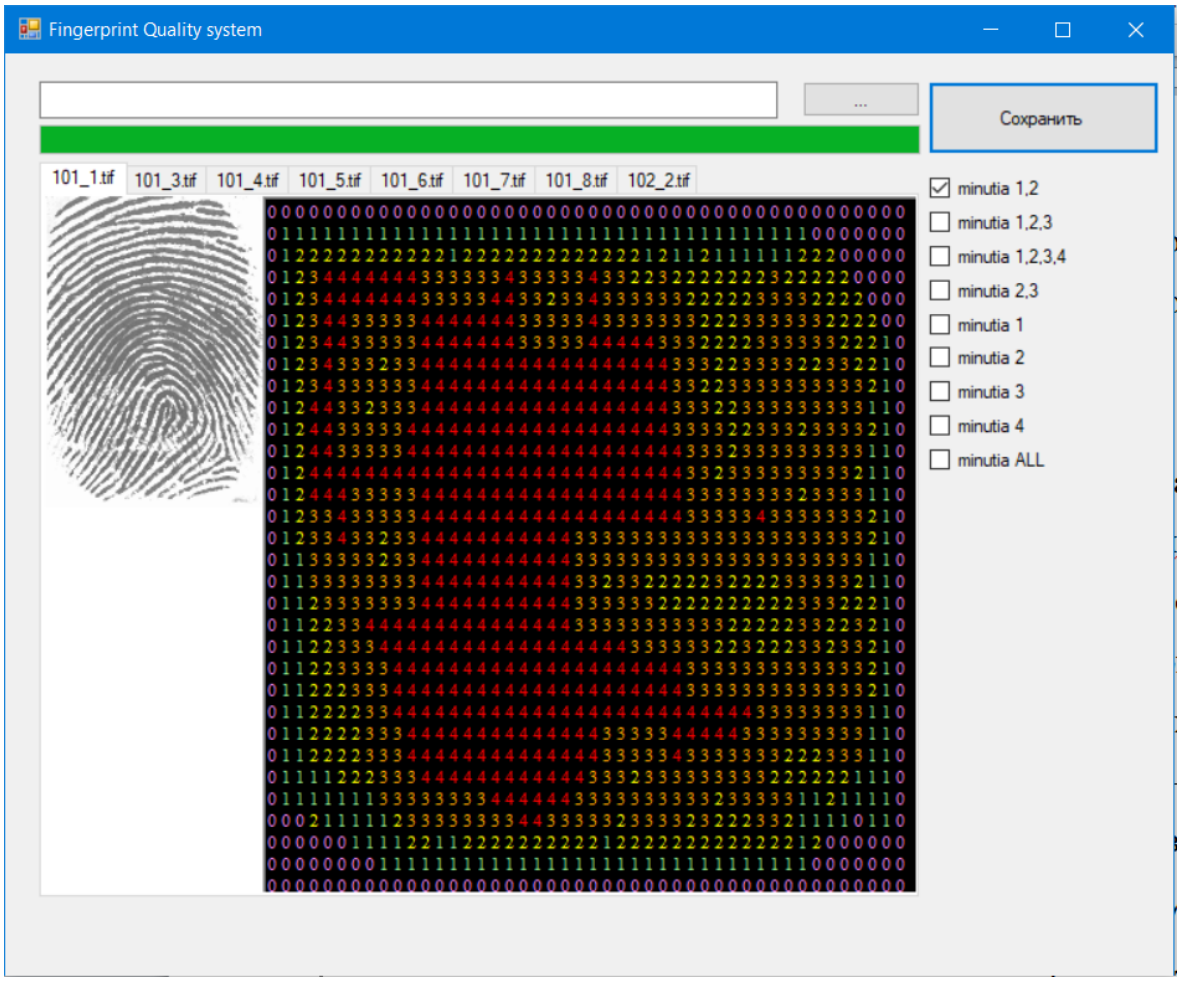


Рисунок 22 – Вікно користувацького програмного забезпечення

На рисунку 22 знаходится модуль вхідних данних OpenFileDialog з функцією вибора багатьох файлів одночасно. Після успішного додавання файлів зображення (*.tiff, *.bmp) до проеку, вони проходять першу стадію обробки – створення карт якості.

Створення карт якості реалізовано за допомогою метода

GetQualityMap класу FingerprintData, як структури, яка містіть у собі об’єкт зчитування відбитка та представлення його як об’єкт класу.

Після конвертацію зображення в об’єкт, ми використовуємо метод Nfiq.GetQualityMap, класу NFIQ, де в подальшому будуть використовуватися методи обробки. Далі кожний об’єкт зображення Bitmap передается параметром до методу CalculateQualityMap.

Цей метод містить функції знаходження мінуцій lfs_detect_minutiae_V2 та метод створення якостних карт gen_quality_map

Функція знаходження мінуцій розполагає створення файлів з вихідними данними:

```
PrintMap(direction_map, @"..\..\..\Results\direction_map.txt");
PrintMap(low_contrast_map, @"..\..\..\Results\low_contrast_map.txt");
PrintMap(low_flow_map, @"..\..\..\Results\low_flow_map.txt");
PrintMap(high_curve_map, @"..\..\..\Results\high_curve_map.txt");
```

Метод створення кінцевої карти визиває такі збереження файлів:

```
PrintMap(quality_map,
$@"..\..\..\Results\maps\quality_map_{name}.txt");
```

,де name – назва файлу збереження.

Сукупність цих операцій дає можливість наглядно проаналізувати поверхневий опис якості зображення.

Опис карт якості:

Кожна карта якості характеризується числом від 0 до 5.

В нашому случаї на рисунку 22, 4 – це погана якість, але не найгірша, 3,2,1 – аналогічно більш стійкіші одиниці виміру якості, за 0 варто ввжати області, які не входять до сукупності відбитку пальця. Кожний сегмент якості підсвічено аналогічним кольором для доступного розуміння де на карті знаходяться сектори с певною якістью. Однак, це лише візуалізація карти якості, в подальшому ці карті будут використовуватися для зіставлення за певними алгоритмами.

Для кожного відбитка в ПЗ создається окреме інтерактивне вікно для перегляду карт якості.

Справа в боковому меню відображається кнопка Сохранить та чекбокси з різноманітними вибірками мінуцій, при натиску на кнопку Сохранить ці відбитки подаються до наступного алгоритма ROC, а саме:

ROC.GetPositiveRates(paths, writer, selecttype), де

paths – це коллекція зображеннь

writer – вказівник на файл зображення

selecttype – це назва вибірки мінуції.

В данному случаї ми використовуємо алгоритм зіставлення відбитків пальців Minutia Cylinder-Code SDK [3]. Ці відбитки колекціонуються в пам’яті та передаються на конвеєр обробника, який зіставляє відбитки за допомогою методу Match, який містить метод вибранної бібліотеки MccSdk.MccSdk.MatchMccTemplates(template1, template2);

,де template1, template2 – це об’єкти порівняння між собою, цей метод знаходжуючі похибки несумісності FMR та FNMR, які потім в свою чергу передаються в обробник похибок для знаходження аппроксимованного значення EER.

Для обробки внутрішньої структуру для подачі на вихідні значення ми використовуємо метод:

GetFingerprintStructure – метод вертає структуру як масив мінуцій

IsGoodMinutia – метод повірки мінуції на її пригодність

ConvertMinutiae – метед конвертації мінуції як об’єкта класу в дані MCC SDK.

Кінцевим результатом роботи алгоритма є збереження на тіж самі місця зображення вже відредагованого за алгоритмом оптимізації якості на рисунку 23 зображено вихідний файл коригування якості зображення у вигляді текстового файлу.

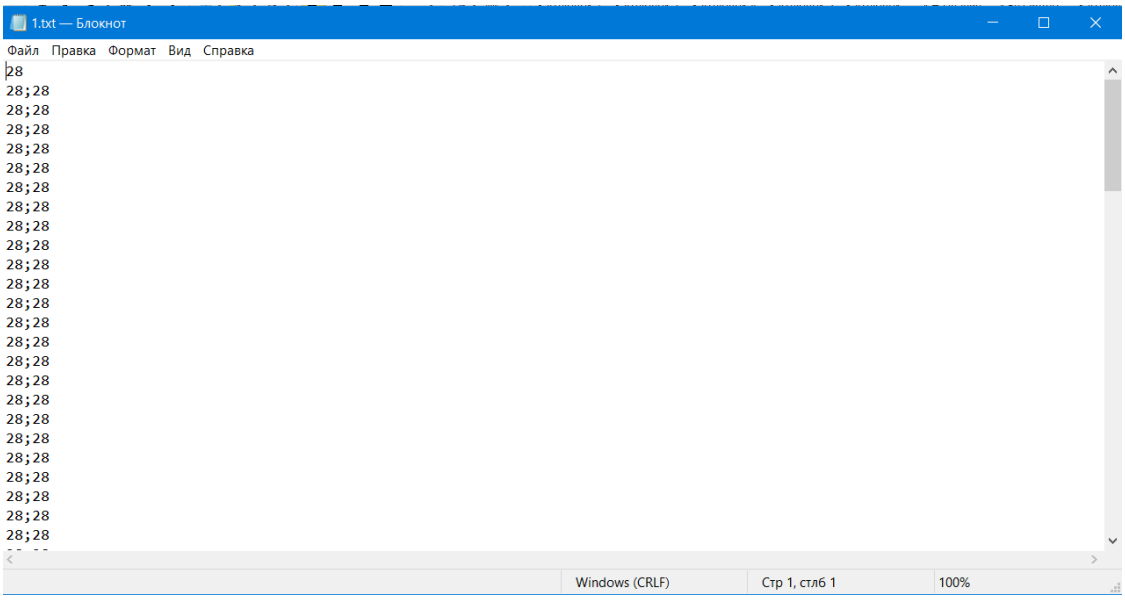


Рисунок 23 – Текстовий файл із вихідними данними

Після зіставлення та виправлення помилок, запустимо проект DrawResults та відштовхуючися від збереженних показів за допомогою рисунка 24 аналізуємо, що зіставлення пройшло успішно. Крива EER знаходится рівно посередині графіка, як повинно бути при максимальній якості зображення, DET графіки зіставлені на одній лінії що свідчить що робота та її вплив на покращення якості зображення є оптимальною да діючою. Помилки в роботі не знайдені, тестування продукту показало відмінний результат. Згідно за експериментами можно сказати що робота логічна та має місце для розгяду в якості алгоритму покращення якості відбитка та можно сказати що для роботи цифрових приладів та сканерів цей алгоритм задоволяє умови для використання у технологічній сфері.

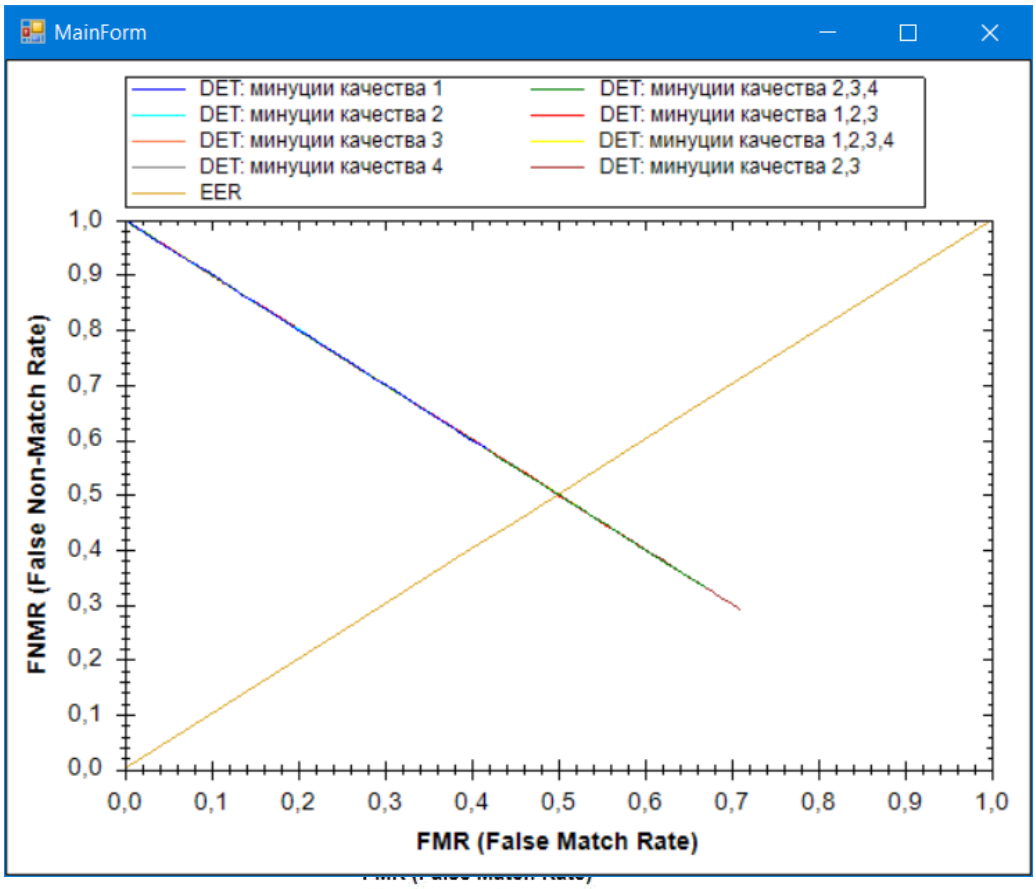


Рисунок 24 – Результат покращення якості відбитків пальців

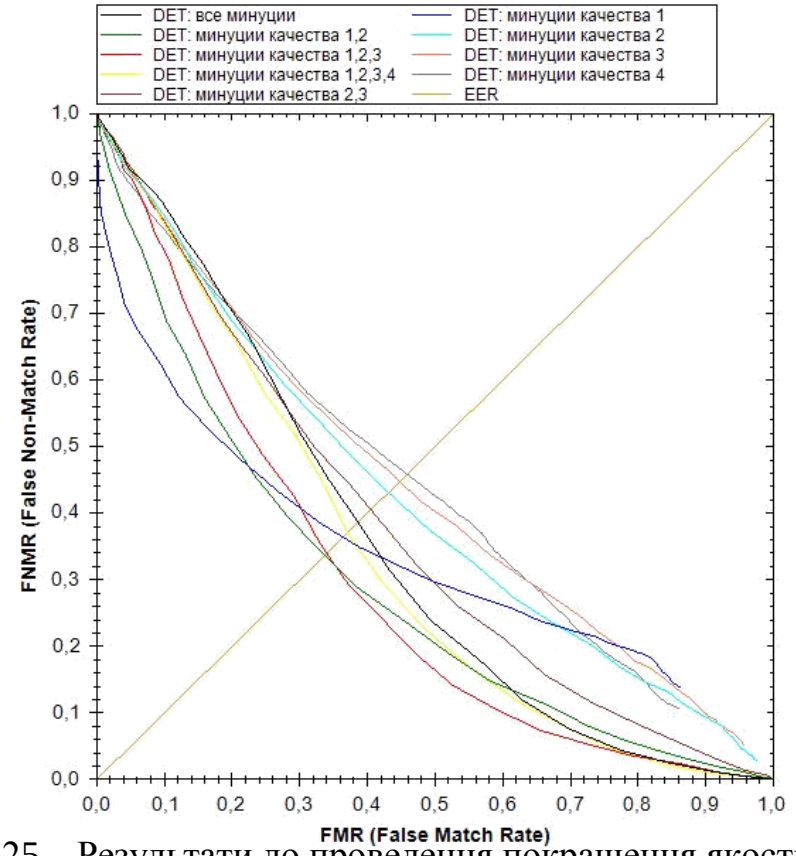


Рисунок 25 – Результати до проведення покращення якості

5.1 Рівні тестування програмного продукту

В цьому розділі описано тестування програмного додатку, яке покаже правильність виконання поставленого завдання та коректність роботи над даними. Тестування програмного забезпечення є невід'ємною частиною циклу розробки програмного продукту. Саме процес тестування забезпечує якість продукту, а також визначає готовність проекту до релізу.

Існує чотири основних рівні тестування програмного забезпечення: модульний(компонентний), інтеграційний, системний і приймальний.

Рисунок 26 – Схема рівнів тестування ПЗ

Модульний рівень тестування перевіряє функціональність і шукає дефекти в частинах програми, які доступні і можуть бути протестовані окремо (модулі програми, об’єкти, функції і т.д.) (рисунок 26).

Зазвичай компонентне (модульне) тестування проводиться шляхом введення коду, який необхідно перевірити, чи шляхом підтримки середовищ розробки, таких як фреймворки (каркаси) для модульного тестування або

інструменти для дебагу. Всі знайдені дефекти, як правило, виправляються в коді без формального їх опису в системі багів (Debugger)

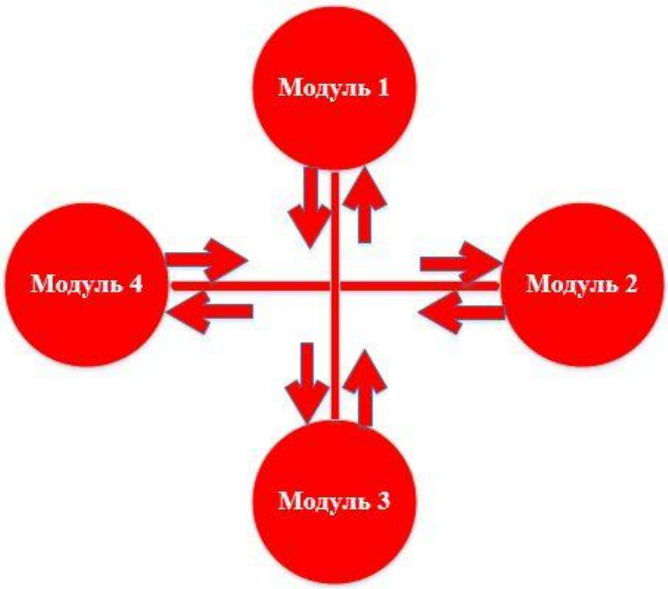


Рисунок 27 – Схема модульного розділення ПЗ

Інтеграційне тестування[15] – це фаза тестування програмного забезпечення, під час якої окремі модулі програми комбінуються та тестуються разом, у взаємодії. Інтеграційне тестування виконується після модульного тестування та перед верифікацією та валідацією ПЗ. Якщо розглядати цей процес як систему, то на вхід їй подаються модулі, які вже пройшли модульне тестування; потім модулі групуються в більші частини, виконуються тести передбачені планом, а на виході системи — інтегрована система, що готова до системного тестування.

Системний рівень тестування – це тестування програмного забезпечення, яке виконується на готовій, інтегрованій системі, з метою перевірки відповідності системи вихідним вимогам, як функціональним, нефункціональним, так і технічним. Системне тестування зазвичай проводять у два етапи: альфа- і бета- тестування.

Приймальне тестування проводиться з метою визначення, чи задовольняє система приймальним критеріям, потім йде винесення рішення замовником або іншою уповноваженою особою, утверджувати програму чи ні. Приймальне тестування виконується відповідно до Плану приймальних Робіт. Рішення про проведення приймального тестування приймається у випадку, коли продукт досягнув необхідного рівня якості, та замовник ознайомлений з Планом приймальних Робіт (Product Acceptance Plan) або іншим документом, де описаний набір дій, пов’язаних з проведенням приймального тестування, дата проведення, відповідальні і т.д.

ВИСНОВОК

В рамках даної роботи були отримані наступні результати.

1. Отримано карти якості зображень відбитків пальців за алгоритмом NFIQ.

2. Запропоновано метод визначення якісних областей відбитків.

3. Проведено ряд зіставлень відбитків пальців для різних вибірок Мінуцій згідно їх якості.

4. Обчислено помилки FMR, FNMR, EER і проведено порівняння результатів зіставлення відбитків.

Під час розробки мною пз я ознайомився з фізичними прототипами та математичними алгоритмами та зробив висновок, що моя реалізація не краще але й не поганіша аніж представлені флагмани на ринку дактилоспопії.

Також при розробці специфікації вимог я розподілив завдання по пріоритету і відносно цього почав розробляти програмне забезпечення. При розробці прототипу я зміг реалізувати всі заплановані рішення для гарного функціонування алгоритмів.

Для отримання інформації я вибрал віндоус форми та декларативний спосіб програмування.

В дипломному проекті було наведено приклади основних методів та полів, що забезпечують правильну роботу та реалізацію всіх функціональних особливостей додатку.

Отож, даний алгоритм є конкурентоспроможним, має всі шанси виходу на ринок, бути якісною програмою для надання біометричних послуг для вузькоспеціалізованих запитів. Тому поставлене мною завдання на дипломне проектування виконане успішно.

СПИСОК ОПРАЦЬОВАНИХ ДЖЕРЕЛ

1. R. Cappelli, M. Ferrara, D. Maltoni Minutia Cylinder-Code: a new representation and matching technique for fingerprint recognition // 2010.

2. Soweon Yoon, Eryun Liu, and Anil K. Jain On Latent Fingerprint Image Quality // 2012.

3. Biometric System Laboratory MCC Software Development Kit (SDK) Version 1.4 // 2014.

4. Інформаційні технології – засіб оптимізації діяльності підприємств [Електронний ресурс] / О. О. Байкарова, Л. М. Тарасюк // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. - 2013. - № 11. - С. 177-182. - Режим доступу: http://nbuv.gov.ua/UJRN/Kitonv_2013_11_32

5. *T.Y. Zhang, C.Y. Suen* A Fast Parallel Algorithm for Thinning Digital Patterns // 1984.

6. // Международный научный журнал // № 6, т. 2, 2016—
МЕТОД ПОЛІПШЕННЯ ЯКОСТІ ЗОБРАЖЕННЯ ВІДБИТКІВ ПАЛЬЦІВ ЗА ДОПОМОГОЮ ФІЛЬТРА ГАБОРА

7. Maio D., Maltoni D., Cappelli R., Wayman J. L., Jain A.K. FVC2000: Fingerprint verification competition // 2002.

8. Мультиспектральная технология в дактилоскопических сенсорах [Електронний ресурс]. — Алексей Гинце

9. Google Maps Вільна енциклопедія [Електронний ресурс]. — Режим доступу: http://uk.wikipedia.org/wiki/Google_Maps

10. Інформаційні системи і технології на підприємствах / Плескач В.Л. [Електронний ресурс]. — Режим доступу: http://pidruchniki.ws/1327010847729/informatika/strukturniy_pidhid_rozroblenny

11. Інформаційні системи і технології на підприємствах / Плескач В.Л. [Електронний ресурс]. — Режим доступу:

http://pidruchniki.ws/1707032647730/informatika/metodi_strukturnogo_analizu_proektuvannya

12. Технології компонентного програмування / Андрущак Олена [Електронний ресурс]. — Режим доступу: http://www.unicyb.kiev.ua/~boiko/it/_ref5k/andru_om/andru.htm

13. Поняття архітектури програмного забезпечення. [Електронний ресурс]. — Режим доступу: http://www.lib.mdpu.org.ua/e-book/web/Lec10.html

14. Рівні тестування [Електронний ресурс]. — Режим доступу: http://qalearning.com.ua/theory/lectures/material/testing-levels/

15. Особливості вимог програмного забезпечення [Електронний ресурс]. – Режим доступу: http://qalearning.com.ua/theory/lectures/material/requirements-testing-methods-equivalence/