Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Пивовар М. А.
(прізвище та ініціали)

Керівник: д. т. н., проф. Романюк О. Н.
(прізвище та ініціали)

Рецензент: к. т. н., доц. Колодний В. В.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Освітньо-кваліфікаційний рівень – магістр

Спеціальність 121 – інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

«___» _____ 2019 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пивовару Миколі Анатолійовичу

1. Тема роботи – розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування.

Керівник роботи: Романюк Олександр Никифорович, д. т. н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: кольоровий режим – TrueColor; метрологічні параметри голови дитини; розмір екрану – 1280x1024; вихідні методи для модифікації – метод Лоя для визначення осі симетрії обличчя людини; архів зображень облич дітей з генетичними захворюваннями; таблиці нормованих значень обхвату голови дитини залежно від віку.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз предметної галузі та постановка задачі розробки; розробка модифікованого методу визначення симетричності обличчя людини; розробка методів для проведення медичного експрес-діагностування; розробка та тестування програмного додатку

для медичного експрес-діагностування; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: галузі застосування зображень облич людей в медицині; метод Лоя для визначення осі симетрії обличчя людини; модель для діагностування генетичних захворювань; аналітичні залежності обхвату голови від віку; блок-схеми алгоритмів роботи розроблених методів; інтерфейс розробленого програмного додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Романюк О. Н., д.т.н., завідувач кафедри ПЗ		
5	Бальзан М. В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі розробки	04.09.2019 – 29.09.2019	Виконано
2	Розробка модифікованого методу визначення симетричності обличчя людини	30.09.2019 – 26.10.2019	Виконано
3	Розробка методів для проведення медичного експрес-діагностування	27.10.2019 – 3.11.2019	Виконано
4	Розробка та тестування програмного додатку для медичного експрес-діагностування	4.11.2019 – 12.11.2019	Виконано
5	Економічна частина	13.11.2019 – 17.11.2019	Виконано

Студент _____ Пивовар М. А.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Романюк О. Н.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі «Розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування» проведено аналіз методів використання графічних зображень для комп'ютерної діагностики.

Розроблено метод діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини, модифіковано метод Лоя для визначення осі симетрії обличчя людини, а також отримано аналітичні залежності обхвату голови дитини від її віку з урахуванням допустимого інтервалу.

У ході виконання роботи було проаналізовано існуючі методи морфологічного аналізу зображень облич людей та існуючі програмні аналоги.

Розроблено та протестовано роботу програмного додатку для медичного експрес-діагностування за допомогою таких засобів: мова програмування C#, Entity Framework, WPF та Emgu CV.

Розроблені в магістерській кваліфікаційній роботі методи, алгоритми та програмні засоби можуть бути використані в медицині для експрес-діагностування захворювань.

ABSTRACT

In master's qualification thesis on «Development of methods and software of human face images morphological analysis for medical express-diagnosis» the methods of using graphical images for computer diagnosis were analyzed.

Developed a genetic diseases diagnosis method based on analysis of visual characteristics of human face, modified Loy's method of human face symmetry detection and received analytical dependencies of child head circumference from his age with considering of the allowed interval.

In the course of work, existing methods of morphological analysis of human faces images and existing software analogues were analyzed. The software application for medical express-diagnosis was developed and tested by the following tools: C# programming language, Entity Framework, WPF and Emgu CV.

Methods, algorithms and software, developed in the master's qualification thesis, can be used in medicine for diseases express-diagnosis.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ 12
1.1. Використання інформаційних технологій в медицині 12
1.2. Симетрія обличчя людини в комп'ютерній діагностиці 15
1.3. Аналіз найпоширеніших алгоритмів визначення осі симетрії обличчя людини 17
1.4. Аналіз існуючих програмних аналогів 21
1.5. Постановка задачі 23
1.6. Висновки 24
2 РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ ВИЗНАЧЕННЯ СИМЕТРИЧНОСТІ ОБЛИЧЧЯ ЛЮДИНИ 25
2.1. Модифікація методу Г. Лоя для визначення осі симетрії обличчя людини 25
2.2. Розробка методу визначення симетричності обличчя людини на базі модифікованого методу визначення осі симетрії 29
2.3. Розробка алгоритму роботи модуля визначення симетричності обличчя людини 32
2.4. Висновки 34
3 РОЗРОБКА МЕТОДІВ ДЛЯ ПРОВЕДЕННЯ МЕДИЧНОГО ЕКСПРЕС-ДІАГНОСТУВАННЯ 35
3.1. Розробка методу діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини 35
3.2. Отримання аналітичних залежностей обхвату голови дитини з урахуванням вікових змін 41
3.3. Розробка алгоритмів для медичного експрес-діагностування 46
3.4. Висновки 50
4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ МЕДИЧНОГО ЕКСПРЕС-ДІАГНОСТУВАННЯ 51
4.1. Варіантний аналіз і обґрунтування вибору програмних засобів 51
4.2. Вибір середовища розробки 55
4.3. Розробка структури бази даних 57

4.4. Розробка інтерфейсу програми 60
4.5. Розробка програмних модулів системи 67
4.6. Вибір методики тестування 72
4.7. Тестування розробленого програмного додатку 74
4.8. Висновки 80
5 ЕКОНОМІЧНА ЧАСТИНА 81
5.1. Технологічний аудит розроблених методів і засобів морфологічного аналізу зображень облич людей 81
5.2. Розрахунок витрат на розробку методів і засобів морфологічного аналізу зображень облич людей 86
5.3. Розрахунок економічного ефекту від можливого впровадження розроблених методів і засобів морфологічного аналізу зображень облич людей 90
5.4. Висновки 96
ВИСНОВКИ 97
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 99
ДОДАТКИ 105
Додаток А. Технічне завдання 106
Додаток Б. Лістинг коду 110
Додаток В. Ілюстративний матеріал 129

ВСТУП

Обґрунтування вибору теми дослідження. В наш час дуже важливими є якість отримання та обробки інформації в усіх видах людської діяльності, в тому числі в медицині. Кожному медичному працівнику щохвилинно доводиться мати справу з великим об’ємом інформації, яка може бути представлена в текстовому, числовому, звуковому чи графічному вигляді. Тому, від ефективності процесів її отримання, збереження, передачі та обробки залежить якість та своєчасність проведення діагностичних, лікувальних та профілактичних заходів та робота системи охорони здоров’я в цілому [1]. Саме використання інформаційних технологій надає можливість підвищити швидкість та точність цих процесів.

Інформаційні технології в медицині можуть використовуватись для вирішення найрізноманітніших завдань: збереження, обробка та пошук інформації про пацієнтів; управління електронними чергами та електронним записом до фахівців; автоматизація підготовки призначень, рецептів, виписок та інших документів та багато інших [2]. Ще одним видом застосування інформаційних технологій в медицині є комп’ютерна діагностика.

Системи комп’ютерної діагностики (computer-aided diagnosis, CAD) – це системи, які допомагають лікарям обробляти медичні зображення, які можуть бути отримані за допомогою рентгену, магнітно-резонансної томографії, ультразвукової діагностики чи іншим способом [3]. Подібні системи аналізують цифрові зображення на наявність типових характеристик та помічають підозрілі області, на які варто звернути увагу лікарю під час діагностування пацієнтів.

Більшість систем комп’ютерної діагностики аналізує зображення, які можна отримати тільки за допомогою спеціального медичного обладнання та які певним чином відображають організм людини зсередини, але і зовнішні ознаки людини також можуть бути дуже інформативними.

Наприклад, використовуючи зображення обличчя людини, можна діагностувати понад 700 хвороб, які мають генетичну природу, з точністю близькою до 90% [4].

Симетричність та асиметричність обличчя також можна визначити по зображенню людини, а вона може бути симптомом багатьох хвороб, пов’язаних з аномаліями лицевого нерву та кісток черепа, порушенням мімічних м’язів та інших [5].

Таким чином, можна зробити висновок, що морфологічний аналіз зображень облич людини може бути дуже інформативним для проведення діагностування перед оглядом у лікаря. Але ця тема є досить новою, а методи та способи її реалізації ще не були достатньо розвинені, тому питання розробки методів та програмних засобів для морфологічного аналізу зображень облич людей є актуальним.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності медичного експрес-діагностування за рахунок розробки нових та модифікації існуючих методів та засобів морфологічного аналізу зображень облич людей.

Основними задачами дослідження є:

- провести аналіз існуючих методів та засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування з метою пошуку способів підвищення їх ефективності;
- запропонувати нові:
    - методи підвищення продуктивності визначення осі симетрії обличчя людини;
    - методи підвищення швидкості визначення типу генетичних захворювань на основі аналізу візуальних характеристик обличчя людини;
- отримати аналітичні залежності обхвату голови дитини від її віку з урахуванням допустимого інтервалу;
- розробити програмний додаток на основі запропонованих методів;
- провести тестування розробленого програмного додатку.

Об’єкт дослідження – процес морфологічного аналізу зображень облич людей в медичному експрес-діагностуванні.

Предмет дослідження – методи та засоби аналізу 2D-зображень.

Методи дослідження. У процесі досліджень використовувались: теорія чисел, теорія ймовірності та статистики для оцінки рівня симетричності, теорія комп’ютерного зору для пошуку ключових точок та характерних точок, комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Вперше розроблено метод діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини, особливість якого полягає у визначені відстаней між встановленими характерними точками обличчя людини відповідно до захворювання, що дозволило підвищити швидкість визначення типу генетичних захворювань.

2. Подальшого розвитку отримав метод Лоя для визначення осі симетрії обличчя людини, який відрізняється від існуючого заміною методу визначення ключових точок SIFT на метод ORB, що дозволило підвищити швидкість та точність процесу пошуку осі симетрії.

3. Вперше отримано аналітичні залежності обхвату голови дитини від її віку з урахуванням допустимого інтервалу, що дає можливість розробки програмного забезпечення для діагностування за допомогою порівняння параметрів голови дитини з допустимими нормами.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено методи та програмний засіб для морфологічного аналізу зображень облич людини, які можуть бути використані в медичних установах для проведення ефективного медичного експрес-діагностування ряду захворювань.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботах,

опублікованих у співавторстві, автору належать такі результати: аналіз візуальних характеристик обличчя людини для діагностування генетичних захворювань [4]; проведення аналізу найвідоміших методів для визначення осі симетрії [5]; розробка модифікованого методу для визначення симетричності обличчя людини [6].

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях:

- XII міжнародна науково-практична конференція «Інформаційні технології і автоматизація – 2019» (Одеса, 2019);
- II Всеукраїнська науково-технічна конференція «Комп'ютерні технології: інновації, проблеми, рішення» (Житомир, 2019);
- Міжнародна науково-практична конференція «Електронні інформаційні ресурси в освіті і науці: створення, використання, доступ». (Вінниця, 2019).

Публікації. Основні результати досліджень опубліковано в 3 наукових працях, у тому числі 3 – у матеріалах конференцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ

1.1. Використання інформаційних технологій в медицині

За останні декілька десятиліть спостерігається значний стрибок у розвитку медицини, однією з причин якого є її зв'язок інформаційними технологіями. Використання інформаційних технологій в медицині продиктовано зростаючими об'ємами інформації, її достовірності, способами отримання та передачі. Процес комп'ютеризації медицини з кожним роком набирає все більших темпів, а з цим зростає кількість медичних закладів, які впроваджують медичні інформаційні системи [6]. Сучасні інформаційні технології, в тому числі і в сфері медицини, допомагають раціоналізувати працю медичних працівників, підвищити ефективність лікувально-профілактичних закладів та вивести системи охорони здоров'я на новий рівень як по об'ємам послуг, так і по якості. Однією з задач медицини, яку допомагають вирішувати інформаційні технології, є медичне діагностування.

Медичне діагностування – це процес встановлення діагнозу, тобто заключення про сутність хвороби і стан пацієнта [7]. Діагностування, як правило, потребує збору історії захворювань хворого та його обстеження. Також, часто воно супроводжується однією або кількома діагностичними процедурами, які називають медичним тестуванням.

Діагностування є складним процесом, оскільки багато ознак та симптомів є неспецифічними, тобто можуть притаманними декільком захворюванням одночасно. Тому, медичне діагностування потребує проведення диференціального аналізу, який передбачає пошук декількох можливих причин симптомів, їх порівняння та аналіз та визначення найоптимальнішого варіанту.

Для вирішення завдань медичного діагностування, в наш час широко використовують відповідне допоміжне програмне забезпечення. Лікар взаємодіє з програмним забезпеченням, використовуючи його та свої власні знання для

кращого аналізу даних пацієнта, ніж на це здатна людина чи програма самостійно.

Використовуючи відповідні програми, лікар може віддалено провести експрес-діагностування пацієнта та встановити йому попередній діагноз по його зовнішнім характеристикам. Наприклад, черепно-лицьові характеристики є дуже інформативними для клінічних генетиків при діагностуванні генетичних захворювань. Рідкісними генетичними захворюваннями страждають близько 8% людей, для значної частини яких характерними є зміни в обличчі та черепі [8].

Багатьом генетичним захворюванням властиві досить характерні ознаки, які можна помітити на зображенні обличчя пацієнта. Розглянемо деякі з них.

Ознаками синдрому Корнелі де Ланге є маленький ніс, вигнуті брови та нетиповий рот [9] (рис. 1.1).

Рисунок 1.1 – Дитина з синдромом Корнелі де Ланге

Загальними риси обличчя пацієнтів з мутацією CHD8 є: макроцефалія, гіпертелоризм, щільна пальцебральна щілина внизу, широкий ніс та загострене підборіддя [10].

Типовими рисами синдрому Дауна є: сплюснуте лице, особливо перенісся; мигдалеподібні очі, які нахилені вгору; коротка шия, маленькі вуха та крихітні білі плями на райдужній оболонці ока [11].

Синдрому Нунан (рис. 1.2) властиві такі ознаки: важкі повіки, які можуть заважати зору, епікантові очні складки, широко розставлені очі зі сплющеним

переніссям, сильно вигнути ромбовидні брови, низько поставлені вуха, низька лінія волосся, коротка шия [12].

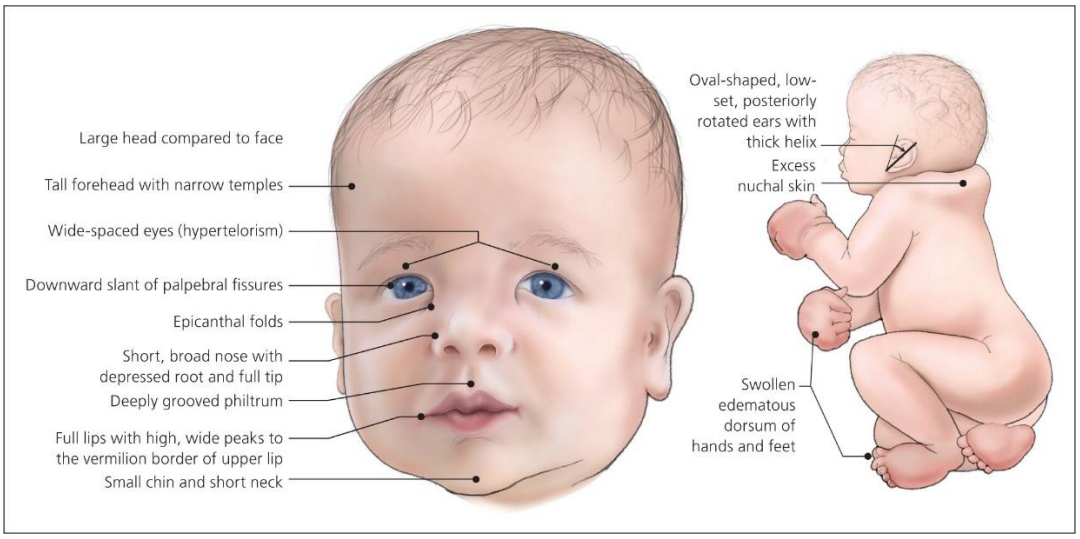


Рисунок 1.2 – Характерні риси синдрому Нунан

Для синдрому Ді Джорджі (рис. 1.3) властиві такі риси обличчя: маленька голова, короткий ніс, низьке перенісся, недорозвинена щелепа, плоска середня частина обличчя та тонка верхня губа [13].

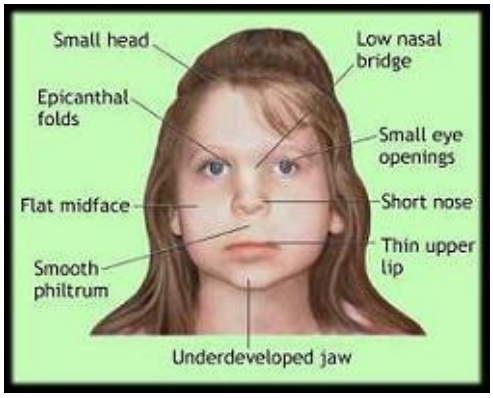


Рисунок 1.3 – Характерні риси синдрому Ді Джорджі

Люди з синдромом PACS1 мають характерний зовнішній вигляд: густі та сильно вигнуті брови, довгі вії, широко розставлені очі, опуклі повіки, заокруглений ніс, широкий рот з куточками, які спрямовані вниз, тонка верхня

губа, гладка область між носом і верхньою губою, широко та низько поставлені вуха, з меншою складкою, ніж зазвичай [14].

Також, по фотографії голови людини зверху можна провести обрахунок обхвату голови. Його відхилення від форми може свідчити про наявність певних захворювань, таких як мікроцефалія та макроцефалія [15].

Аналізуючи все вищеперераховане, можна зробити висновок, що можливим є проведення медичного експрес-діагностування шляхом морфологічного аналізу зображення людини. Тому, доцільно розробити методи, які будуть надавати можливість проводити подібний аналіз.

1.2. Симетрія обличчя людини в комп'ютерній діагностиці

Обличчя виконує важливу роль в процесі міжособистісного спілкування і є об'єктом дослідження діячів мистецтва, анатомів, психологів та представників медицини [16]. Важливу роль в характеристиці обличчя відіграє ступінь його симетричності/асиметричності.

Асиметрія обличчя – це відмінності морфофункціональних характеристик лівої та правої сторін обличчя [17]. Асиметрія обличчя є важливим фактором індивідуальної краси та може бути пов'язана з відмінністю тонусу м'язів обличчя, який зумовлений генетичним фактором, змінами м'яких тканин, судин обличчя, а також різноманітними аспектами іннервації: впливом м'язової системи на кісткові утворення, від яких починаються та до яких прикріпляються м'язи, що призводить до їх зміни, яка особливо помітна при тривалому впливі.

Дослідження показують, що симетрія обличчя пов'язана з моделлю «великої п'ятірки» особистості [18]. Найбільш однозначні положення стверджують, що симетрія обличчя позитивно пов'язана з рівнем екстравертності особистості, тобто люди з більш симетричними обличчями є більшими екстравертами.

Дослідження, проведене Б. Фінком [19] показало, що людям з низьким ступенем асиметрії обличчя, оточуючі частіше присвоюють позитивні якості, ніж людям з сильно несиметричним обличчям (рис. 1.4).

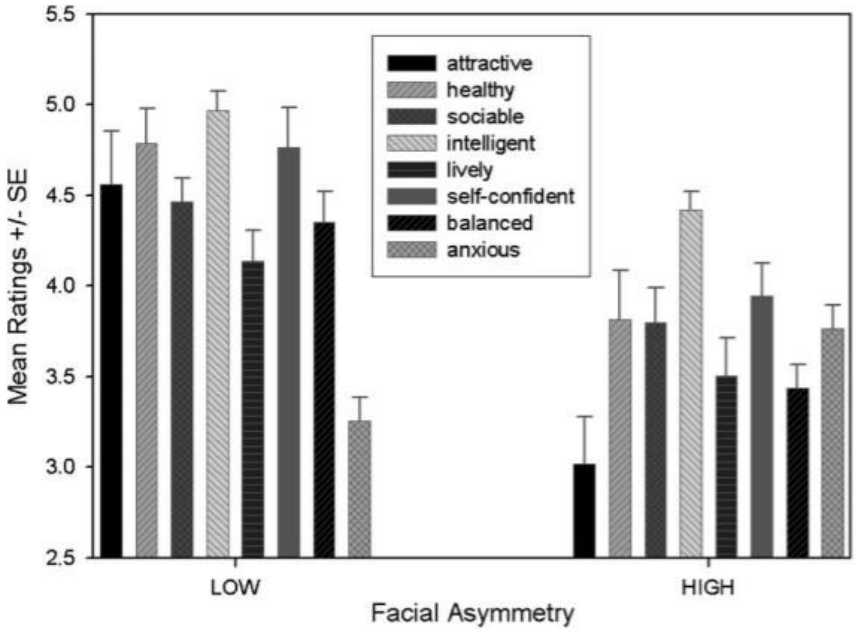


Рисунок 1.4 – Оцінки якостей людини в залежності від рівня асиметрії обличчя

Сила нервової системи людини позитивно корелює з щелеповим індексом по горизонталі. Цей індекс показує відношення ширини лівої сторони обличчя по лінії щелепи до правої сторони. Чим більша ліва сторона обличчя по лінії щелепи по відношенню до правої, тим сильніша нервова система по правій півкулі і навпаки [16].

Переважання лівої сторони обличчя по лінії нижньої щелепи свідчить про пониження тонусу як парасимпатичного відділу вищої нервової діяльності (ВНД), так і симпатичного. Відповідно, переважання правої половини обличчя свідчить про підвищенні тонусу обох відділів вегетативної нервової системи. Якщо враховувати той факт, що людина знаходиться в нормі у випадку балансу тонусів цих відділів, то про наявність такого балансу свідчить відносна симетрія обличчя по лінії нижньої щелепи [16].

Черепний індекс по вертикалі позитивно корелює з задоволеністю життям та меланхолічними почуттями. З цього випливає, що люди з «дитячою схемою» обличчя демонструють задоволеність життям в цілому, процесом самореалізації, здатні долати складнощі на шляху реалізації своїх здібностей [20].

Про явне переважання симпатичного тонусу свідчить кореляція положення нижнього повіка відносно райдужки ока. Чим нижче знаходиться нижнє повіко

відносно нього, тим яскравіше відображається тонус симпатичного відділу вегетативної нервової системи порівняно з парасимпатичною [16].

Таким чином, можна зробити висновок, що симетрія обличчя людини є важливим фактором при діагностуванні ряду захворювань, тому, було прийнято рішення реалізувати в розроблюваному програмному додатку метод визначення симетричності обличчя людини.

1.3. Аналіз найпоширеніших алгоритмів визначення осі симетрії обличчя людини

Під час аналізу бінарних зображень у задачах комп'ютерного діагностування важливу роль відіграє дзеркальна (осьова) симетрія. Очевидно, що реальні зображення обличчя практично ніколи не бувають ідеально симетричні, тому виникає задача високопродуктивного та ефективного пошуку наближеної симетрії з подальшою оцінкою ступеня симетричності зображення.

Розглянемо декілька найпоширеніших методів визначення осі симетрії обличчя людини.

Епіфанцев Б. Н. та Архіпов А. А. [21] пропонують метод пошуку осі симетрії обличчя людини на базі ключових точок, який є оптимізацією алгоритму, запропонованого Й. Лі та К. Шмідтом [22]. Згідно нього, спочатку, виконується пошук ключових точок, а саме – зовнішніх і внутрішніх кутів очей людини та підносового жолобка (philtrum) (рис. 1.5).

Рисунок 1.5 – Ключові точки для визначення осі симетрії

Потім, через внутрішні ключові точки проводиться пряма і виконується операція нормалізації зображення відносно нахилу голови шляхом повороту зображення на кут α, який є кутом між прямою між внутрішніми ключовими точками та віссю абсцис. Після цього, через середину відрізку, який з'єднує ключові точки, проводиться попередня лінія симетрії, для якої обраховується коефіцієнт симетрії за формулою 1.1.

$$r = \frac{\sum_{\mu=1}^{M}\sum_{\pi=1}^{N}[B_л(-x_\mu, y_\pi)-\overline{B}_л][B_п(x_\mu, y_\pi)-\overline{B}_п]}{\sqrt{\left\{\sum_{\mu=1}^{M}\sum_{\pi=1}^{N}[B_л(-x_\mu, y_\pi)-\overline{B}_л]^2\right\}\left\{\sum_{\mu}\sum_{\pi}[B_п(x_\mu, y_\pi)-\overline{B}_п]^2\right\}}}, \quad (1.1)$$

де $B_л((-x_\mu, y_\pi)$, $B_п((-x_\mu, y_\pi)$ – матриці значень інтенсивності пікселів лівої та правої частин зображення лиця; $M = x_к/\Delta x$, $N = 3x_к/\Delta y$ – кількість пікселів на кожній з них; Δx, Δy – інтервали дискретизації по відповідним осям; $\overline{B}_л$, $\overline{B}_п$ – середнє значення інтенсивності порівнюваних зображень.

Після цього, початкова лінія симетрії поступово зміщується на *k* пікселів у кожну сторону і при кожному зміщенні обраховується поточний коефіцієнт симетрії. Максимальне значення *r* буде відповідати знайденій осі симетрії зображення. Швидкодія цього алгоритму залежить від розмірів інтервалів дискретизації.

Алгоритм визначення симетрії, запропонований Журавською О. В. [23], передбачає виконання таких дій. На вхід приймається послідовність чисел, які описують контур фігури. На виході отримують координати точки на контурі, яка знаходиться на одній з можливих осей симетрії, та кут нахилу цієї осі до осі абсцис.

Для кожної точки контуру обчислюється коефіцієнт дискретного перетворення Фур'є (ДПФ) за формулою 1.2 (l – номер пікселя, p – кількість позицій, на яку зсувається контур):

$$f_l^p = \exp\left(i * \frac{2\pi}{N} * l * p\right), \; l = \overline{0, N-1}. \quad (1.2)$$

Після цього обраховується кут нахилу осі симетрії та величина середнього квадратичного відхилення коефіцієнтів Фур'є від цієї осі за формулою 1.3.

$$Q(p) = \sqrt{\frac{\sum_{l=1}^{N-1} Im(f_l^p * \exp(-i\alpha^p))^2}{N-1}}. \quad (1.3)$$

Після цього аналізується значення Q(p). Чим ближче воно до нуля, тим більше задана пряма відповідає вісі симетрії.

Цей алгоритм має складність $O(N^2)$, тобто є квадратичним до кількості точок в контурі фігури. Він дозволяє отримувати точний дескриптор Фур'є за рахунок постійної кількості точок для кроку (1-2 пікселі), але має низьку швидкість роботи для контурів з великою кількістю точок і може зазнавати впливу шуму на границях фігури.

Гарет Лой [24] пропонує алгоритм визначення симетрії, який використовує ряд ключових точок на обличчі людини.

Спочатку, за допомогою методу виявлення ознак SIFT (scale-invariant feature transform) [25], визначаються локальні ключові точки зображення, які називають дескрипторами. Вони описуються координатами, положенням та інколи масштабом. Після цього, виконується нормалізація цих точок і для отриманих значень генерується масив віддзеркалених дескрипторів відносно певної осі, наприклад, осі ординат, які співвідносяться з початковим масивом дескрипторів.

Потім, для кожної пари точок обраховується рівень симетрії, оцінка масштабу та відстані між точками. Рівень симетрії кожної пари точок описується як $\Phi_{ij} \in [-1, 1]$, значення якого обраховується за формулою 1.4.

$$\Phi_{ij} = 1 - \cos(\varphi_i + \varphi_j - 2\theta_{ij}). \quad (1.4)$$

Оцінка масштабу описується як $S_{ij} \in [0, 1]$ і обраховується за формулою 1.5.

$$S_{ij} = \exp\left(\frac{-|s_i - s_j|}{\sigma_s(s_i + s_j)}\right)^2. \quad (1.5)$$

У цій формулі σ_s є коефіцієнтом, який контролює допустимий ступінь варіації масштабу.

Далі проводиться обрахунок оцінка Гаусівської відстані, яка описується як $D_{ij} \in [0, 1]$ і визначається за формулою 1.6.

$$D_{ij} = exp\left(\frac{-d^2}{2\sigma_d{}^2}\right). \quad (1.6)$$

Усі отримані значення комбінуються для отримання значення величини симетрії (symmetry magnitude) для кожної пари точок за формулою 1.7.

$$M_{ij} = \begin{cases} \Phi_{ij} S_{ij} D_{ij}, & \Phi_{ij} > 0, \\ 0, & \Phi_{ij} \leq 0. \end{cases} \quad (1.7)$$

Кожна пара точок є потенційною віссю симетрії, яка перпендикулярно проходить через середину відрізку, який з'єднує ці точки, як показано на рисунку 1.6.

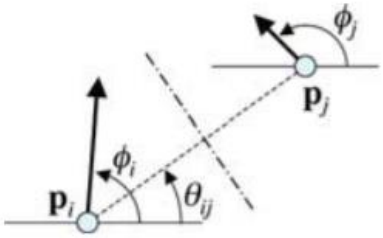


Рисунок 1.6 – Пара точок, отримана в результаті віддзеркалення

Цю пряму можна описати в полярних координатах за формулою 1.8.

$$r_{ij} = x_c cos\theta_{ij} + y_c sin\theta_{ij} \quad (1.8)$$

де (x_c,y_c) – це координати середини лінії, яка об'єднує цю пару точок, а θ_{ij} – це кут, який вона створює з віссю OX.

Далі, використовується лінійне перетворення Хафмана для виявлення домінантної осі симетрії. Кожна пара симетричних точок аналізується в просторі Хафмана, зваженому по рівню її величини симетрії (M_{ij}). Отриманий простір значень фільтрується за допомогою Гаусівського згладжування, в результаті чого знаходиться максимум, який описує шукану вісь симетрії.

Алгоритм Г. Лоя є найшвидшим та найбільш точним з розглянутих, тим не менш, він має простір для покращення. Його швидкодія та точність сильно залежать від методу визначення ключових точок. В цьому алгоритмі пропонується використовувати метод SIFT, але він є застарілим, тому було прийнято рішення модифікувати його, шляхом використання більш ефективного алгоритму визначення ключових точок зображення.

1.4. Аналіз існуючих програмних аналогів

Більшість з існуючих програм для комп'ютерного діагностування представляють собою програмні інтерфейси для роботи з зображеннями. Розглянемо декілька найбільш відомих аналогів.

«SuperKluge» – програмний додаток, призначений для роботи з МРТ зображеннями, розроблений М. Вільяненом, Й. Феном та Н. Йонгчіном [26] (рис. 1.7).

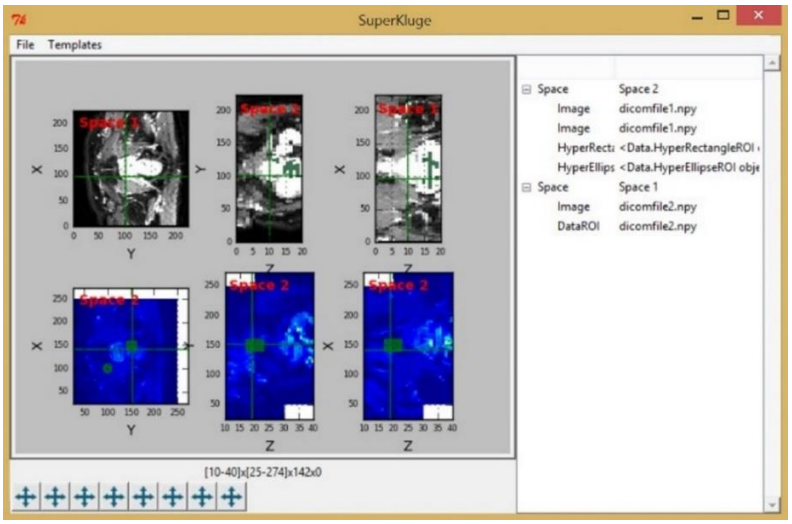


Рисунок 1.7 – Інтерфейс програми «SuperKluge»

Він дозволяє відображати 2D проекції одного зображення в декількох вікнах, що надає можливість лікарю переглянути та проаналізувати важливі для нього області зображення в пошуках патологій. Недоліком цього програмного засобу є обмежений функціонал по оперуванню з обраним зображенням.

Програмний додаток, розроблений Г. Карнамом, представляє собою програмний засіб для комп'ютерного діагностування раку головного мозку на основі МРТ зображення [27] (рис. 1.8).

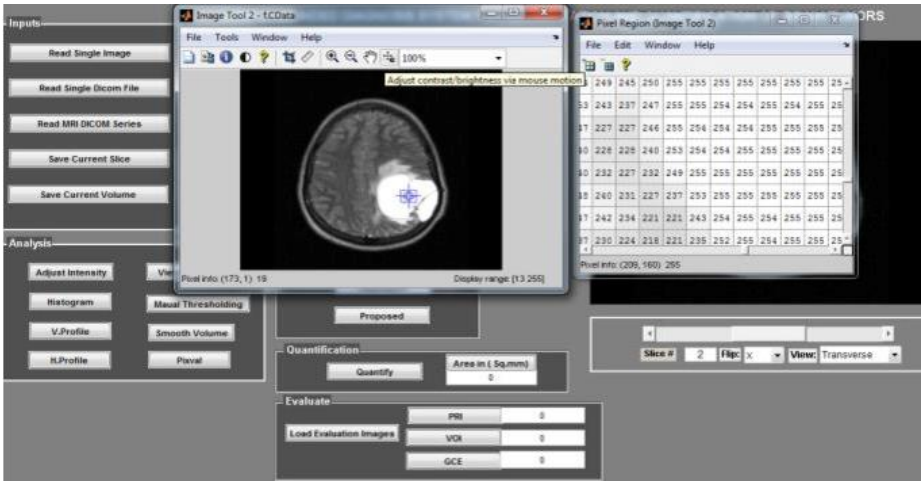


Рисунок 1.8 – Інтерфейс програми, розробленої Г. Карнамом

У цій програмі реалізовано ряд функцій по роботі з зображенням, таких як масштабування, порівняння подібності областей, відображення значення пікселів у виділеній області, зміна розміру зображення та ряд інших. Недоліком цього програмного засобу є відсутність можливості виміряти точні розміри досліджуваних областей зображення.

«Face2Gene» – це програмний додаток, розроблений компанією FDNA, який призначений для діагностування генетичних захворювань на базі зображень обличчя людини [28] (рис. 1.9).

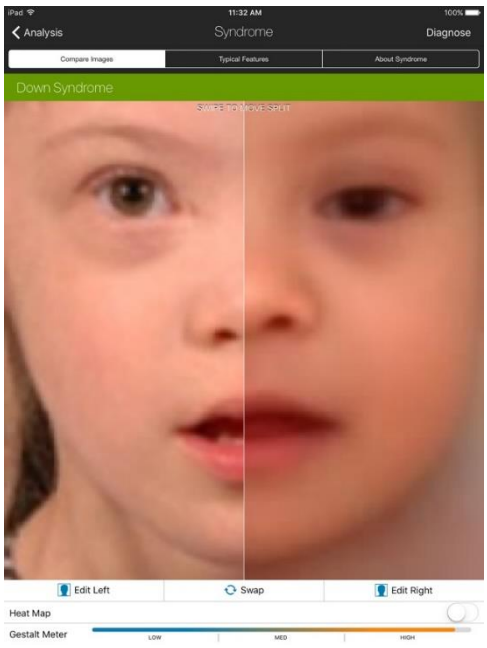


Рисунок 1.9 – Інтерфейс програми «Face2Gene»

Ця програма використовує системи комп’ютерного розпізнавання об’єктів та алгоритми глибинного навчання (deep learning) для встановлення характерних ознак генетичних захворювань та їх пошуку на зображенні обличчя людини. Недоліком цієї програми є відсутність можливості виміру рівня подібності окремих областей зображення та відсутність автоматичного пошуку осі симетрії.

Таким чином, аналіз існуючих програм для комп’ютерного діагностування показав, що їм властиві певні недоліки, тому, доцільно розробити власний програмний продукт, який буде об’єднувати їх переваги, усувати наявні недоліки та реалізовувати розроблені методи.

1.5. Постановка задачі

Метою магістерської кваліфікаційної роботи є підвищення ефективності проведення медичного експрес-діагностування за рахунок морфологічного аналізу зображень облич людини.

Проведений аналіз показав, що для її досягнення необхідно вирішити наступні задачі:

1. Провести аналіз існуючих методів визначення ключових точок.
2. Модифікувати метод Лоя для визначення осі симетрії обличчя людини.

3. Розробити метод визначення симетричності обличчя людини на основі модифікованого методу Лоя.

4. Розробити метод діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини

5. Отримати аналітичні залежності обхвату голови дитини з урахуванням допустимого інтервалу, та на їх основі розробити модуль для проведення медичного експрес-діагностування.

6. Розробити алгоритм роботи програми.

7. Обґрунтувати вибір програмних засобів та середовища розробки.

8. Розробити алгоритми роботи програмних модулів, які реалізовують розроблені методи.

9. Розробити структуру бази даних.

10. Реалізувати розроблені методи за допомогою обраних засобів.

11. Провести тестування розроблених модулів з метою виявлення та усунення можливих помилок.

1.6. Висновки

У результаті проведеного аналізу предметної галузі було зроблено висновок про можливість проведення медичного експрес-діагностування шляхом морфологічного аналізу зображень обличчя людини, який потребує створення відповідного методу. Також, було підтверджено важливість симетрії в процесі діагностування та необхідність модифікації існуючого алгоритму визначення осі симетрії обличчя людини.

Було проведено аналіз існуючих аналогів для проведення медичного експрес-діагностування, який показав, що їм властиві певні недоліки і тому було прийнято рішення розробити власний програмний додаток, який буде їх усувати, а також реалізовувати розроблені методи. Також, було проведено постановку основних задач, які необхідні для досягнення поставленої мети.

2 РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ ВИЗНАЧЕННЯ СИМЕТРИЧНОСТІ ОБЛИЧЧЯ ЛЮДИНИ

2.1. Модифікація методу Г. Лоя для визначення осі симетрії обличчя людини

Для того, щоб обрати який метод визначення ключових точок доцільно використовувати для модифікації існуючого методу визначення осі симетрії обличчя людини, доцільно розглянути його існуючі аналоги. Найбільш відомими методами визначення ключових точок є: SIFT, KAZE, AKAZE, SURF. ORB та BRISK.

Масштабовано-інваріантна трансформація ознак (SIFT) – це метод визначення ключових точок, який був запропонований Д. Лоу в 2004 році, який базується на алгоритмі різниці по Гаусу (Difference-on-Gaussians, DoG) [29].

Цей метод складається з наступних кроків [30]:

1. Пошук екстремумів масштабованого простору, отриманих за допомогою розмиття зображень по Гаусу.
2. Локалізація ключових точок.
3. Інтерполяція суміжних даних для підвищення точності визначення положень точок.
4. Відкидання точок з низьким контрастом.
5. Вилучення впливу ребер для підвищення стабільності.
6. Присвоєння орієнтації.
7. Формування дескрипторів ключових точок.

Метод SIFT є дуже стійким до повороту, масштабування та незначних афінних перетворень зображення, але потребує значних обчислювальних витрат.

Метод прискорених стійких ознак (Speeded Up Robust Features, SURF) був запропонований Г. Беєм в 2008 році як модифікація методу SIFT [31].

SURF використовує фільтри квадратної форми для апроксимації Гаусівського згладжування [32], в той час як SIFT використовує каскадні фільтри для виявлення незалежних від масштабу ключових точок, що потребує

постійного обрахунку Гаусівської різниці для кожного з масштабованих зображень. Використання фільтрів квадратної форми дозволяє значно підвищити швидкість виконання, оскільки обрахунки виконуються тільки на кутах зображення, а не кожному пікселі [31].

Перевагою цього методу, порівняно з SIFT, є значно вища швидкодія, але є один недолік – нижча точність при незначних афінних перетвореннях.

Метод ознак KAZE – це метод пошуку та опису ознак, розроблений П. Алькантарілья, Е. Бартолі та Е. Девідсоном у 2012 році, який знаходиться у вільному доступі та має відкритий вихідний код [33]. Він використовує нелінійний простір масштабування через нелінійне дифузійне фільтрування [34], що робить розмиття зображень локально адаптивним для ключових точок, в результаті чого зменшується кількість шумів на границях досліджуваного зображення. KAZE детектор базується на масштабовано-нормалізованому детермінанті матриці Гессе [35], яка обраховується на декількох рівнях масштабування зображення.

Цей метод не зазнає впливу повороту та масштабування зображення та є більш точним для зображень різного масштабу, але має низьку швидкодію.

Метод AKAZE є модифікацію методу KAZE за допомогою нового фреймворку Fast Explicit Diffusion (FED), який дозволив значно підвищити його швидкодію у порівнянні з KAZE [35].

Метод ORB (Oriented FAST and Rotated BRIEF) є поєднанням методу виявлення ознак FAST (Features from Accelerated Segment Test) та нормалізованого методу опису BRIEF (Binary Robust Independent Elementary Features), який був представлений І. Ріблі в 2011 році [36].

Цей метод складається з наступних кроків [37]:

1. Виконується пошук ключових точок за допомогою деревовидного FAST алгоритму на базовому зображенні та декількох зображеннях з піраміди зменшених зображень.

2. Обраховується міра Харріса [38] для отриманих точок. Точки, для яких отримане значення є низьким, відкидаються.

3. Обраховується кут орієнтації ключової точки.

Для цього, виконується обрахунок моментів яскравості навколо ключової точки за формулою 2.1.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y)\,, \tag{2.1}$$

де x, y – піксельні координати, I – яскравість.

Після цього обчислюється кут орієнтації ключової точки за формулою 2.2.

$$\theta = atan2(m_{01}, m_{10})\,. \tag{2.2}$$

Отримане значення називають «центроїдом орієнтації», яке описує напрям для області навколо ключової точки.

4. Використовуючи отриманий кут орієнтації, виконується поворот послідовності точок для бінарних порівнянь в дескрипторі BRIEF згідно цього кута.

Нові координати точок обраховуються за формулою 2.3.

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = R(\theta) * \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{2.3}$$

5. По отриманим точкам обраховується бінарний дескриптор BRIEF.

Цей метод має високу швидкодію та точність і є інваріантним для масштабування та обертання.

Метод BRISK (Binary Robust Invariant Scalable Keypoints) був представлений С. Лейтенгером в 2011 році та використовує метод AGAST для пошуку кутів зображення та фільтрує їх за допомогою FAST Corner score методу для пошуку максимумів в піраміді масштабованих зображень [39]. Опис BRISK базується на визначенні напрямку кожної ключової точки для досягнення інваріантності афінних перетворень.

Для того, щоб обрати який метод визначення ключових точок доцільно

використовувати для покращення методу Г. Лоя, було прийнято рішення протестувати роботу цих алгоритмів на практиці.

Для тестування використовувався пакет прикладних програм MATLAB та бібліотека OpenCV, яка містить готову реалізацію розглянутих вище алгоритмів. Тестування проводилось з використанням 20 зображень різної якості та форматів, для кожного з яких виконувався пошук ключових точок за допомогою кожного алгоритму пошуку ключових точок (рис. 2.1).

Рисунок 2.1 – Результати обробки зображення усіма методами

Отримані результати порівняння методів представлено в таблиці 2.1.

Таблиця 2.1

Середні значення результатів роботи методів пошуку ключових точок

Алгоритм	Кількість знайдених точок	Тривалість роботи, с
SIFT	3424.9	0.2665
SURF	4143.1	0.1847
KAZE	1586.5	0.2820
AKAZE	1743.2	0.0994
ORB	9754.3	0.0393
BRISK	6375.8	0.1695

Аналіз отриманих результатів показав, що для підвищення ефективності методу Г. Лоя для визначення осі симетрії обличчя людини, доцільно замінити метод SIFT на ORB, так, як він знаходить найбільшу кількість ключових точок

за найменший проміжок часу серед розглянутих методів, що дозволить значно підвищити його точність та швидкодію.

2.2. Розробка методу визначення симетричності обличчя людини на базі модифікованого методу визначення осі симетрії

Щоб реалізувати автоматичне визначення симетричності обличчя необхідно, крім використання модифікованого методу визначення осі симетрії, обрати спосіб отримання поверхні для порівняння та метрику оцінки міри симетричності.

Для отримання поверхні для порівняння було прийнято рішення розглянути метод С. Колкура [40] для виявлення шкіри на зображенні. Цей метод передбачає конвертування зображення в дві колірні моделі – HSV та YCbCr, оскільки саме вони найкраще підходять для задач розпізнавання шкіри, як стверджує А. Альбіол в свої роботі [41]

HSV (Hue, Saturation, Value) – колірна модель, за якою координатами кольору є: кольоровий тон (H), насиченість (S) та значення кольору або яскравість (V) [42].

Представлення цієї моделі в конічній формі зображено на рисунку 2.2.

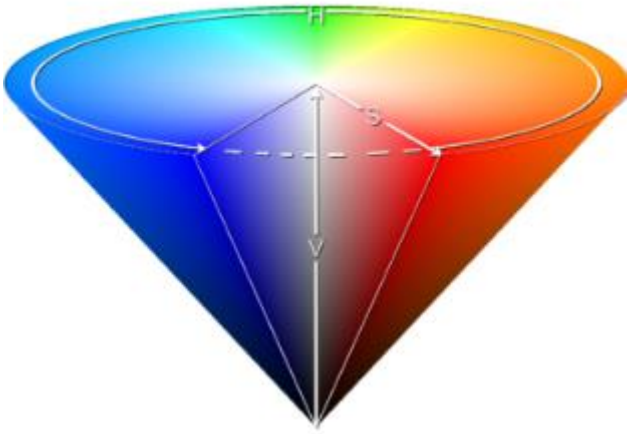


Рисунок 2.2 – Конічне представлення моделі HSV

За цією моделлю, значення H може варіюватись в межах від 0 до 360 градусів, але, зазвичай, приводиться до діапазону від 0 до 100. Значення S може варіюватись в межах від 0 до 100 або 0 до 1 і відповідає за «чистоту» кольору,

тому чим ближче воно до 0, тим колір є ближчим до сірого. Значення V відповідає за яскравість і теж, як правило задається в межах від 0 до 100 або 0 до 1 [42].

YCbCr – це колірна модель, яка зазвичай використовується на телебаченні для стиснення зображень і представляє собою спосіб кодування інформації сигналів RGB [43].

Графічне представлення цієї моделі зображено на рисунку 2.3.

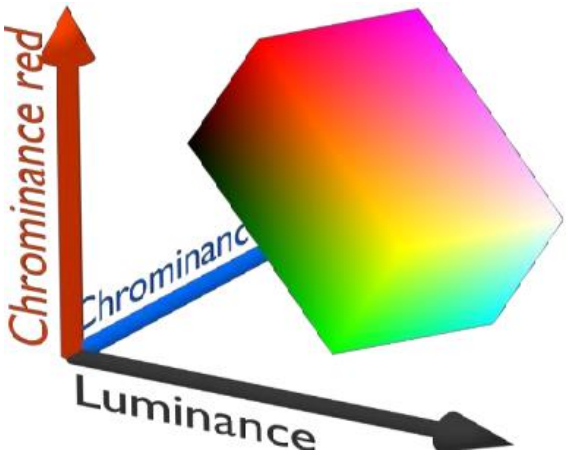


Рисунок 2.3 – Графічне представлення моделі YCbCr

Значення цієї моделі обраховуються за формулами 2.4-2.6.

$$Y = 0.299R + 0.287G + 0.11B. \quad (2.4)$$

$$Cr = R - Y. \quad (2.5)$$

$$Cb = B - Y. \quad (2.6)$$

де Y – яскравість, R, G, B – складові кольору.

Як можна помітити з наведених вище формул, яскравість в цій моделі обраховується як зважена сума значень RBG складових кольору. Таке представлення дає можливість легко позбуватись деякої непотрібної інформації про колір, тому ця модель використовується в таких стандартах стиснення зображень, як JPEG, MPEG1, MPEG2 та MPEG4 [40].

С. Колкура пропонує наступні залежності для перевірки на відповідність пікселя шкірі:

$$0 \leq H \leq 17, 15 \leq S \leq 170, 0 \leq V \leq 255,$$

$$0 \leq Y \leq 255, 135 \leq Cr \leq 180, 85 \leq Cb \leq 135.$$

Якщо значення пікселя в двох колірних моделях відповідає цим умовам, то цей піксель є шкірою.

Точність цього методу було перевірено на базі даних зображень облич людей SFA, яка знаходиться у вільному доступі.

Результат перевірки роботи методу зображено на рисунку 2.4.

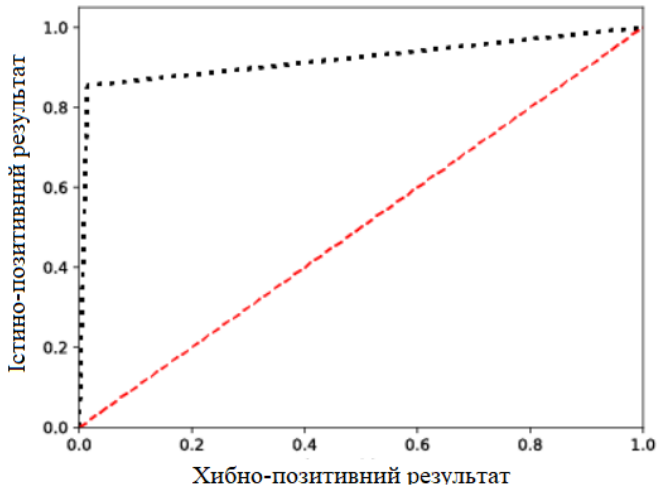


Рисунок 2.4 – ROC-крива роботи методу визначення шкіри

Як можна помітити на рисунку 2.4, точність обраного методу є близькою до 90%, що є дуже високим показником, тому було прийнято використовувати саме цей метод для отримання поверхні для порівняння під час автоматичної оцінки рівня симетрії обличчя людини.

Для порівняння міри симетричності було вирішено використовувати значення нормованої середньоквадратичної похибки (NMSE) [44], яке обраховується за формулою 2.7.

$$NMSE = \frac{\sum_i (R_1(i) - R_2(i))^2 + (G_1(i) - G_2(i))^2 + (B_1(i) - B_2(i))^2}{\sum_i R_1(i)^2 + G_1(i)^2 + B_1(i)^2}, \quad (2.7)$$

де i – номер пікселя, а $(R_1(i), G_1(i), B_1(i)), (R_2(i), R_2(i), R_2(i))$ це інтенсивність складових кольору і-го пікселя для виділеної та віддзеркаленої області зображення.

Отримане значення трактується наступним чином: якщо NMSE<=0,0001, то асиметрія обличчя відсутня; якщо 0,0001<NMSE<=0,00025, то присутня незначна асиметрія; якщо 0,00025<NMSE<=0,001, то рівень асиметрії є середнім; якщо NMSE > 0,001, то рівень асиметрії є високим [44].

2.3. Розробка алгоритму роботи модуля визначення симетричності обличчя людини

Розроблюваний програмний додаток буде містити модуль для визначення симетричності обличчя людини.

Алгоритм його роботи такий:

1. Початок.
2. Завантаження зображення.

Користувач або завантажує нове зображення, або ж обирає одне з уже існуючих в базі даних програми.

3. Пошук осі симетрії.

Використовуючи розроблений модифікований метод визначення осі симетрії, програма виконує її пошук та відображає її на зображенні у вигляді червоної лінії.

4. Вибір способу аналізу. Якщо «Так», то перейти до кроку 5, інакше – перейти до кроку 7.

Користувач обирає, як він хоче перевіряти симетричність: всього обличчя чи лише його частини.

5. Виділення необхідної області обличчя.

Користувач за допомогою миші виділяє квадратом потрібну область.

6. Віддзеркалення виділеної користувачем області відносно встановленої осі симетрії. В результаті отримуються дві області для порівняння, як показано на рисунку 2.5.

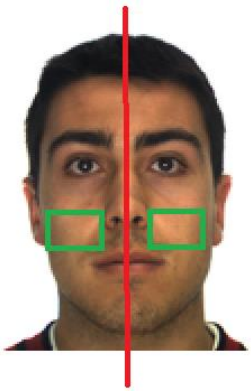


Рисунок 2.5 – Області для оцінки рівня симетричності, які були отримані віддзеркаленням

Перехід до кроку 8.

7. Автоматичне виділення лівої та правої частини обличчя.

Автоматичне виділення виконується за допомогою методу Колкура. В результаті отримується область шкіри людини, як показано на рисунку 2.6.

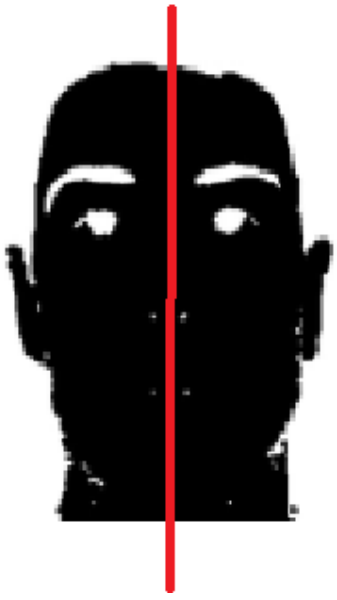


Рисунок 2.6 – Області для оцінки рівня симетричності, які були отримані автоматично

8. Порівняння виділених областей.
9. Виведення результату порівняння.
10. Кінець.

Блок-схему алгоритму роботи модуля визначення симетричності обличчя людини зображено на рисунку 2.7.

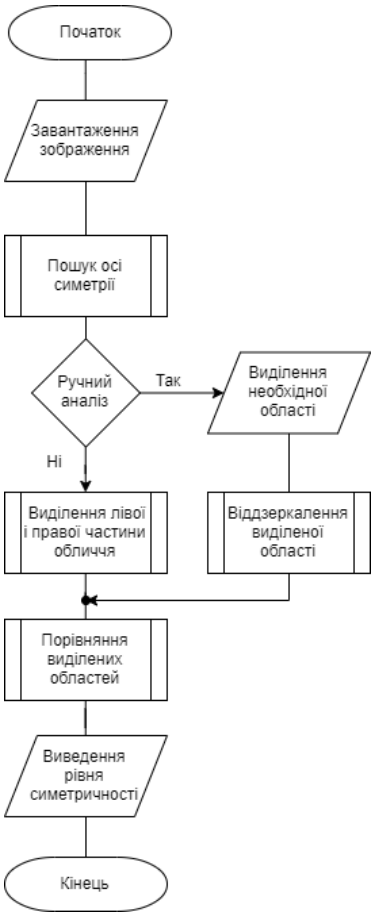


Рисунок 2.7 – Блок-схема алгоритму роботи модуля визначення симетричності обличчя людини

2.4. Висновки

Отже, було модифіковано метод Г. Лоя для визначення осі симетрії обличчя людини шляхом заміни методу визначення ключових точок SIFT на ORB, що дозволило значно підвищити швидкість та точність процесу пошуку осі симетрії обличчя людини.

Також, було обрано метод С. Колкура для виявлення шкіри на зображенні та значення нормованої середньоквадратичної похибки NMSE для проведення автоматизованого визначення симетричності обличчя людини та розроблено алгоритм роботи модуля, який буде його реалізовувати.

3 РОЗРОБКА МЕТОДІВ ДЛЯ ПРОВЕДЕННЯ МЕДИЧНОГО ЕКСПРЕС-ДІАГНОСТУВАННЯ

3.1. Розробка методу діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини

Як вже сказано в 1 розділі, ряду генетичних захворювань властиві прояви на обличчі людини, по яким їх можна діагностувати. Цю тему досліджували у своїй роботі К. Феррі та Д. Штейнберг [45].

Суть їх дослідження полягала у отриманні усереднених зображень облич пацієнтів отриманих за допомогою моделі активного зовнішнього вигляду (active appearance model) [46]. Ця статистична модель зображень, яка шляхом різних видів деформацій може підігнати їх під реальне зображення.

Кількість зображень, які використовувались для усереднення, представлено в таблиці 3.1.

Таблиця 3.1

Кількість зображень, яка відповідає захворюванню

Синдром	Кількість зображень
Ангельмана	205
Аперта	203
Корнеліа де Ланге	179
Дауна	199
Мартіна-Бел	164
Прогерія	78
Трічера-Колінза	103
Вільямса	232

В результаті, було отримано усереднені зображення облич людей з генетичними захворюваннями, які зображено на рисунку 3.1.

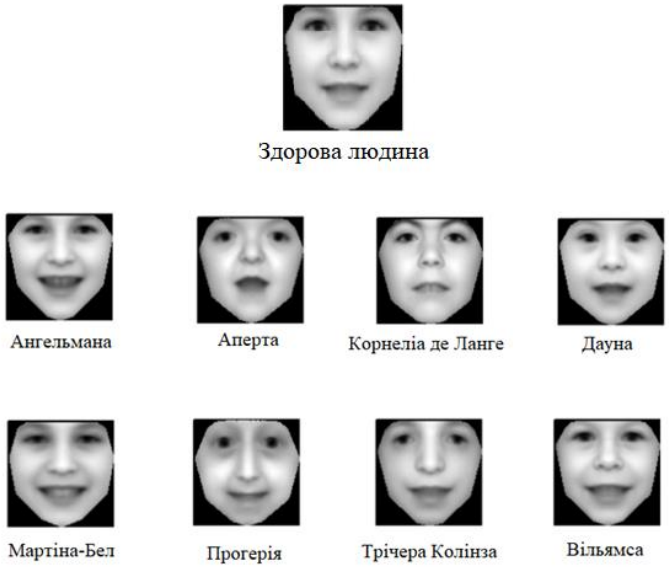


Рисунок 3.1 – Усереднені зображення облич людей з генетичними захворюваннями

Можна помітити, що відстані на обличчі людини при відповідних захворюваннях відрізняються від середньостатистичних. Тому, на основі цих зображень, було запропоновано ряд характерних точок (landmark points) обличчя людини, які відіграють важливу роль в діагностуванні ряду генетичних хвороб. Ці точки зображено на рисунку 3.2.

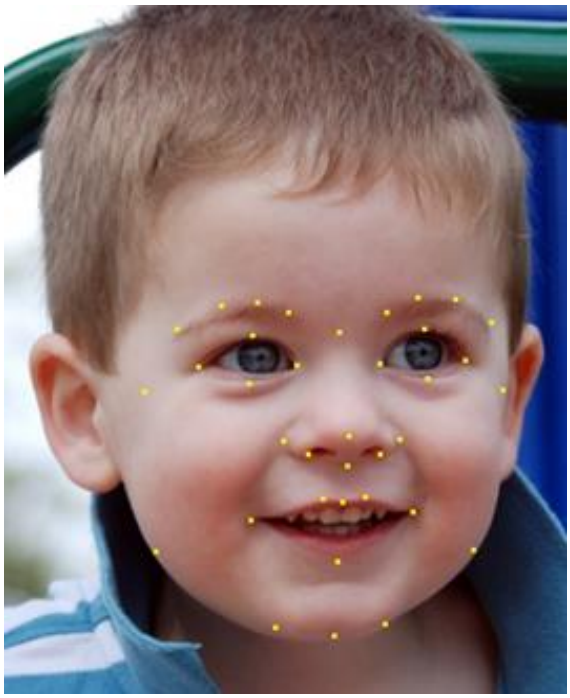


Рисунок 3.2 – Запропоновані характерні точки для діагностування захворювань

Далі, використовуючи характерні точки з рисунку 3.1 та описи генетичних захворювань, представлених в таблиці 3.1, було розроблено ряд рисунків з відстанями на обличчі людини, які можуть свідчити про наявність того чи іншого захворювання. Товстішою лінію на рисунках представлено відстані, які при захворюванні є більшими за середньостатистичні значення, а тоншою – ті, які є меншими за них.

Отримані зображення представлено на рисунках 3.3–3.10.

Рисунок 3.3 – Синдром Ангельмана

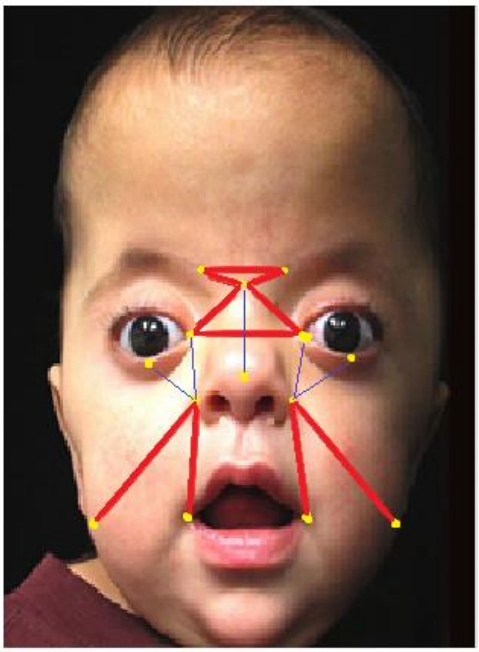


Рисунок 3.4 – Синдром Аперта

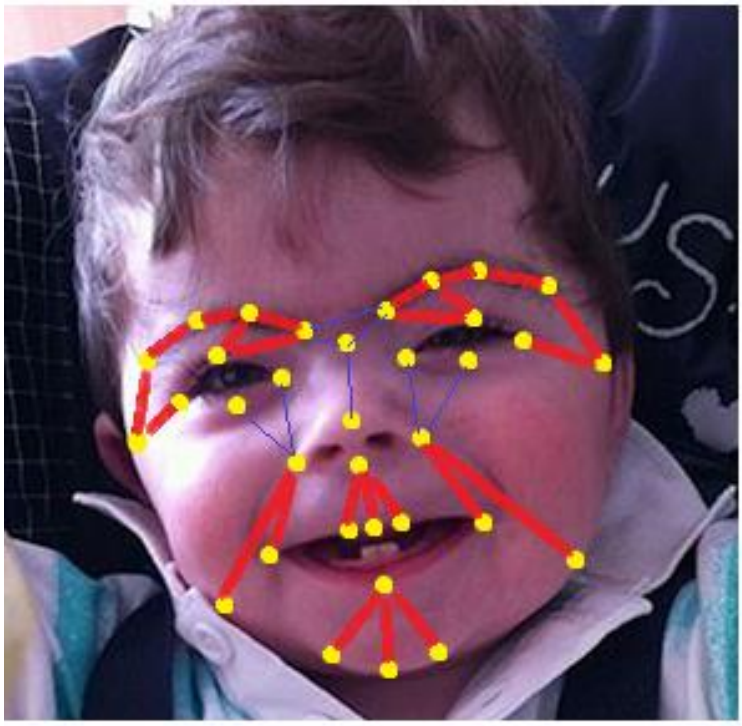


Рисунок 3.5 – Синдром Корнелі де Ланге

Рисунок 3.6 – Синдром Дауна

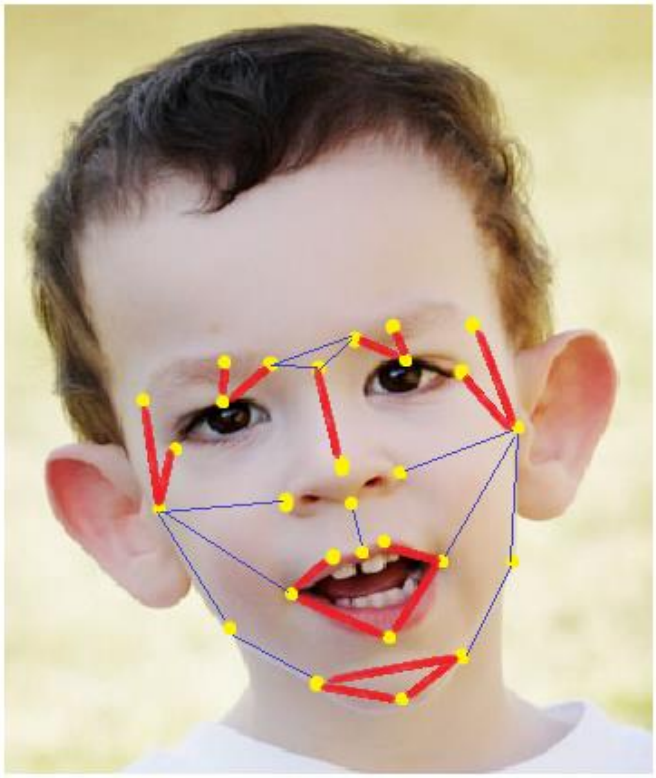


Рисунок 3.7 – Синдром Мартіна-Бел

Рисунок 3.8 – Прогерія

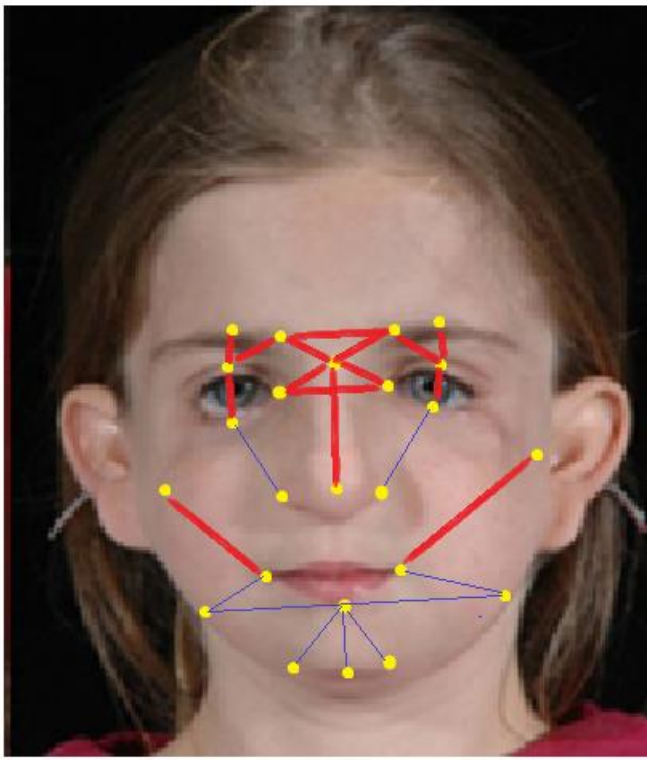


Рисунок 3.9 – Синдром Трічера Колінза

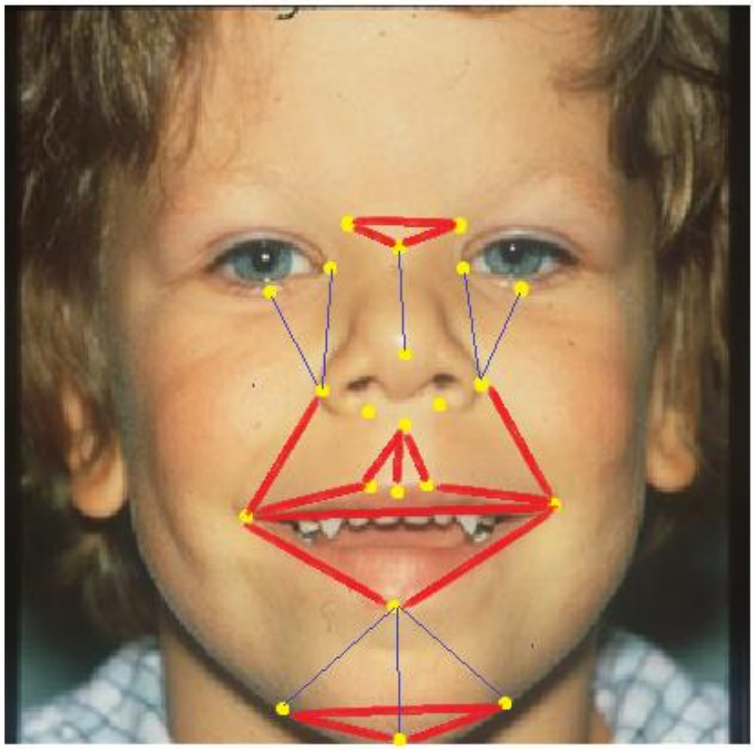


Рисунок 3.10 – Синдром Вільямса

Проаналізувавши та поєднавши отримані рисунки, було розроблено загальну модель відстаней, які необхідно вимірювати для проведення медичного експрес-діагностування генетичних захворювань (рис. 3.11).

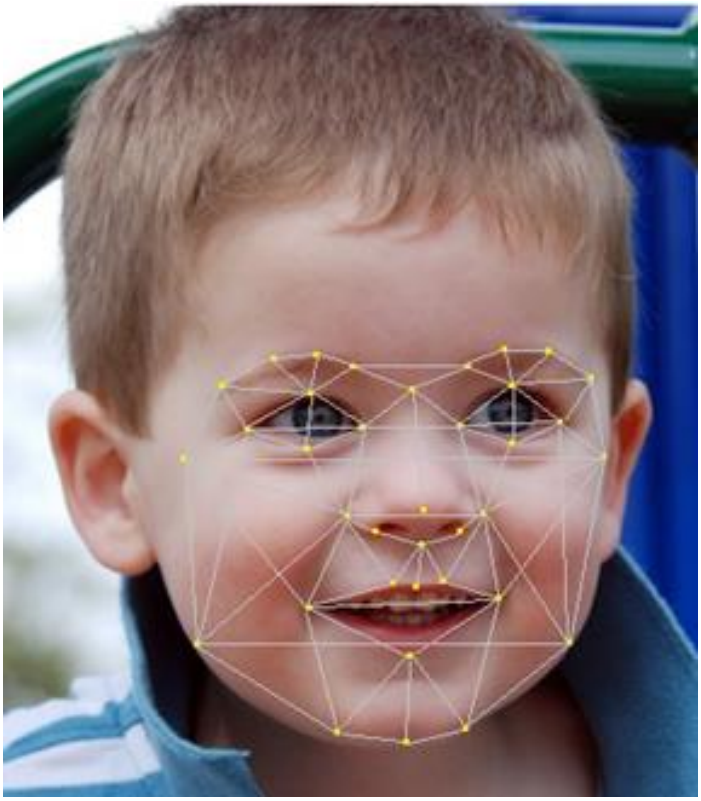


Рисунок 3.11 – Відстані для діагностування генетичних захворювань

Виконуючи вимірювання цих відстаней, можна швидко виключати з можливих ті чи інші генетичні захворювання в процесі діагностування пацієнта, що значно підвищить швидкість визначення типу генетичного захворювання лікарем.

3.2. Отримання аналітичних залежностей обхвату голови дитини з урахуванням вікових змін

Обхват голови дитини є важливим показником його нормального розвитку, тому лікарі щомісяця проводять вимірювання цього значення та звіряють зі спеціальними таблицями. Ці таблиці розроблені спеціалістами-педіатрами, по

яким лікарі звіряють отримані вимірювання та приймають рішення про додаткове обстеження, можливість допомоги або корекції розвитку дитини.

По відхиленням значень обхвату голови можна: розпізнати інфекцію, що розвивається; розпізнати зародження нервового розладу або ж визначити наявність патологій у процесі розвитку головного мозку [47].

Дані таблиць норм розвитку обхвату голови дитини з сайту Всесвітньої організації охорони здоров'я [48] представлено в таблицях 3.2–3.3.

Таблиця 3.2

Обхват голови дівчат від народження до п'яти років

	SD -3	SD -2	SD -1	SD 1	SD 2	SD 3
Новонароджена	30,3	31,5	32,7	35,1	36,2	37,4
1 місяць	33,0	34,2	35,4	37,7	38,9	40,1
2 місяці	34,6	35,8	37,0	39,5	40,7	41,9
3 місяці	35,8	37,1	38,3	40,8	42,0	43,3
4 місяці	36,8	38,1	39,3	41,8	43,1	44,4
5 місяців	37,6	38,9	40,2	42,7	44,0	45,3
6 місяців	38,3	39,6	40,9	43,5	44,8	46,1
7 місяців	38,9	40,2	41,5	44,1	45,5	46,8
8 місяців	39,4	40,7	42,0	44,7	46,0	47,4
9 місяців	39,8	41,2	42,5	45,2	46,5	47,8
10 місяців	40,2	41,5	42,9	45,6	46,9	48,3
11 місяців	40,5	41,9	43,2	45,9	47,3	48,6
12 місяців	40,8	42,2	43,5	46,3	47,6	49,0
24 місяці	43,0	44,4	45,8	48,6	50,0	51,4
36 місяців	44,3	45,7	47,1	49,9	51,3	52,7
48 місяців	45,1	46,5	47,9	50,8	52,2	53,6
60 місяців	45,7	47,1	48,5	51,3	52,8	54,2

Таблиця 3.3

Обхват голови хлопців від народження до п'яти років

	SD -3	SD -2	SD -1	SD 1	SD 2	SD 3
1	2	3	4	5	6	7
Новонароджений	30,7	31,9	33,2	35,7	37,0	38,3
1 місяць	33,8	34,9	36,1	38,4	39,6	40,8
2 місяці	35,6	36,8	38,0	40,3	41,5	42,6
3 місяці	37,0	38,1	39,3	41,7	42,9	44,1
4 місяці	38,0	39,2	40,4	42,8	44,0	45,2

Продовження таблиці 3.3

1	2	3	4	5	6	7
5 місяців	38,9	40,1	41,4	43,8	45,0	46,2
6 місяців	39,7	40,9	42,1	44,6	45,8	47,0
7 місяців	40,3	41,5	42,7	45,2	46,4	47,7
8 місяців	40,8	42,0	43,3	45,8	47,0	48,3
9 місяців	41,2	42,5	43,7	46,3	47,5	48,8
10 місяців	41,6	42,9	44,1	46,7	47,9	49,2
11 місяців	41,9	43,2	44,5	47,0	48,3	49,6
12 місяців	42,2	43,5	44,8	47,4	48,6	49,9
24 місяці	44,2	45,5	46,9	49,6	51,0	52,3
36 місяців	45,2	46,6	48,0	50,9	52,3	53,7
48 місяців	45,8	47,3	48,7	51,7	53,1	54,6
60 місяців	46,3	47,7	49,2	52,2	53,7	55,2

Дані цих таблиць трактують наступним чином [49]:

1. Значення між показниками колонок -1 SD та 1 SD – область середніх величин. Половина здорових дітей такого віку мають таке значення.

2. Значення між показниками -3 SD та -1 SD, 1 SD та 3 SD – області пониженого або підвищеного зросту. Таких дітей в популяції близько 44% і це теж варіант норми. Причиною нижчих/вищих значень можуть бути генетичні особливості (зріст батьків, особливості конституції тіла).

3. Значення окружності голови виходить за межі колонок -3 SD та 3 SD є приводом звернутися до лікаря для обстеження та з’ясування причин відставання/випередження зросту.

Використовуючи дані таблиць 3.2 та 3.3, було побудовано графіки залежності обхвату голови дитини від її віку (рис. 3.12-3.13).

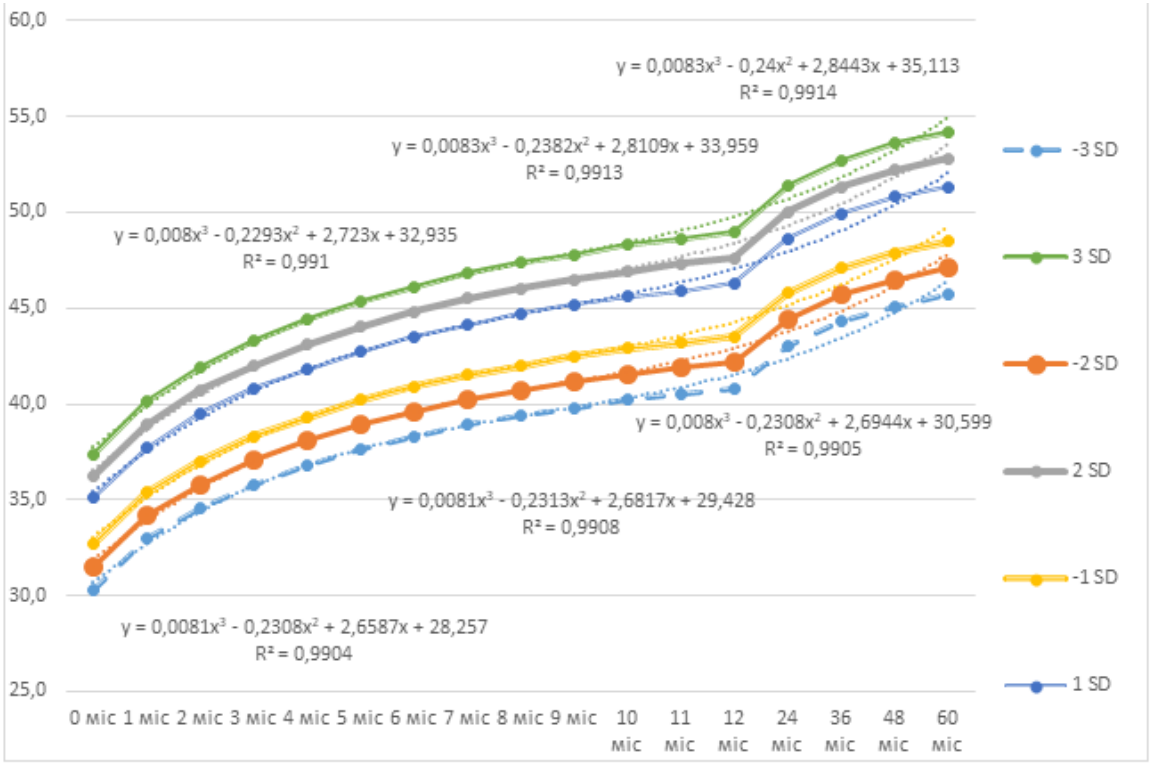


Рисунок 3.12 – Графік залежності обхвату голови дівчини від віку

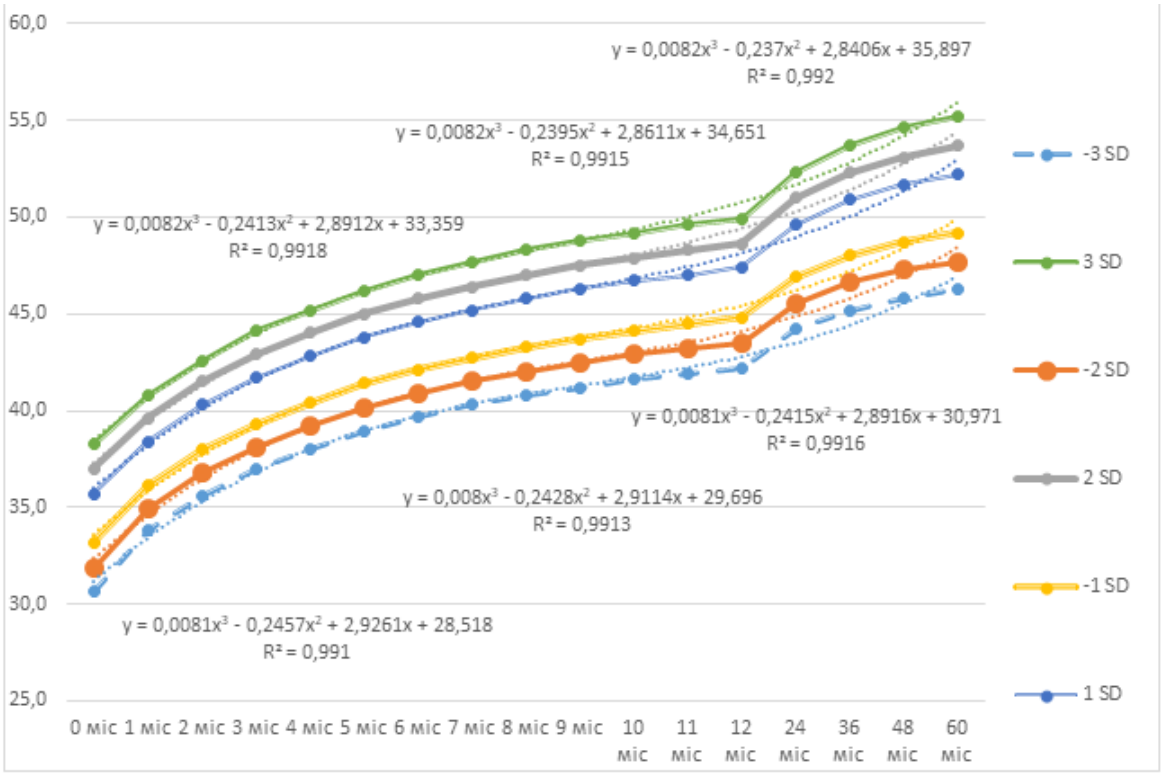


Рисунок 3.13 – Графік залежності обхвату голови хлопця від віку

Використовуючи отримані графіки, було отримано наступні аналітичні залежності (y – обхват голови дитини, x – вік дитини):

1. Дівчата:

$$y_{Д3SD} = 0{,}0083x^3 - 0{,}24x^2 + 2{,}8443x + 35{,}113,$$
$$y_{Д-3SD} = 0{,}0081x^3 - 0{,}2308x^2 + 2{,}6587x + 28{,}257,$$
$$y_{Д2SD} = 0{,}0083x^3 - 0{,}2382x^2 + 2{,}8109x + 33{,}959,$$
$$y_{Д-2SD} = 0{,}0081x^3 - 0{,}2313x^2 + 2{,}6817x + 29{,}428,$$
$$y_{Д1SD} = 0{,}008x^3 - 0{,}2293x^2 + 2{,}723x + 32{,}935,$$
$$y_{Д-1SD} = 0{,}008x^3 - 0{,}2308x^2 + 2{,}6944x + 30{,}599.$$

Якщо y <= $y_{Д1SD}$ && y >= $y_{Д-1SD}$, то дівчина повністю здорова.

Якщо (y >= $y_{Д2SD}$ && y <= $y_{Д3SD}$) && (y <= $y_{Д-2SD}$ && y >= $y_{Д-3SD}$), то дівчина здорова, але розміри голови є трохи вищими/нижчими за норму.

Якщо (y >= $y_{Д3SD}$ || y <= $y_{Д-3SD}$), то у дівчини є значні відхилення від норми і потрібне додаткове обстеження лікарем.

2. Хлопці:

$$y_{Х3SD} = 0{,}0082x^3 - 0{,}237x^2 + 2{,}8406x + 35{,}897,$$
$$y_{Х-3SD} = 0{,}0081x^3 - 0{,}2457x^2 + 2{,}9261x + 28{,}518,$$
$$y_{Х2SD} = 0{,}0082x^3 - 0{,}2395x^2 + 2{,}8611x + 34{,}651,$$
$$y_{Х-2SD} = 0{,}008x^3 - 0{,}2428x^2 + 2{,}9114x + 29{,}696,$$
$$y_{Х1SD} = 0{,}0082x^3 - 0{,}2413x^2 + 2{,}8912x + 33{,}359,$$
$$y_{Х-1SD} = 0{,}0081x^3 - 0{,}2415x^2 + 2{,}8916x + 30{,}971.$$

Якщо y <= $y_{Х1SD}$ && y >= $y_{Х-1SD}$, то хлопець повністю здоровий.

Якщо (y >= $y_{Х2SD}$ && y <= $y_{Х3SD}$) && (y <= $y_{Х-2SD}$ && y >= $y_{Х-3SD}$), то хлопець здоровий, але розміри голови є трохи вищими/нижчими за норму.

Якщо (y >= $y_{Х3SD}$ || y <= $y_{Х-3SD}$), то у хлопця є значні відхилення від норми і потрібне додаткове обстеження лікарем.

Використовуючи отримані аналітичні залежності, можна розробити програмне забезпечення для діагностування шляхом порівняння параметрів голови дитини з допустимими нормами.

3.3. Розробка алгоритмів для медичного експрес-діагностування

Розроблюваний програмний додаток буде містити модуль для проведення експрес-діагностування генетичних захворювань.

Для його роботи необхідно обрати базу даних облич людей, на базі яких можна проводити навчання системи визначення характерних точок.

Існує цілий ряд баз даних обличь у відкритому доступі. Їх можна класифікувати на дві категорії: 1 – ті, які було отримано в контрольованих умовах (Multi-PIE, XM2VTS, FRGC-V2); 2 – ті, які було отримано за неконтрольованих умов (AR, LFPW, HELEN, AFW, AFLW). Всі вони охоплюють різні варіації, включаючи різні пози, вирази обличчя та види освітлення [50].

Ключові точки, які отримуються при використанні описаних баз даних зображено на рисунку 3.14.

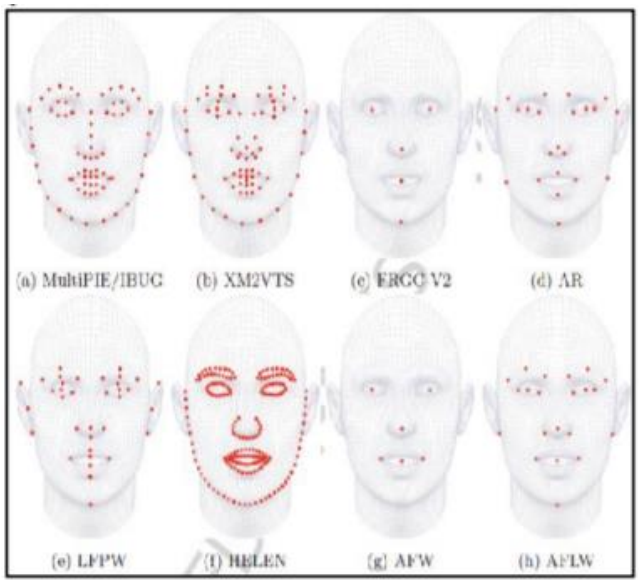


Рисунок 3.14 – Ключові точки для кожної бази даних

Загальну характеристику кожної з розглянутих баз даних представлено в таблиці 3.4.

Таблиця 3.4

Характеристика баз даних для визначення ключових точок

Назва	Вид умов	Кількість обличь	Кількість ключових точок
FRGC-V2	контрольовані	4950	5
XM2VTS		2360	68
Multi-PIE		750000	68
AR		4000	22
LFPW	неконтрольовані	1035	35
AFW		468	6
AFLW		25993	21
HELEN		2330	194

Проаналізувавши дані таблиці 3.4, було встановлено, що найбільш ефективною базою даних для визначення ключових точок є Multi-PIE та HELEN. Для розроблюваного додатку було прийнято рішення використовувати саме Multi-PIE, оскільки він має значно більшу точність за рахунок великої кількості зображень, що є дуже важливим в задачах діагностування генетичних захворювань.

Загальний алгоритм роботи модуля проведення експрес-діагностування генетичних захворювань:

1. Початок.
2. Завантаження зображень з бази даних Multi-PIE для тренування системи визначення характерних точок.
3. Обробка завантажених зображень та формування на їх основі моделі для визначення ключових точок.
4. Вибір зображень користувачем, для яких буде проводитись аналіз.
5. Пошук характерних точок на обраних зображеннях.
6. Вимірювання відстаней між точками, які передбачені розробленою моделлю, яка описана в розділі 3.1.

7. Візуалізація результатів вимірювань та порівняння з нормованими значеннями.

8. Якщо користувач задоволений ними, то відбувається збереження отриманих результатів в базу даних і перехід до кроку 10. Якщо ні, то перехід до кроку 9.

9. Редагування користувачем нормованих значень відстаней і перехід до кроку 5.

10. Кінець.

Блок-схему роботи описаного алгоритму зображено на рисунку 3.15.

Рисунок 3.15 – Блок-схема алгоритму роботи модуля для експрес-діагностування генетичних захворювань

Також, програмний додаток буде містити модуль для оцінки розвитку дитини за допомогою аналітичних залежностей, отриманих в розділі 3.2.

Алгоритм роботи цього модуля такий:

1. Початок.
2. Вибір зображення користувачем.
3. Вилучення фону зображення для отримання контуру голови.
4. Вимірювання довжини та ширини голови по крайнім точкам отриманого зображення.
5. Обрахунок обхвату голови.
6. Порівняння отриманого значення обхвату голови з нормованими значеннями.
7. Виведення результату.
8. Кінець.

Блок-схему роботи модуля для оцінки розвитку дитини зображено на рисунку 3.16.

Рисунок 3.16 – Блок-схема алгоритму роботи модуля для оцінки розвитку дитини

3.4. Висновки

Отже, під час виконання третього розділу було розроблено метод діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини, а саме вимірювання відстаней між характерними точками, які було запропоновано, та їх порівняння з нормованими значеннями. Цей метод дає можливість проводити медичне-експрес діагностування таких захворювань, як синдроми: Ангельмана, Аперта, Корнеліа де Ланге, Дауна, Мартіна-Бел, Трічера-Колінза, Вільямса та прогерія.

Було отримано аналітичні залежності обхвату голови дитини від її віку, які можна використовувати для розробки програмного забезпечення для діагностування ряду захворювань шляхом порівняння параметрів голови дитини з допустимими нормами.

Також, було розроблено алгоритми роботи модулів для проведення медичного-експрес діагностування та які будуть використовуватись під час реалізації програмного додатку.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ МЕДИЧНОГО ЕКСПРЕС-ДІАГНОСТУВАННЯ

4.1. Варіантний аналіз і обґрунтування вибору програмних засобів

Наступним кроком для розробки програмного додатку для медичного експрес-діагностування є вибір мови програмування, а також інструментів та середовище розробки.

Було прийнято рішення проводити розробку під операційну систему Windows, так як вона є найрозповсюдженішою у світі на даний момент. Оскільки, компанія Microsoft зупинила підтримку таких операційних систем, як Windows XP/7/8.1, тому доцільно розробляти програму саме для Windows 10. Популярними мовами розробки додатків для ОС Windows є C++, C# та VB.NET.

Мова програмування C++ є мовою високого рівня, яка підтримує кілька парадигм програмування: процедурну, узагальнену та об’єктно-орієнтовану. Вона була розроблена Б’ярном Страуструпом у 1979 році, на основі мови С. У порівнянні зі своїм попередником, вона приділяє більше уваги саме об’єктно-орієнтованому та узагальненому програмуванню. Цю мову використовують для розробки прикладних програм, драйверів, серверних та клієнтських програм, а також відеоігор [51].

Особливостями мови C++ є:

1. Висока продуктивність.
2. Підтримка різних стилів програмування.
3. Наявність великої кількості навчальних матеріалів.
4. Можливість використання стандартної бібліотеки шаблонів та створення власних шаблонів класів.
5. Можливість розширення мови для підтримки парадигм, які не підтримуються компілятором на пряму.
6. Підтримка обробки винятків, перезавантаження операторів та вбудованих функцій.

Мова Visual Basic .NET (VB.NET) – це об’єктно-орієнтована мова програмування, яка можна розглядати як розвиток мови Visual Basic, який реалізований на платформі Microsoft.NET [52].

Особливостями мови VB.NET є:

1. Підтримка концепцій об’єктно-орієнтованого програмування з конструкторами, деструкторами, наслідуванням та перезавантаженням методів.
2. Компіляція в байт-код, який виконується за допомогою CLR.
3. Підтримка вільної багатопоточності.
4. Використання об’єктних бібліотек .NET Framework, включають засоби по роботі з формами (Windows Forms), базами даних (ADO.NET), графікою (GDI+), засобами забезпечення безпеки та іншими.
5. Підтримка структурної обробки виключень.

Мова C# є об’єктно-орієнтованою мовою програмування, яка була розроблена у 1998-2001 роках групою інженерів компанії Microsoft, під керівництвом Андерса Хейлсберга та Скотта Вільтаумота як мова розробки додатків для платформи Microsoft .NET Framework. Вона належить до сімейства мов з C-подібним синтаксисом, найбільш близькими до якої є мови C++ та Java. Ця мова надає можливість просто та швидко розробляти як малі, так і великі проекти [53].

Особливостями мови C# є:

1. Автоматичне звільнення динамічно розподіленої пам’яті.
2. Атрибути, які дають можливість отримати декларативні метадані про типи під час виконання програми.
3. Наявність властивостей, які виконують функцію аксесорів для закритих змінних.
4. Відсутність множинного наслідування.
5. Доступ до бібліотеки базових класів .NET.
6. Підтримка універсальних методів та типів, які забезпечують вищий рівень безпеки та продуктивності.

7. Конструкції LINQ для забезпечення запитів до різноманітних джерел даних.

Порівняння мов програмування наведено в таблиці 4.1.

Таблиця 4.1

Порівняння мов програмування

	C++	VB.NET	C#
Підтримка об'єктно-орієнтованого програмування	+ (0.8)	+ (0.8)	+ (0.8)
Швидкодія	+ (0.7)	+ (0.7)	+ (0.7)
Простота та зручність розробки	–	+ (0.8)	+ (0.8)
Наявність великої кількості документації та прикладів	+ (0.9)	–	+ (0.9)
Сума	2.4	2.3	3.2

Проаналізувавши дані таблиці 4.1, можна зробити висновок, що найкращий результат (3.2) у мови C#. Тому, для реалізації програмного додатку було вирішено використати мову програмування C#, оскільки вона надає усі необхідні умови для проведення швидкої та якісної розробки.

Тепер необхідно обрати технологію для розробки інтерфейсу додатку. В мові C# є такі технології, як Windows Forms, Windows Presentation Foundation та Universal Windows Platform.

Windows Forms – це інтерфейс програмування додатків, який використовується для розробки користувацького інтерфейсу та є частиною .NET Framework [54]. Цей інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Особливістю цієї технології є простота у використанні та можливість роботи в різних операційних системах. Недоліками Windows Forms

є відсутність шаблонів, системи прив’язки даних, та складність створення власних елементів інтерфейсу з деталізованим оформленням.

WPF (Windows Presentation Foundation) – це графічний інтерфейс для створення графічних програмних додатків з насиченим дизайном та можливостями взаємодії з користувачем, який входить до складу .NET Framework і використовує мову XAML [55]. Використання технології WPF надає можливість створювати широкий спектр як автономних додатків, так і додатків, які запускаються у браузері, в основі якої знаходиться потужна інфраструктура, яка базується на DirectX.

Особливостями цього інтерфейсу є гнучкість у створенні зовнішнього вигляду інтерфейсу користувача, можливість описувати інтерфейс користувача у декларативній манері за допомогою мови XAML, наявність технології «зв’язування даних», яка дозволяє оновлювати стан інтерфейсу користувача у відповідь на зміну властивостей, до яких виконується прив’язування.

Universal Windows Platform (UWP) – це інтерфейс, який був розроблений компанією Microsoft виключно для операційної системи Windows 10. Його особливістю є те, що він надає можливість розробляти універсальні інтерфейси, які будуть працювати на усіх пристроях з ОС Windows 10, без змін в програмному коді [56]. Він був розроблений як розширення для Windows Runtime і дозволяє запускати програми на різних апаратних платформах.

Особливостями технології UWP є: безпека; можливість використання спільного API на всіх пристроях під управлінням Windows 10; здатність використання можливостей конкретного пристрою для адаптації користувацького інтерфейсу для різних розмірів екрану. Але їй властиві значні недоліки, а саме: новизна технології, у зв’язку з чим її робота може бути не завжди стабільною; прив’язаність до Microsoft Store, який є відносно непопулярним серед користувачів цієї операційної системи.

Враховуючи особливості розглянутих технологій, було прийнято рішення використовувати технологію WPF, оскільки вона найкраще підходить для розробки програмних додатків для операційної системи Windows.

Для спрощення процесу роботи з зображеннями було прийнято рішення використовувати бібліотеку EMGU CV. Це крос-платформна обгортка для бібліотеки OpenCV для обробки зображень, яка надає можливість викликати її функції з мовами, які підтримують .NET Framework, такі як C#, VC++, VB, IronPython та інші [57].

Для роботи з базою даних було прийнято рішення використовувати Entity Framework. Це набір технологій в ADO.NET, який спрощує взаємодію програмних додатків з базами даних шляхом використання об’єктів .NET [58].

4.2. Вибір середовища розробки

Для розробки програмного додатку, який буде надавати можливість проводити медичне експрес-діагностування, потрібно обрати інтегроване середовище розробки (IDE − Integrated Development Environment).

IDE − це комп’ютерна програма, що допомагає програмісту розробляти нове програмне забезпечення чи модифікувати або удосконалювати вже існуюче. Для порівняння було вирішено обрати найпопулярніші середовища розробки, які підтримують мову програмування C#, а саме: Microsoft Visual Studio, MonoDevelop та JetBrains Rider.

Microsoft Visual Studio – це платформа для розробки програмного забезпечення від компанії Microsoft, яка надає можливість розробляти програмні додатки у різних операційних системах, до складу якої входить інтегроване середовище розробки, а також набір різноманітних додаткових інструментів [59].

Використання цієї платформи надає можливість розробляти консольні програмні додатки, програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Presentation Foundation, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Основними можливостями цього середовища є: підсвічування синтаксису; можливість покрокового виконання коду; вбудована підтримка Git; налагоджувач який може працювати як на рівні вихідного коду, так і на машинному рівні; вбудований редактор форм та можливість підключення додаткових бібліотек.

MonoDevelop – вільне середовище розробки, призначене для створення додатків на мовах C#, Java, Boo, Nemerle, Visual Basic .NET, Vala, CIL, C та C++. MonoDevelop є частиною проекту Mono і містить вбудований дистрибутив Unity3D як засіб написання скриптів [60].

До основних можливостей цього середовища можна віднести підсвітку синтаксису, згортання коду, автодоповнення, браузер класів, підтримка плагінів, вбудований налагоджувач, візуальний конструктор форм та підтримка модульного тестування.

JetBrains Rider – це кросплатформне інтегроване середовище розробки програмного забезпечення для платформи .NET, розроблюване компанією JetBrains. Підтримуються мови програмування C#, VB.NET та F#. У його основі лежить інший продукт JetBrains – ReSharper [61]. Середовище підтримує платформи .NET Framework, .NET Core та Mono та працює на операційних системах Windows, MacOS і Linux.

До особливостей цього середовища можна віднести кросплатформність, системи навігації та рефакторингу коду, перевірку наявності помилок та вбудований декомпілятор. У зв’язку з тим, що це середовище було випущено відносно нещодавно, йому властива наявність певних помилок та збоїв.

Результати порівняльного аналізу IDE наведено в таблиці 4.2.

Таблиця 4.2

Порівняльний аналіз інтегрованих середовищ розробки

	Visual Studio 2017	MonoDevelop	JetBrains Rider
Підсвітка синтаксису	+ (0.5)	+ (0.5)	+ (0.5)
Автодоповнення коду	+ (0.4)	+ (0.4)	+ (0.4)
Вбудований налагоджувач	+ (0.6)	+ (0.6)	+ (0.6)
Додаткові налаштування середовища	+ (0.5)	–	–
Візуальний конструктор форм	+ (0.7)	+ (0.7)	+ (0.7)
Наявність безкоштовної версії	+ (0.6)	+ (0.6)	–
Вбудована система управління пакетами NuGet	+ (0.7)	+ (0.7)	+ (0.7)
Сума	4.0	3.5	2.9

Аналіз результатів порівняння описаних середовищ розробки показав, що для реалізації розроблюваного програмного додатку буде доцільно обрати середовище Microsoft Visual Studio 2017, оскільки воно найбільше підходить для вирішення поставлених задач розробки.

4.3. Розробка структури бази даних

Для проектування бази даних необхідно сформувати універсальне відношення. Універсальним відношенням називають відношення, яке вміщує в себе всі атрибути, які будуть використовуватись у базі даних. Під час розробки невеликих баз даних його використовують в якості відправної точки при проектуванні.

Атрибут – це характеристика певної сутності бази даних. Кожен атрибут відображає відповідну характеристику певного інформаційного об’єкта [62].

Розроблювана база даних повинна включати в себе атрибути, які представлені в таблиці 4.3.

Таблиця 4.3

Перелік атрибутів універсального відношення

№	Назва атрибута	Ім’я поля	Коментар
1	ID пацієнта	PatientID	Ідентифікаційний номер пацієнта
2	ПІБ пацієнта	PatientName	ПІБ пацієнта
3	Телефон пацієнта	PatientPhone	Телефон пацієнта
4	Стать пацієнта	PatientSex	Стать пацієнта
5	Електронна адреса пацієнта	PatientEmail	Електронна адреса пацієнта
6	Дата народження пацієнта	PatientDateOfBirth	Дата народження пацієнта
7	Обхват голови пацієнта	PatientCircumference	Обхват голови пацієнта
8	Рівень симетричності обличчя пацієнта	PatientSymmetryLevel	Рівень симетричності обличчя пацієнта
9	Діагноз пацієнта	PatientDiagnosis	Діагноз пацієнта
10	ID зображення	ImageID	Ідентифікаційний номер зображення
11	Шлях до зображення	ImagePath	Шлях до зображення
12	Дата зображення	ImageDate	Дата зображення
13	Тип зображення	ImageType	Тип зображення
14	ID відстані	DistanceID	Ідентифікаційний номер відстані на обличчі
15	Назва відстані	DistanceName	Назва відстані
16	Значення відстані	DistanceValue	Значення відстані
17	Мінімальна відстань	DistanceMin	Мінімальна відстань
18	Максимальна відстань	DistanceMax	Максимальна відстань

Відповідно до таблиці 4.3 універсальне відношення має наступний вигляд: R(PatientID, PatientName, PatientPhone, PatientSex, PatientEmail, PatientDateOfBirth, PatientCircumference, PatientSymmetryLevel, PatientDiagnosis, ImageID, ImagePath, ImageDate, ImageType, DistanceID, DistanceName, DistanceValue, DistanceMin, DistanceMax).

Потужність універсальної множини – 18.

На основі інформаційних об’єктів бази даних, можна виділити такі сутності та їх атрибути:

1. Patient (<PatientID>, PatientName, PatientPhone, PatientSex, PatientEmail, PatientDateOfBirth, PatientCircumference, PatientSymmetryLevel, PatientDiagnosis).
2. Image (<ImageID>, ImagePath, ImageDate, ImageType).
3. Distance (<DistanceID>, DistanceName, DistanceValue, DistanceMin, DistanceMax).

Відповідно до отриманих сутностей, було розроблено структуру бази даних, яка зображена на рисунку 4.1.

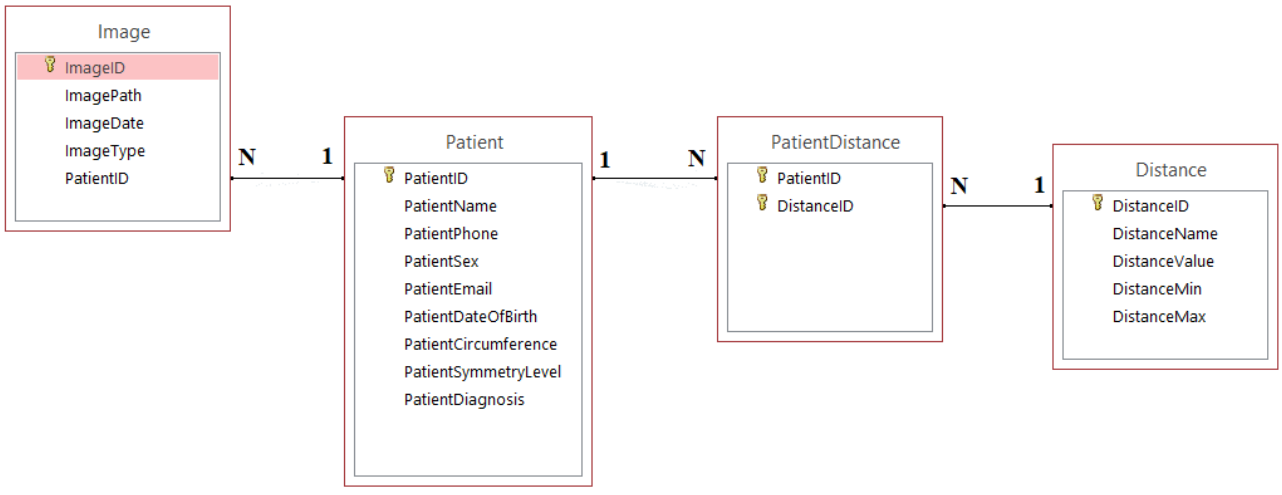


Рисунок 4.1 – Структура бази даних

Сутність Patient знаходяться у відношенні 1:N до сутності Image, оскільки у одного пацієнта може бути декілька його зображень. Тому, в сутності Image,

згідно з 4 правилом нормалізації методом «сутність-зв'язок», з'явилось додаткове поле «PatientID» [63].

Сутність Patient знаходиться у відношенні N:M до сутності Distance, оскільки в пацієнтів може бути ряд відстаней на обличчі. Тому, згідно 6 правила нормалізації, було створено додаткове відношення, яке містить серед своїх атрибутів ключі зв'язних сутностей «PatientID» та «DistanceID».

4.4. Розробка інтерфейсу програми

Важливим етапом розробки будь-якого програмного додатку є розробка його інтерфейсу. Наявність якісного інтерфейсу є дуже важливою для усіх програм, оскільки саме з інтерфейсом стикається користувач при першому запуску програми і якщо цей інтерфейс йому не сподобається – він може одразу її вимкнути або видалити. Щоб цього не сталось, при проектуванні інтерфейсу програм необхідно прагнути того, щоб він був якісним, виразним, послідовним, естетично красивим, продуктивним та поблажливо ставився до помилок користувача. Для цього, спочатку необхідно розробити структуру вікон програми.

Розроблюваний програмний додаток буде базуватись на шаблоні проектування MVVM [64]. Це архітектурний шаблон, суть якого полягає у відокремленні розробки інтерфейсу користувача та бізнес-логіки (рис. 4.2).

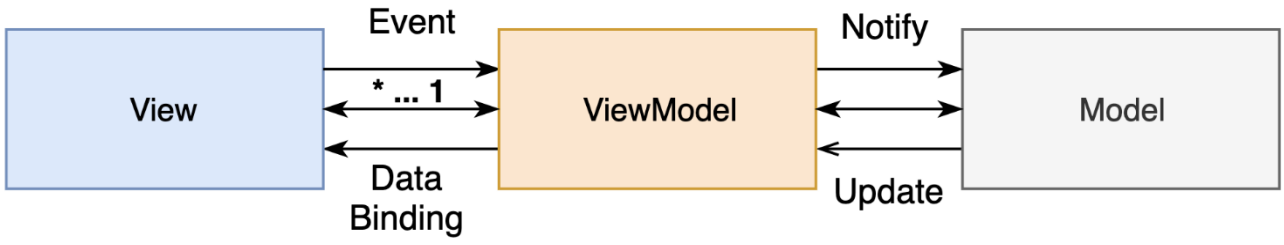


Рисунок 4.2 – Структура шаблону проектування MVVM

У цьому шаблоні проектування, Model (модель) виконує роль DAL (data access level) і відповідає за пряму взаємодію з даними програми.

ViewModel (модель представлення) виконує роль проміжної ланки між моделлю та представленням за допомогою команд та публічних властивостей класів, яку можна описати як поточний стан даних в моделі.

View (представлення) – це структура та розмітка інтерфейсу, яку бачить користувач. Вона відповідає за відображення даних, які зберігаються в моделі, та отримання і передачу взаємодії користувача до ViewModel (моделі представлення) через прив’язку даних.

Слідуючи цій архітектурі, інтерфейс користувача буде складатись з одного вікна, в якому будуть відображатись різні варіанти представлень.

Розроблюваний додаток буде складатись з таких представлень:

1. «MainView» – головне вікно програми.
2. «CreatePatientView» – вікно створення пацієнта.
3. «SelectPatientView» – вікно вибору пацієнта.
4. «DistancesNormsView» – вікно для редагування нормованих значень відстаней на обличчі людини.
5. «SymmetryNormsView» – вікно для редагування нормованих значень симетричності обличчя людини.

Структурна схема інтерфейсу додатку зображена на рисунку 4.3.

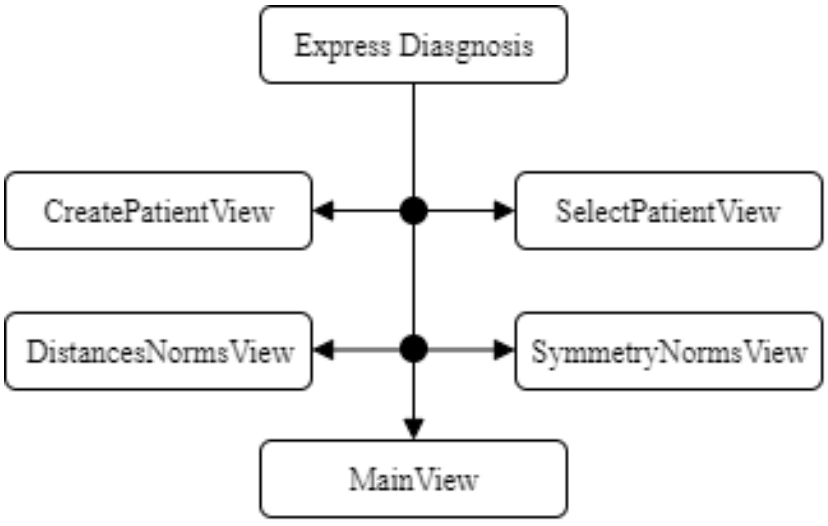


Рисунок 4.3 – Структурна схема додатку «Express Diagnosis»

Графічну схему інтерфейсу головного вікна програми зображено на рисунку 4.4.

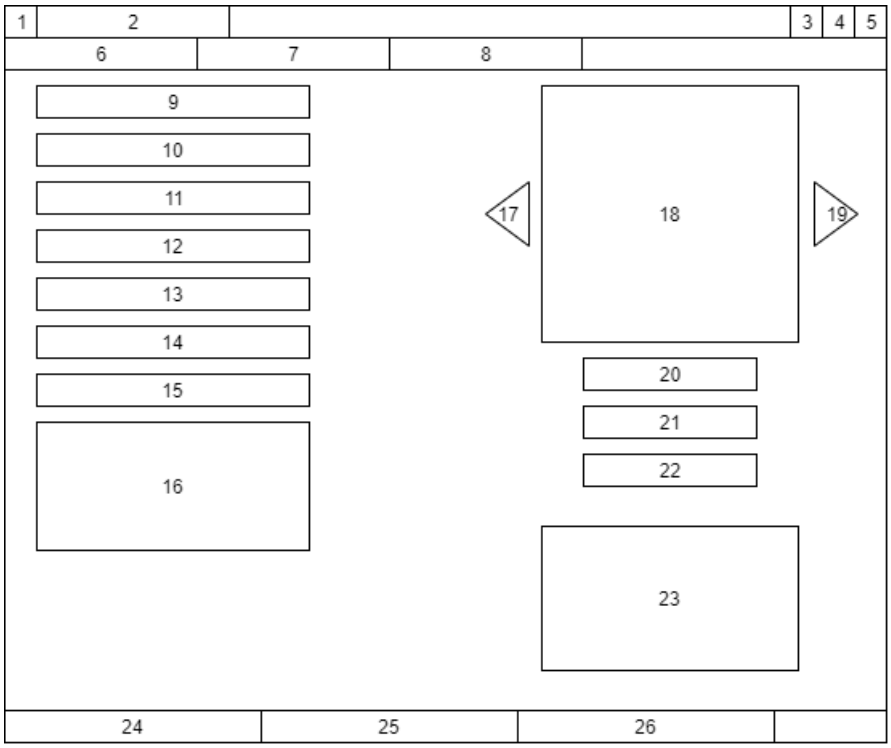


Рисунок 4.4 – Графічна схема головного вікна програми

Головне вікно програми складається з таких елементів:

1. Логотип програми.
2. Назва програми.
3. Кнопка «Згорнути».
4. Кнопка «Розгорнути».
5. Кнопка «Закрити».
6. Пункт меню «Пацієнт».
    6.1. Пункт меню «Створити».
    6.2. Пункт меню «Обрати».
    6.3. Пункт меню «Додати зображення».
    6.4. Пункт меню «Вихід».
7. Пункт меню «Автоматичний аналіз».
    7.1. Пункт меню «Генетичні захворювання».

7.2. Пункт меню «Симетрія».

7.3. Пункт меню «Обхват голови».

8. Пункт меню «Налаштування».

8.1. Пункт меню «Нормовані відстані».

8.2. Пункт меню «Ступені асиметрії».

9. Поле для виведення імені.

10. Поле для виведення статі.

11. Поле для виведення дати народження.

12. Поле для виведення телефону.

13. Поле для виведення електронної пошти.

14. Поле для виведення обхвату голови.

15. Поле для виведення рівня симетричності обличчя.

16. Поле для виведення діагнозу.

17. Кнопка для повернення на попереднє зображення.

18. Область для виведення зображення пацієнта.

19. Кнопка для переходу на наступне зображення.

20. Кнопка «Аналіз генетичних захворювань».

21. Кнопка «Аналіз симетричності»

22. Кнопка «Аналіз обхвату голови».

23. Область для виведення результатів аналізу.

24. Область для виведення імені зображення.

25. Область для виведення номеру поточного зображення.

26. Область для виведення відстані між точками.

Навігація між представленнями додатку буде відбуватись по натисненню пунктів меню 6-8.

При натисненні на пункт меню «Створити» буде відкриватись вікно для створення нового пацієнта.

Графічну схему вікна створення нового пацієнта зображено на рисунку 4.5.

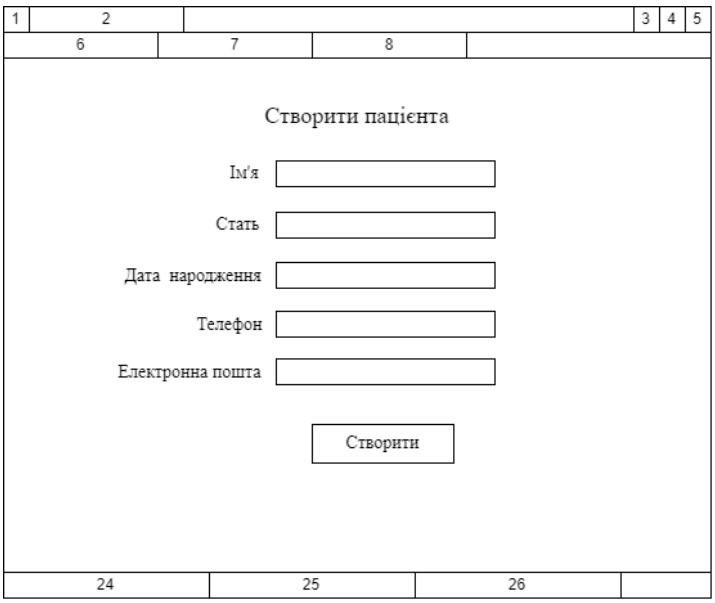


Рисунок 4.5 – Графічна схема вікна «CreatePatientView»

При натисненні на пункт меню «Обрати» буде відкриватись вікно з усіма створеними пацієнтами, де користувач може або перейти до роботи з одним із них, або ж видалити його.

Графічну схему вікна вибору пацієнта зображено на рисунку 4.6.

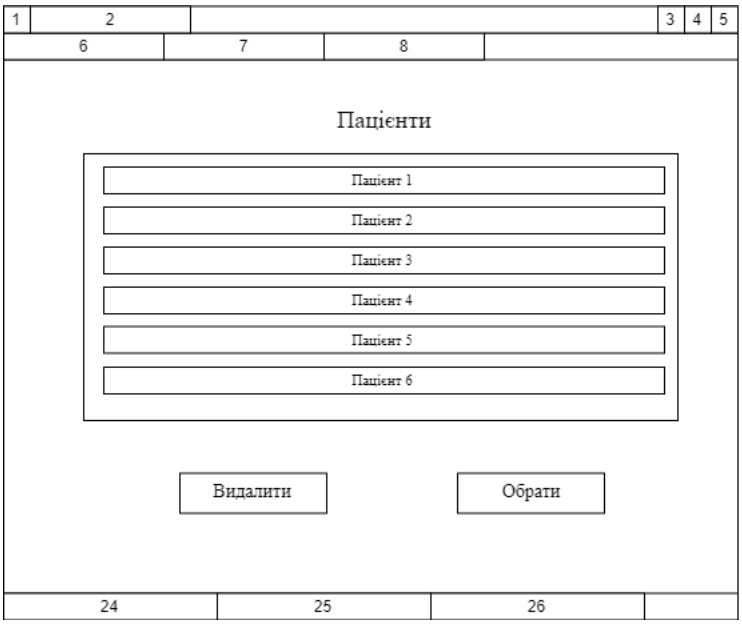


Рисунок 4.6 – Графічна схема вікна «SelectPatientView»

При натисненні на пункт меню «Нормовані відстані» буде відкриватись вікно для редагування нормованих значень відстаней на обличчі людини.

Графічну схему вікна редагування нормованих відстаней зображено на рисунку 4.7.

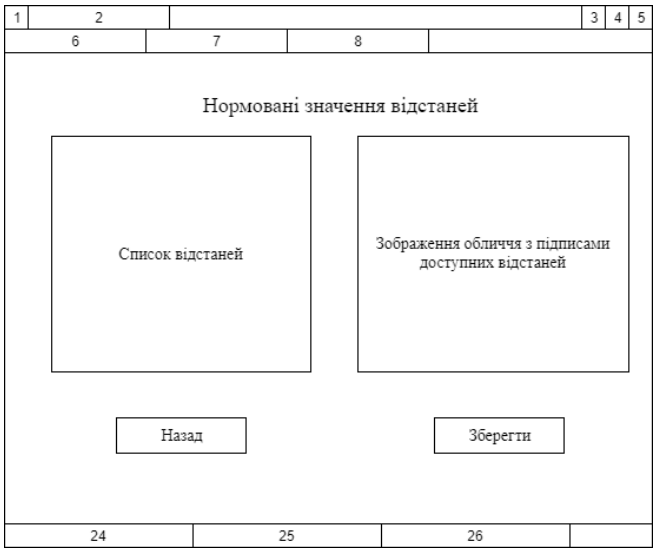


Рисунок 4.7 – Графічна схема вікна «DistancesNormsView»

При натисненні на пункт меню «Ступені асиметрії» буде відкриватись вікно для редагування нормованих значень асиметричності обличчя людини.

Графічну схему вікна редагування нормованих значень асиметричності зображено на рисунку 4.8.

Рисунок 4.8 – Графічна схема вікна «SymmetryNormsView»

Після розробки структури вікон необхідно підібрати кольорову гаму інтерфейсу програми. Правильний підбір основних кольорів інтерфейсу надає можливість розробнику вплинути на емоції користувача, викликати в нього довіру або зацікавленість, в той час, як неправильний може його роздратувати або відштовхнути від використання додатку.

Для вибору кольорової гами інтерфейсу користуються кругом кольорів Ітена, який зображено на рисунку 4.9 [65].

Рисунок 4.9 – Круг кольорів Ітена

В його склад входять 12 кольорів, які вважаються найбільш звичними і простими для сприйняття людиною. При виборі кольорової гамми використовують колірні моделі поєднання кольорів: монохромна, аналогова, комплементарна, здвоєна комплементарна та тріадна.

За основу кольорової гами було обрано монохромну колірну модель та синій, сірий та білий кольори, оскільки розроблюваний додаток буде працювати в ОС Windows 10, де саме ці кольори використовуються для основних елементів інтерфейсу за замовчуванням. Кольором тексту було обрано чорний колір, який добре видно на білому та синьому фоні. Таке поєднання кольорів є оптимальним для розроблюваного програмного продукту.

4.5. Розробка програмних модулів системи

Робота програми починається з виклику методу OnStartup(), який виконує створення усіх основних компонентів програми: моделей, представлень та представлень моделей (рис. 4.10).

```
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);

        ApplicationView app = new ApplicationView();
        ApplicationViewModel context = new ApplicationViewModel();
        app.DataContext = context;
        app.Show();
    }
}
```

Рисунок 4.10 – Метод OnStartup

Як можна помітити, в методі OnStartup, виконується створення основної моделі представлення програми – класу ApplicationViewModel. Він відповідає за створення усіх вікон програми та встановлює початкову сторінку (рис. 4.11).

```
public class ApplicationViewModel : ObservableObject
{
    private ICommand _changePageCommand;
    private IPageViewModel _currentPageViewModel;
    private List<IPageViewModel> _pageViewModels;


    public ApplicationViewModel()
    {
        // створення усіх існуючих viewModel
        PageViewModels.Add(new MainViewModel());
        PageViewModels.Add(new CreatePatientViewModel());
        PageViewModels.Add(new SelectPatientViewModel());
        PageViewModels.Add(new DistancesNormsViewModel());
        PageViewModels.Add(new SymmetryNormsViewModel());

        // задання початкової сторінки
        CurrentPageViewModel = PageViewModels[0];
    }
```

Рисунок 4.11 – Конструктор класу ApplicationViewModel

Цей клас містить метод ChangeViewModel, який викликається при натисненні пунктів меню програми і який встановлює відповідне поточне представлення (рис. 4.12).

```
private void ChangeViewModel(IPageViewModel viewModel)
{
    if (!PageViewModels.Contains(viewModel))
        PageViewModels.Add(viewModel);

    CurrentPageViewModel = PageViewModels
        .FirstOrDefault(vm => vm == viewModel);
}
```

Рисунок 4.12 – Метод ChangeViewModel

В якості моделі програми було створено класи Patient, Image, Distance та PatientDistance, які використовуються роботи з даними програми (рис. 4.13).

```
public class Patient
{
    public int PatientID { get; set; }
    public string PatientName { get; set; }
    public string PatientPhone { get; set; }
    public string PatientSex { get; set; }
    public string PatientEmail { get; set; }
    public DateTime PatientDateOfBirth { get; set; }
    public double PatientCircumferance { get; set; }
    public double PatientSymmetryLevel { get; set; }
    public string PatientDiagnosis { get; set; }
    public ICollection<Image> Image { get; set; }
    public ICollection<PatientDistance> PatientDistance { get; set; }
}
public class Image
{
    public int ImageID { get; set; }
    public string ImagePath { get; set; }
    public DateTime ImageDate { get; set; }
    public int ImageType { get; set; }
    public Patient Patient { get; set; }
}
public class Distance
{
    public int DistanceID { get; set; }
    public string DistanceName { get; set; }
    public double DistanceValue { get; set; }
    public double DistanceMin { get; set; }
    public double DistanceMax { get; set; }
    public ICollection<PatientDistance> PatientDistance { get; set; }
}
public class PatientDistance
{
    public Patient Patient { get; set; }
    public Distance Distance { get; set; }
}
```

Рисунок 4.13 – Класи моделі програми

Як уже було сказано раніше, для роботи з зображеннями використовується бібліотека Emgu CV. Вона містить ряд зручних інструментів для обробки зображень. Так, наприклад, в ній вже присутня реалізація методу ORB для виявлення ключових точок зображення у вигляді класу ORBDetector та його методу DetectAndCompute (рис. 4.14).

```
public void DetectAndCompute(
    IInputArray image,
    IInputArray mask,
    VectorOfKeyPoint keyPoints,
    IOutputArray descriptors,
    bool useProvidedKeyPoints
)
```

Рисунок 4.14 – Метод DetectAndCompute

Метод приймає зображення та необов’язкову маску і повертає масиви ключових точок та дескрипторів.

Використовуючи цей клас, було розроблено метод DetectSymmetryAxis, який знаходить вісь симетрії обличчя людини. (рис. 4.15).

```
public static List<Point> DetectSymmetryAxis(Image<Bgr, byte> image)
{
    Mat mimage = CvInvoke.Flip(image, 2);
    ORBDetector detector = new ORBDetector();
    IOutputArray des1, des2;
    double theta, r, mIJ;
    VectorOfKeyPoint kp1 = detector.DetectAndCompute(image, des1, false);
    VectorOfKeyPoint kp2 = detector.DetectAndCompute(mimage, des2, false);
    ConvertDegreesToRadians(kp1);
    ConvertDegreesToRadians(kp2);
    BFMatcher matcher = new BFMatcher(DistanceType.Hamming2);
    VectorOfVectorOfDMatch matches = matcher.KnnMatch(des1, des2, 2);

    Array houghr, houghth, weights = Array.Clear(matches, 0, matches.Length);
    List<VectorOfDMatch> good = new List<VectorOfDMatch>();
    FindMatches(good, kp1, kp2, matches, theta, r, mIJ);
    List<Point> result = new List<Point>();
    foreach(Point point in good)
    {
        if ((Math.PI / 4 < theta) && (theta < 3 * Math.PI / 4))
        {
            Point rPoint = new Point();
            rPoint.X = Convert.ToInt32(r - point.Y * Math.Sin(theta) / Math.Cos(theta));
            rPoint.Y = Convert.ToInt32(r - point.X * Math.Cos(theta) / Math.Sin(theta));
            result.Add(rPoint);
        }
    }
    return result;
}
```

Рисунок 4.15 – Метод DetectSymmetryAxis

Цей метод повертає масив точок отриманої осі симетрії на зображенні. Після його виконання викликається метод DetectSkin для виявлення областей обличчя для порівняння (рис. 4.16).

```
public static Mat DetectSkin(string imagePath)
{
    Mat result;
    Mat img = CvInvoke.Imread(imagePath, Emgu.CV.CvEnum.ImreadModes.AnyColor);
    Mat img_HSV, HSV_mask, yCrCb_mask;
    CvInvoke.CvtColor(img, img_HSV, Emgu.CV.CvEnum.ColorConversion.Rgb2Hsv);
    CvInvoke.InRange(img_HSV, new ScalarArray(new MCvScalar(0, 15, 0)), new ScalarArray(new MCvScalar(17, 170, 255)), HSV_mask);
    CvInvoke.MorphologyEx(HSV_mask, HSV_mask, Emgu.CV.CvEnum.MorphOp.Open, 1);

    Mat img_YcrCb = CvInvoke.CvtColor(img, Emgu.CV.CvEnum.ColorConversion.YCrCb2Rgb);
    CvInvoke.InRange(img_YcrCb, new ScalarArray(new MCvScalar(0, 15, 0)), new ScalarArray(new MCvScalar(17, 170, 255)), yCrCb_mask);
    CvInvoke.MorphologyEx(yCrCb_mask, yCrCb_mask, Emgu.CV.CvEnum.MorphOp.Open, 1);

    Mat globalMask = CvInvoke.BitwiseAnd(yCrCb_mask, HSV_mask);
    globalMask = CvInvoke.MedianBlur(globalMask, 3);
    globalMask = CvInvoke.MorphologyEx(globalMask, globalMask, Emgu.CV.CvEnum.MorphOp.Open, 1);

    result = CvInvoke.BitwiseNot(globalMask);
    return result;
}
```

Рисунок 4.16 – Метод DetectSkin

Отримані області порівнюються за допомогою міри NMSE, яку реалізовує метод GetSymmetryLevel (рис. 4.17).

```
public static double GetSymmetryLevel(List<Pixel> area1, List<Pixel> area2)
{
    double NMSE = 0;
    var values = area1.Zip(area2, (a1, a2) => new { A1 = a1, A2 = a2 });

    int rx = 0, gx = 0, bx = 0, a1Total = 0;
    foreach(var value in values)
    {
        rx += Math.Pow(value.A1.R - value.A2.R, 2);
        gx += Math.Pow(value.A1.G - value.A2.G, 2);
        bx += Math.Pow(value.A1.B - value.A2.B, 2);
        a1Total = Math.Pow(value.A1.R, 2) + Math.Pow(value.A1.G, 2) + Math.Pow(value.A1.B, 2);
    }
    NMSE = (rx + gx + bx) / a1Total;

    return NMSE;
}
```

Рисунок 4.17 – Метод GetSymmetryLevel

Для визначення характерних точок на обличчі людини було розроблено метод DetectFacialLandmarks, який використовує модель «trainedModel.yaml», отриману в результаті обробки бази даних Multi-PIE (рис. 4.18).

```
public static Image<Bgr, byte> DetectFacialLandmarks(Image<Bgr, byte> img)
{
    String facemarkFileName = "trainedModel.yaml";
    LoadModel(facemarkFileName);

    using (FacemarkLBFParams facemarkParam = new CV.Face.FacemarkLBFParams())
    using (FacemarkLBF facemark = new CV.Face.FacemarkLBF(facemarkParam))
    using (VectorOfRect vr = new VectorOfRect(faceRegions.ToArray()))
    using (VectorOfVectorOfPointF landmarks = new VectorOfVectorOfPointF())
    {
        facemark.LoadModel(facemarkFileName);
        facemark.Fit(img, vr, landmarks);

        foreach (Rectangle face in faceRegions)
        {
            CvInvoke.Rectangle(img, face, new MCvScalar(0, 255, 0));
        }

        int len = landmarks.Size;
        for (int i = 0; i < landmarks.Size; i++)
        {
            using (VectorOfPointF vpf = landmarks[i])
                FaceInvoke.DrawFacemarks(img, vpf, new MCvScalar(255, 0, 0));
        }

    }
    return img;
}
```

Рисунок 4.18 – Метод DetectFacialLandmarks

Аналіз відповідності форми голови дитини нормованим значенням відбувається в декілька кроків. Спочатку, виконується вилучення фону зображення для встановлення області вимірювання. Цей функціонал реалізовує метод RemoveBackground (рис. 4.19).

```
public static Image<Bgr, byte> RemoveBackground(Image<Bgr, byte> source)
{
    Image<Gray, byte> mask = new Image<Gray, byte>(source.Width + 2, source.Height + 2);
    MCvConnectedComp comp = new MCvConnectedComp();
    //центр зображення в якості початкової точки
    Point point1 = new Point(source.Width / 2, source.Height / 2);

    CvInvoke.cvFloodFill(source, point1, new MCvScalar(255), new MCvScalar(50, 30, 10),
                         new MCvScalar(50, 30, 10), out comp, 8 | (1 << 17) | (255 << 8), mask);

    Image<Gray, byte> reducedMask = mask.Copy(new Rectangle(1, 1, source.Width, source.Height));
    Image<Gray, byte> morphImg = new Image<Gray, byte>(source.Size);
    StructuringElementEx element = new StructuringElementEx(3, 3, 1, 1, Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_CROSS);

    CvInvoke.cvMorphologyEx(reducedMask, morphImg, IntPtr.Zero, element, CV_MORPH_OP.CV_MOP_CLOSE, 4);
    Image<Bgr, Byte> imageROI = source.Copy(morphImg);

    return imageROI;
}
```

Рисунок 4.19 – Метод RemoveBackground

Після вилучення фону знаходяться крайні точки контурів, по яких обраховується обхват голови. Порівняння отриманого значення обхвату голови з нормованим значенням виконує метод AnalyzeCircumference (рис. 4.20).

```
public static int AnalyzeCircumference(double value, double age, int sex)
{
    double x3 = Math.Pow(age, 3);
    double x2 = Math.Pow(age, 2);
    double x = age;
    // 1 - хлопець, 2 - дівчина
    if(sex == 1)
    {
        double SD3 = 0.0082 * x3 - 0.237 * x2 + 2.8406 * x + 35.897;
        double nSD3 = 0.0081 * x3 - 0.2457 * x2 + 2.9261 * x + 28.518;
        double SD2 = 0.0082 * x3 - 0.2395 * x2 + 2.8611 * x + 34.651;
        double nSD2 = 0.008 * x3 - 0.2428 * x2 + 2.9114 * x + 29.696;
        double SD1 = 0.0082 * x3 - 0.2413 * x2 + 2.8912 * x + 33.359;
        double nSD1 = 0.0081 * x3 - 0.2415 * x2 + 2.8916 * x + 30.971;

        //1 - повністю здоровий, 2 - незначне відхилення, 3 - серйозне відхилення
        if (value >= nSD1 && value <= SD1) return 1;
        if (value > SD2 && value < SD3 && value < nSD2 && value > nSD3) return 2;
        if (value >= SD3 || value <= nSD3) return 3;
    }
    else
    {
        double SD3 = 0.0083 * x3 - 0.24 * x2 + 2.8443 * x + 35.113;
        double nSD3 = 0.0081 * x3 - 0.2308 * x2 + 2.6587 * x + 28.257;
        double SD2 = 0.0083 * x3 - 0.2382 * x2 + 2.8109 * x + 33.959;
        double nSD2 = 0.0081 * x3 - 0.2313 * x2 + 2.6817 * x + 29.428;
        double SD1 = 0.008 * x3 - 0.2293 * x2 + 2.723 * x + 32.935;
        double nSD1 = 0.008 * x3 - 0.2308 * x2 + 2.6944 * x + 30.599;

        if (value >= nSD1 && value <= SD1) return 1;
        if (value > SD2 && value < SD3 && value < nSD2 && value > nSD3) return 2;
        if (value >= SD3 || value <= nSD3) return 3;
    }
    return 0;
}
```

Рисунок 4.20 – Метод AnalyzeCircumference

Отже, було завершено розробку основних програмних модулів програми для медичного експрес-діагностування. Лістинг решти розроблених програмних модулів наведено в додатку Б.

4.6. Вибір методики тестування

Тестуванням називають процес перевірки відповідності між реальною та очікуваною поведінкою програми, який здійснюють на кінцевому наборі тестів, які були обрані за певним підходом. Воно надає можливість отримати об’єктивне та незалежне уявлення про програмне забезпечення, щоб оцінити ризики його реалізації [66]. Тестування включає в себе виконання цілого ряду етапів, таких як: планування видів роботів, проведення проектування та виконання тестів, а

також аналіз отриманих результатів. Існують різні види класифікацій видів тестування, але загалом його можна поділити на статичне та динамічне.

Статичним називають тестування, при якому код програми не виконується. Під час тестування перевіряються не результати роботи програми, а сам код. Статичне тестування можна проводити вручну або за допомогою спеціальних програм для аналізу коду. Динамічне тестування передбачає виконання коду програми.

Виділяють три основних підтипи динамічного тестування [67]:

1. Тестування методом «чорної скриньки».
2. Тестування методом «білої скриньки».
3. Тестування методом «сірої скриньки».

Тестуванням методом «чорної скриньки» називають тестування, при якому особа, яка перевіряє програму, має доступ до неї тільки через інтерфейс. При використанні цього підходу програма розглядається як чорна скринька, поведінку якої досліджується шляхом аналізу її початкових та кінцевих даних. Застосування цього методу базується на використанні специфікації, документації та описах інтерфейсу програмного забезпечення або системи. Основними перевагами цього методу є ефективність для великих сегментів коду, простота у сприйнятті та висока швидкість проведення.

Тестування методом «білої скриньки» – це тестування, при якому тестувальник має доступ до коду програми [68]. Під час такого тестування виконується перевірка не зовнішньої, а внутрішньої поведінки програми. При використанні цього методу перевіряється коректність побудови всіх елементів програми та правильність їх взаємодії між собою. Зазвичай, тестувальник аналізує керуючі зв’язки елементів, хоча інколи і може перевіряти інформаційні зв’язки. Результати тестування методом «білої скриньки» характеризуються ступенем, в якому розроблені тести покривають вихідний текст програми. Вона вважається повністю перевіреною, якщо було проведено вичерпне тестування маршрутів її графу управління. Перевагою цього методу є те, що він дає можливість виявити помилки програми, які залежать не від вхідних даних, а від

внутрішніх станів системи, чого не можна досягнути при використанні методу «чорної скриньки» Крім того, це тестування можна проводити на ранніх етапах розробки, за відсутності користувацького інтерфейсу. Основними недоліками цього методу є значні витрати на проведення тестування, високі вимоги до кваліфікації тестувальника та ймовірність пропуску помилок.

В результаті поєднання методів «білої та чорної скриньки» з'явилось тестування «сірої скриньки». Його суть полягає в тому, що тестувальник знає частину внутрішньої структури програми, але проводить тестування з позиції користувача. Цей метод часто використовують для тестування веб-додатків.

Враховуючи особливості кожного з розглянутих методів, для тестування розробленого додатку було вирішено використати метод «чорної скриньки», оскільки воно найкраще підходить для тестування продуктів, для яких є чіткі вимоги до функціоналу.

4.7. Тестування розробленого програмного додатку

В якості методу тестування розробленого програмного додатку було обрано метод «чорної скриньки», який базується на виконанні заданих сценаріїв, які перевіряють роботу певної частини програми. Ці сценарії називають тест-кейсами.

При запуску програми відкривається початкове вікно програми, яке зображено на рисунку 4.21.

Рисунок 4.21 – Початкове вікно програми

Робота з програмою починається з цього вікна, тому і тестування буде починатись також з нього.

Для тестування створеного додатку було розроблено набір тест-кейсів, які перевіряють роботу основних режимів роботи програми. Розглянемо деякі з них.

Тест-кейс №1 – Створення нового пацієнта.

1. Запустити програму.
2. Обрати пункт меню «Пацієнт – Створити».
3. Заповнити усі наявні поля.
4. Натиснути кнопку «Створити».
5. Переглянути вміст діалогового вікна, що з’явилось.

Очікуваний результат – на екрані відображається повідомлення «Пацієнт був успішно створений».

Результат виконання тест-кейсу №1 зображено на рисунку 4.22.

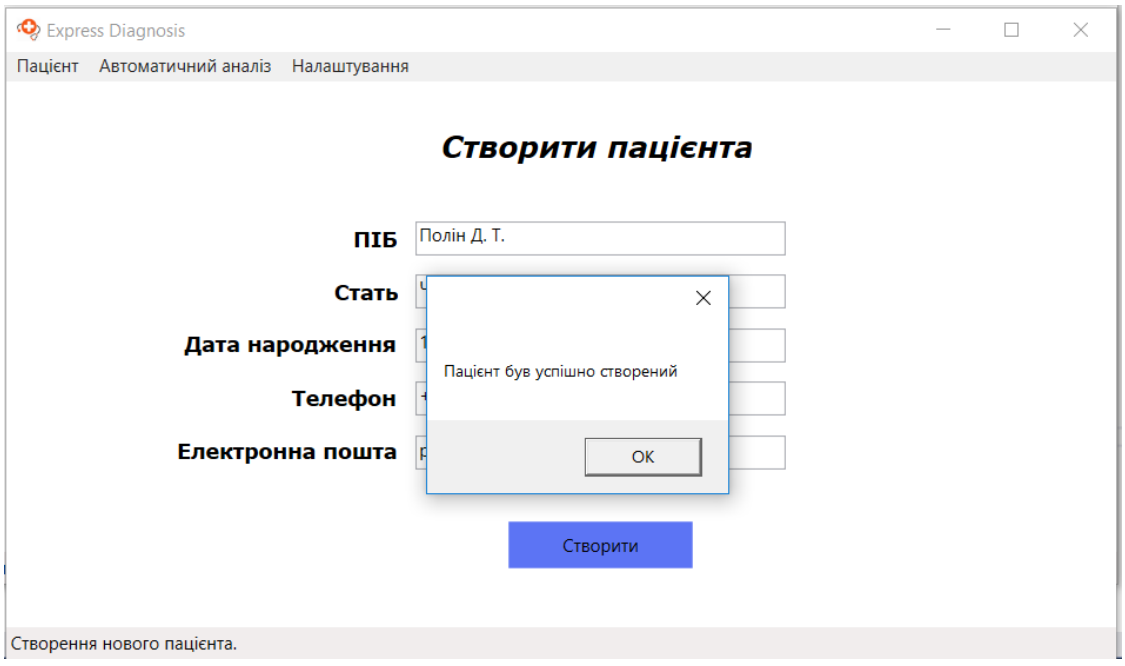


Рисунок 4.22 – Результат виконання тест-кейсу №1

Отриманий результат відповідає очікуваному, отже тест-кейс №1 успішно пройдено.

Тест-кейс №2 – Видалення існуючого пацієнта.

1. Запустити програму.
2. Обрати пункт меню «Пацієнт – Обрати».
3. Обрати будь-якого пацієнта.
4. Натиснути кнопку «Видалити».
5. Переглянути вміст таблиці з існуючими пацієнтами.

Очікуваний результат – у таблиці відсутній обраний раніше пацієнт.

Результат виконання тест-кейсу №2 зображено на рисунку 4.23.

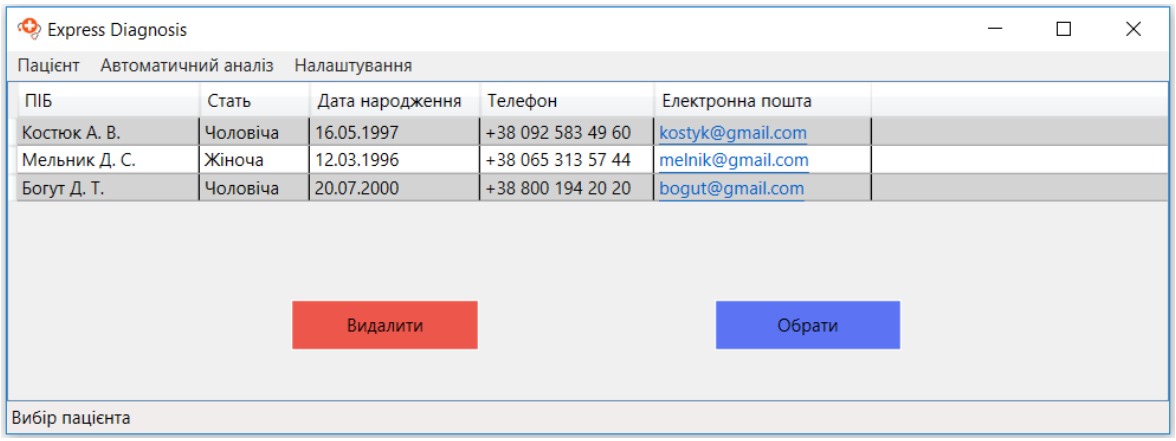


Рисунок 4.23 – Результат виконання тест-кейсу №2

Отриманий результат відповідає очікуваному – видалений пацієнт відсутній у таблиці, отже тест-кейс №2 успішно пройдено.

Тест-кейс №3 – Перехід на головне вікно пацієнта.

1. Запустити програму.
2. Обрати пункт меню «Пацієнт – Обрати».
3. Виділити будь-якого пацієнта.
4. Натиснути кнопку «Обрати».
5. Проаналізувати вміст поточного вікна.

Очікуваний результат – відображається головне вікно пацієнта.

Результат виконання тест-кейсу №3 зображено на рисунку 4.24.

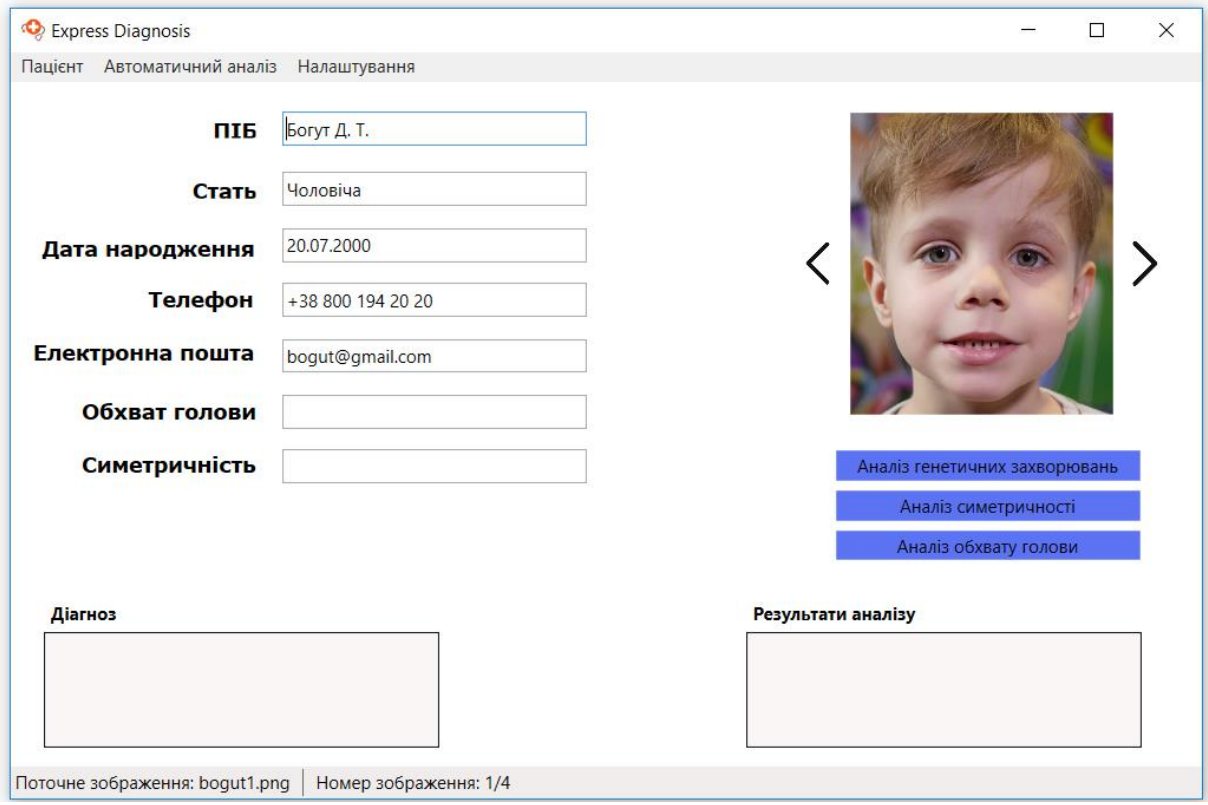


Рисунок 4.24 – Результат виконання тест-кейсу №3

Отриманий результат відповідає очікуваному, отже тест-кейс №3 успішно пройдено.

Тест-кейс №4 – Аналіз генетичних захворювань.

1. Запустити програму.
2. Обрати пункт меню «Пацієнт – Обрати».
3. Виділити будь-якого пацієнта.
4. Натиснути кнопку «Обрати».
5. Обрати пункт меню «Пацієнт – Додати зображення».
6. Завантажити зображення обличчя людини з відомим діагнозом.
7. Натиснути кнопку «Аналіз генетичних захворювань».
8. Переглянути вміст поля «Результати аналізу».

Очікуваний результат – результати аналізу відповідають діагнозу пацієнта.

Результат виконання тест-кейсу №4 зображено на рисунку 4.25.

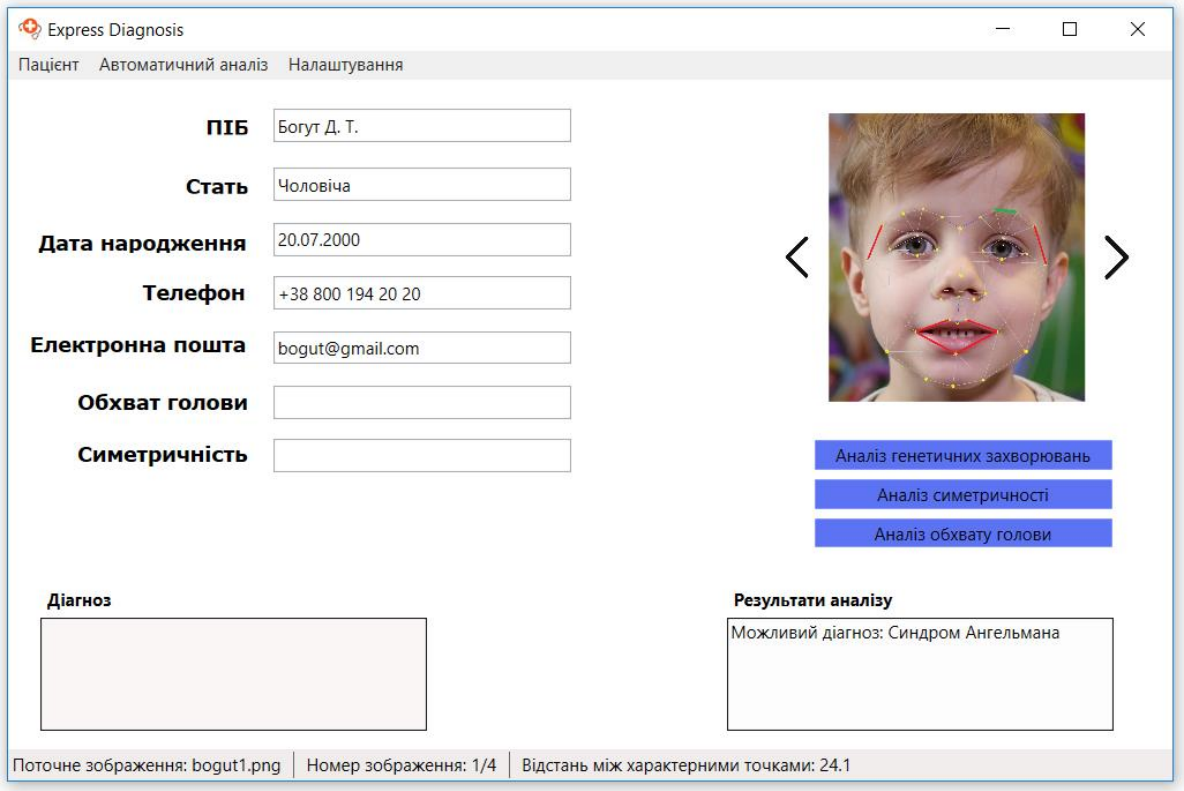


Рисунок 4.25 - Результат виконання тест-кейсу №4

Отриманий результат відповідає очікуваному, результати аналізу та діагноз пацієнта співпадають, отже тест-кейс №4 успішно пройдено.

Тест-кейс №5 – Аналіз симетричності обличчя.

1. Запустити програму.
2. Обрати пункт меню «Пацієнт – Обрати».
3. Виділити будь-якого пацієнта.
4. Натиснути кнопку «Обрати».
5. Обрати пункт меню «Пацієнт – Додати зображення».
6. Завантажити зображення обличчя людини з явною асиметрією.
7. Натиснути кнопку «Аналіз симетричності».
8. Переглянути вміст поля «Результати аналізу».

Очікуваний результат – результати аналізу показують високий рівень асиметрії.

Результат виконання тест-кейсу №5 зображено на рисунку 4.26.

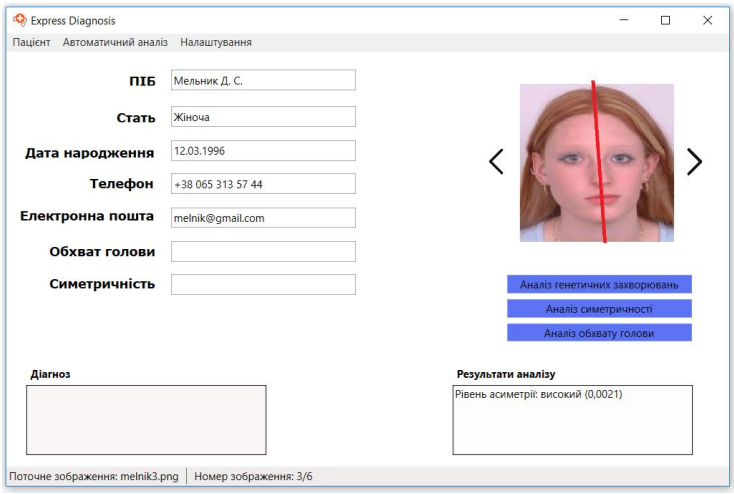


Рисунок 4.26 – Результат виконання тест-кейсу №5

Отриманий результат відповідає очікуваному, результати аналізу та діагноз пацієнта співпадають, отже тест-кейс №5 успішно пройдено.

Тест-кейс №6 – Аналіз обхвату голови.

1. Запустити програму.
2. Обрати будь-якого пацієнта.
3. Ввести дату народження та значення обхвату голови..
4. Натиснути кнопку «Аналіз обхвату голови».
5. Переглянути вміст поля «Результати аналізу».

Очікуваний результат – результати аналізу відповідають значенню, визначеному за допомогою нормованих таблиць.

Результат виконання тест-кейсу №6 зображено на рисунку 4.27.

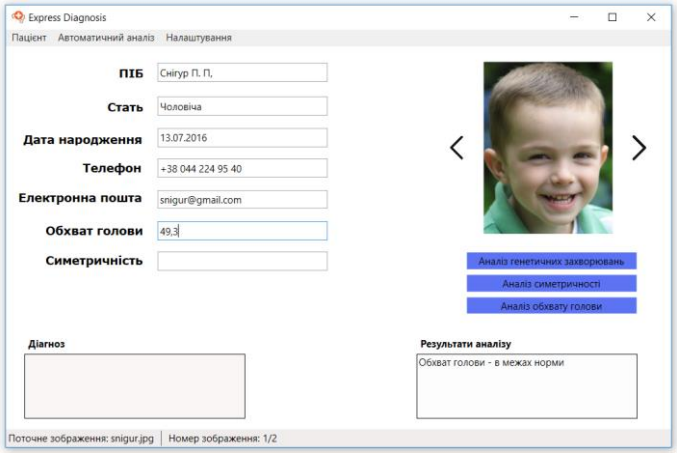


Рисунок 4.27 – Результат виконання тест-кейсу №6

Отриманий результат відповідає очікуваному, отже тест-кейс №6 успішно пройдено.

Крім описаних вище тест-кейсів, було розроблено та виконано ще ряд тестів, які перевіряють роботу модулів програми. Усі вони були успішно пройдені, у зв’язку з чим можна зробити висновок, що програма функціонує нормально і є придатною для використання.

4.8. Висновки

Отже, було проведено аналіз варіантний аналіз та обґрунтування вибору програмних засобів для розробки, в результаті якого було прийнято рішення використовувати мову програмування C#, технологію WPF для розробки інтерфейсу, а також бібліотеку EMGU CV та Entity Framework.

Було проведено аналіз існуючих середовищ розробки для мови C#, в результаті якого було підтверджено доцільність використання середовища Microsoft Visual Studio 2017, як найбільш оптимального.

Також, було розроблено структуру бази даних, структурну схему додатку та графічні схеми інтерфейсу головних вікон. Було розроблено програмні модулі, які забезпечують роботу програмного додатку та реалізовують запропоновані раніше методи. Крім цього, було проведено тестування роботи розробленого додатку за допомогою методу «чорної скриньки», в ході якого не було виявлено жодних помилок, що свідчить про стабільність та придатність до використання додатку для проведення медичного експрес-діагностування.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1. Технологічний аудит розроблених методів і засобів морфологічного аналізу зображень облич людей

Як було підкреслено у попередніх розділах роботи, за останні декілька десятиліть спостерігається значний стрибок у розвитку медицини, і однією із причин цього є її зв'язок з інформаційними технологіями. Сучасні інформаційні технології допомагають раціоналізувати працю медичних працівників, підвищують ефективність діяльності лікувально-профілактичних закладів та виводять систему охорони здоров'я на новий рівень як по обсягу наданих послуг, так і по їх якості. Одним із напрямів застосування в медицині інформаційних технологій є медичне діагностування.

Медичне діагностування – це процес встановлення висновку про сутність хвороби пацієнта та стан його здоров'я. Для вирішення завдань медичного діагностування лікарі широко використовують відповідне допоміжне програмне забезпечення. Лікар, використовуючи програмне забезпечення, має можливість краще проаналізувати дані пацієнта, провести експрес-діагностування пацієнта та встановити йому попередній діагноз по зовнішнім характеристикам пацієнта.

Особливо актуальним в наш час є діагностування генетичних захворювань. Як було зазначено раніше, рідкісними генетичними захворюваннями страждають близько 8% людей, для значної частини яких характерними є зміни в обличчі та черепі. Тому багатьом генетичним захворюванням властиві певні характерні ознаки, які можна помітити на зображенні обличчя пацієнта. В результаті було доведено, що застосування медичного експрес-діагностування шляхом морфологічного аналізу зображення обличчя людини діє змогу своєчасно виявляти ті чи інші генетичні захворювання.

Також, було встановлено, що важливу роль у вивченні стану здоров'я людини відіграє ступінь симетричності/асиметричності її обличчя, тому симетрія обличчя людини є важливим фактором при діагностуванні низки захворювань.

У зв’язку з цим, сьогодні багатьма дослідниками ведеться робота з розробки програмного забезпечення, яке б удосконалювало методи діагностування захворювань на основі морфологічного аналізу зображень облич людей. Разом з тим, аналіз існуючих методів, проведених нами у попередніх розділах роботи, показав, що більшості з них властиві певні недоліки. Тому було прийнято рішення модифікувати декілька існуючих методів, а також створити власний програмний додаток, який буде їх реалізовувати.

В результаті, на основі аналізу візуальних характеристик обличчя людини було розроблено декілька нових методів, які дозволяють підвищити швидкодію визначення типу захворювання та проводити високоефективне медичне експрес-діагностування.

Для оцінювання технічного рівня та комерційного потенціалу нашої розробки проведемо її технологічний аудит. Для проведення аудиту запросимо 3-х експертів, які є відомими фахівцями в цій галузі знань: д.т.н., професора Романюка О. Н.; к.т.н., доцента Колодного В. В. та к.т.н., доцентку Черноволик Г.О.. Фахівці здійснювали оцінювання технічного рівня та комерційного потенціалу нашої розробки за методикою, [69], наведеною в таблиці 5.1.

Таблиця 5.1

Критерії оцінювання технічного рівня та комерційного потенціалу розробки та їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатацій-ні витрати значно вищі, ніж в аналогів	Експлуатацій-ні витрати дещо вищі, ніж в аналогів	Експлуатацій-ні витрати на рівні Експлуатацій-них витрат аналогів	Експлуатацій ні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовую ться у військово-промисловом у комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовую-ться у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати проведеного запрошеними експертами технологічного аудиту нашої розробки зведено в таблицю 5.2.

Таблиця 5.2

Результати технологічного аудиту нашої розробки

Критерії	Прізвище, ініціали експерта		
	Романюк О.М.	Колодний В.В.	Черноволик Г.О.
	Бали, виставлені експертами:		
1	3	3	3
2	4	4	4
3	4	4	4
4	4	4	4
5	4	4	4
6	3	3	3
7	4	4	3
8	4	3	3
9	3	3	3
10	4	4	4
11	4	4	4
12	3	3	3
Сума балів	$СБ_1$ =44	$СБ_2$ = 43	$СБ_3$ = 42

Далі розрахуємо середньоарифметичну суму балів, що їх виставили запрошені експерти:

Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{44+43+42}{3} = \frac{129}{3} = 43$

На підставі рекомендацій, наведених в [69] (див. табл. 5.3), можна зробити висновок, що розроблені нами методи і засоби морфологічного аналізу зображень обличь людей мають технічний рівень та комерційний потенціал, який є «високим» (43 бали).

Таблиця 5.3

Технічні рівні та комерційний потенціал розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Технічний рівень та комерційний потенціал розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Такий високий технічний рівень та комерційний потенціал нашої розробки пояснюється тим, що нами було здійснено модифікацію методу Г. Лоя для визначення осі симетрії обличчя людини, а також розроблено власні методи діагностування на основі морфологічного аналізу зображень облич людей, що значно підвищує точність та швидкодію проведення експрес-діагностування.

5.2. Розрахунок витрат на розробку методів і засобів морфологічного аналізу зображень облич людей

Витрати на розробку методів і засобів морфологічного аналізу зображень обличь людей складаються з таких основних статей:

а). Основна заробітна плата $З_о$ виконавців:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн.}, \qquad (5.1)$$

де М – місячний посадовий оклад конкретного виконавця, грн.

$Т_р$ – число робочих днів в місяці; приймемо $Т_р = 20$ днів;

t – число робочих днів роботи виконавців.

Зроблені розрахунки основної заробітної плати зведемо до таблиці 5.4:

Таблиця 5.4

Основна заробітна плата виконавців

Найменування посади виконавця	Місяч-ний посадо-вий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн	Примітка
1. Науковий керівник магістерської роботи	15000	750	25 годин	3125	-
2. Магістрант	2000	100	60	6000	-
3. Консультант з ЕЧ	12100	605	2,5 год.	≈ 253	-
4. Інші	8000	400	1	400	
Всього				$З_о$ = 9778 грн	

б). Додаткова заробітна плата $З_д$ виконавців розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о . \quad (5.2)$$

Для нашого випадку отримаємо:

$$З_д = 0{,}11 \times 9778 \approx 1076 \text{ грн.}$$

в). Нарахування на заробітну плату $Н_{зп}$ виконавців:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \quad (5.3)$$

де β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %. β = 22%. Тоді:

$Н_{зп}$ = (9778+1076) × 0,22 = 2388 грн.

г). Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час виконання даної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \quad (5.4)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання роботи, грн.;

$Н_а$ – річна норма амортизаційних відрахувань. $Н_а$= (2,5...25)%;

Т – термін, використання обладнання, приміщень тощо, місяці.

Зроблені розрахунки зведемо у таблицю 5.5.

Таблиця 5.5

Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, місяці	Величина амортизаційних відрахувань, грн
1. Обладнання: комп'ютери, принтери тощо	51000	18	3 (25% використання)	≈574
2. Приміщення університету та кафедри	37000	3,3	3 (25% використання)	76,31 ≈ 77
Всього				А = 651 грн

д). Витрати на матеріали М:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \quad грн., \qquad (5.5)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг; $Ц_i$ – вартість матеріалу *i*-го найменування, грн./кг.; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *i*-го найменування, кг; $Ц_в$ – ціна відходів матеріалу *i*-го найменування, грн/кг; n – кількість видів матеріалів.

е) Витрати на комплектуючі К:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \quad грн., \qquad (5.6)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.; $Ц_i$ – ціна комплектуючих *i*-го виду, грн; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Загальна вартість основних матеріалів та комплектуючих, які були використані під час виконання цієї роботи, складає приблизно 650 грн.

ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2019 р. В ≈ 2,2 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,30 кВт;

Ф – фактична кількість годин роботи обладнання, годин. Приймемо, що Ф = 150 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ = 0,91.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,79.

Тоді витрати на силову електроенергію складуть:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{2,2 \cdot 1,3 \cdot 150 \cdot 0,91}{0,79} = 494,16 \approx 495 \text{ грн.}$$

і). Інші витрати $В_{ін}$ можна прийняти як (100…300)% від суми основної заробітної плати виконавців, тобто:

$$В_{ін} = (1..3) \times З_о. \quad (5.8)$$

Для нашого випадку отримаємо:

$$В_{ін} = 1,5 \times 9778 = 14667 \text{ грн.}$$

к). Сума всіх попередніх статей витрат дає витрати на виконання цього етапу роботи магістрантом – В.

В = 9778+ 1076 + 2388 + 651 + 650 + 495 + 14667 = 29705 грн.

л). Загальні витрати ЗВ на розробку та можливе впровадження результатів виконаної нами роботи розраховуються за формулою:

$$ЗВ = \frac{В}{\beta}, \quad (5.9)$$

де β – коефіцієнт, який характеризує етап виконання роботи на шляху до її можливого впровадження. Оскільки наша розробка потребує певного доопрацювання, то згідно з [69, 70] $\beta \approx 0{,}6$.

Тоді: $ЗВ = \frac{29705}{0,6} = 49508,22$ грн або приблизно 50 тис. грн.

Тобто прогнозовані загальні витрати на виконання та завершення нашої роботи становлять приблизно 50 тис. грн.

5.3. Розрахунок економічного ефекту від можливого впровадження розроблених методів і засобів морфологічного аналізу зображень облич людей

Аналіз місткості ринку даної продукції показує, що на сьогодні кількість реальних користувачів подібних розробок (а це переважно медичні заклади) коливається в межах N= 20-100 осіб. Для орієнтиру приймемо N=50. Разом з тим, не виникає сумніву, що кількість зацікавлених осіб у придбанні подібних розробок буде стрімко зростати (приватні медичні клініки, навчальні медичні заклади, приватні особи тощо).

На сьогодні ціна подібних розробок становить приблизно 20 тис. грн. Оскільки розроблені нами методи і засоби морфологічного аналізу зображень обличь людей мають значно кращі функціональні характеристики (значно вища швидкодія, висока точність, низька вартість тощо), то нашу розробку можна реалізовувати на ринку дорожче, наприклад за 21 тис. грн. Але ми не будемо цього робити. Навпаки, для подолання опору конкурентів ми можемо реалізовувати нашу розробку трохи дешевше, наприклад, за 19 тис. грн., що автоматично збільшить попит на нашу розробку.

Практично нашу розробку можна буде впровадити з 2021 року, і користуватися попитом на ринку наша розробка буде не менше 3-х років після впровадження. Прогноз зростання попиту на нашу розробку складе по роках:

1-й рік після впровадження (2021 р.) – приблизно на 50 шт.;

2-й рік після впровадження (2022 р.) – приблизно на 75 шт.;

3-й рік після впровадження (2023 р.) – приблизно на 100 шт.

Можливе збільшення чистого прибутку $\Delta П_i$, що його можна отримати потенційний інвестор від впровадження нашої розробки, розраховується за формулою [70]:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

де $\Delta Ц_o$ – зміна основного якісного показника від впровадження нашої розробки.

Таким показником є зміна ціни реалізації нової розробки, тобто

$$\Delta Ц_o = (19 - 20) = -1 \text{ тис. грн;}$$

N – основний кількісний показник, який визначає обсяг діяльності у цьому році до впровадження результатів розробки; N = 50 шт.;

ΔN – покращення основного кількісного показника від збільшення попиту на розробку. Як було зазначено вище, таке збільшення становить, відповідно по роках, +50, +75 та +100 шт.;

$Ц_o$ – основний якісний показник, який визначає обсяг діяльності (тобто ціну реалізації) у році після впровадження результатів розробки, грн.

$Ц_o$ = 19 тис. грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; n = 3 роки;

λ – коефіцієнт, який враховує сплату податку на додану вартість

$\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = (0{,}2...0{,}5)$; візьмемо $\rho = 0{,}5$;

υ – ставка податку на прибуток. У 2019 році υ = 18%.

Величина чистого прибутку $\Delta П_1$, що його може отримати потенційний інвестор протягом першого року після реалізації нашої розробки (2021 р.):

$$\Delta П_1 = [-1 \cdot 50 + 19 \cdot 50] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 307{,}49 \approx 308 \text{ тис. грн.}$$

Величина можливого чистого прибутку $\Delta П_2$ для потенційного інвестора від реалізації нашої розробки протягом другого (2022 р.) року складе:

$$\Delta П_2 = [-1 \cdot 50 + 19 \cdot 75] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 469{,}77 \approx 470 \text{ тис. грн.}$$

Величина можливого чистого прибутку $\Delta П_3$ для потенційного інвестора протягом третього (2023 р.) року складе:

$$\Delta П_3 = [-1 \cdot 50 + 19 \cdot 100] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 632{,}01 \approx 632 \text{ тис. грн.}$$

Теперішня вартість інвестицій PV, що можуть бути вкладені в нашу розробку, становитиме: PV = (2…5) × ЗВ.

Для нашого випадку PV = 2,5 × 50 = 125 тис. грн.

Розрахуємо абсолютний ефект вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \qquad (5.11)$$

де ПП – приведена вартість всіх чистих прибутків від можливого впровадження нашої розробки, грн:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки; т = 3 роки;

τ – ставка дисконтування (або прогнозований рівень інфляції). Для наших розрахунків приймемо, що τ = 0,10 (або 10%);

t – період часу (в роках) від моменту отримання прибутків до початку впровадження розробки.

Тоді приведена вартість всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження нашої розробки, складе:

$$ПП = \frac{308}{(1+0,1)^2} + \frac{470}{(1+0,1)^3} + \frac{632}{(1+0,1)^4} \approx 255 + 351 + 475 = 1081 \text{ тис. грн.}$$

Абсолютний ефект ід впровадження нашої розробки за 3 роки може становити::

$Е_{абс}$ = 1081 –125 = 956 тис. грн. або 318,67 тис. грн щороку.

Далі розрахуємо відносну ефективність $Е_в$ вкладених у розробку інвестицій. Для цього скористаємося формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 1081 тис. грн;

PV – теперішня вартість початкових інвестицій PV = 125 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки. $Т_ж$ = 5 років (2019-2023 рр.).

Для нашого випадку:

$$E_в = \sqrt[5]{1+\frac{1081}{125}} - 1 = \sqrt[5]{1+8,648} - 1 = \sqrt[5]{9,648} - 1 = 1,5735 - 1 = 0,5735 = 57,35\%.$$

Визначимо ту мінімальну дохідність, нижче за яку кошти в нашу розробку вкладатися не будуть.

Мінімальна дохідність $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = (0,10...0,15);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5), але може бути і значно більше.

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,15 + 0,4 = 0,55 \text{ або } \tau_{мін} = 55\%.$$

Оскільки величина $Е_в$ =57,35% > $\tau_{мін}$ = 55%, то інвестор у принципі може бути зацікавлений у фінансуванні нашої розробки.

Далі розраховуємо термін окупності коштів, які можуть бути вкладені у впровадження нашої розробки:

$$Т_{ок} = \frac{1}{Е_в}. \qquad (5.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів, вкладених у впровадження нашої розробки, складе:

$$T_{ок} = \frac{1}{0,5735} \approx 1,743 \text{ років},$$

що свідчить про потенційну комерційну привабливість нашої розробки, тобто розроблених нами методів і засобів морфологічного аналізу зображень облич людей.

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю 5.6:

Таблиця 5.6

Результати виконання економічної частини

Показники	Вимоги	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку методів і засобів морфологічного аналізу зображень облич людей.	Не більше 50 тис. грн	50 тис. грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки (щорічний), тис. грн.	не менше 300 тис. грн	318,67 тис. грн щороку	Досягнуто
3. Внутрішня норма дохідності інвестицій, %	не менше 55%	57,35%	Досягнуто
4. Термін окупності інвестицій, роки	до 3-х років	1,743 років	Досягнуто

Таким чином, основні техніко-економічні характеристики розроблених нами методів і засобів морфологічного аналізу зображень обличь людей для медичного експрес-діагностування, виконані.

5.4. Висновки

Отже, було проведено технологічний аудит розроблених методів і засобів морфологічного аналізу зображень облич людей, в ході якого було встановлено високий технічний рівень та комерційний потенціал нашої розробки за рахунок розроблених та модифікованих методів для медичного експрес-діагностування на основі морфологічного аналізу зображень облич людей.

Було виконано розрахунок витрат на розробку запропонованих методів та програмного забезпечення, яке їх реалізовує, в ході якого було встановлено, що вони складуть близько 49,508 тис. гривень, що не перевищує поставлені вимоги до розробки.

Було проведено розрахунок економічного ефекту від можливого впровадження розроблених методів і засобів, в ході якого було встановлено, що абсолютний ефект від впровадження розробки складе близько 318,67 тис. грн., а період окупності інвестицій – 1,743 роки, що свідчить про доцільність фінансування цієї наукової розробки.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розроблено методи засоби морфологічного аналізу зображень облич людей для медичного експрес-діагностування.

В результаті виконання кваліфікаційної роботи, було отримано наступні результати:

1. Проведено аналіз предметної галузі, в результаті якого було зроблено висновок про можливість медичного експрес-діагностування за допомогою морфологічного аналізу зображень облич людини та необхідність розробки власного методу, який буде його реалізовувати, та модифікації існуючого методу визначення осі симетрії обличчя людини.

2. Проведено аналіз існуючих аналогів, який підтвердив актуальність розробки власного програмного додатку, який буде нівелювати недоліки існуючих та реалізовувати розроблені методи.

3. Модифіковано метод Г. Лоя для визначення осі симетрії обличчя людини шляхом заміни методу визначення ключових точок SIFT на ORB, що дозволило значно підвищити швидкість та точність процесу пошуку осі симетрії обличчя людини.

4. Розроблено метод діагностування генетичних захворювань на основі аналізу візуальних характеристик обличчя людини, а саме вимірювання відстаней між характерними точками, які було запропоновано, та їх порівняння з нормованими значеннями, що дає можливість швидко визначити тип генетичного захворювання.

5. Отримано аналітичні залежності обхвату голови дитини від її віку з урахуванням допустимого інтервалу, що дає можливість розробки програмного забезпечення для діагностування шляхом порівняння параметрів голови дитини з допустимими нормами.

6. Розроблено структуру бази даних, структурну схему додатку та графічні схеми інтерфейсу головних вікон. Було розроблено програмний додаток

за допомогою мови програмування C#, технології WPF, а також бібліотеки EMGU CV та Entity Framework в середовищі Microsoft Visual Studio 2017.

7. Проведено тестування роботи розробленого програмного додатку, яке підтвердило його стабільність та придатність до використання в медичному експрес-діагностуванні.

8. Проведено технологічний аудит розроблених методів і засобів морфологічного аналізу зображень облич людей, в ході якого було встановлено високий технічний рівень та комерційний потенціал розробки. Також, виконано розрахунок економічного ефекту від можливого впровадження розроблених методів і засобів, в ході якого підтверджено доцільність фінансування цієї наукової розробки.

Підсумовуючи усе вищесказане, можна зробити висновок, що задачі магістерської кваліфікаційної роботи було виконано у повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фейламазова С. А. Информационные технологии в медицине: Учебное пособие для медицинских колледжей. Махачкала: ДБМК, 2016.

2. IT технології в медицині. URL: http://naukam.triada.in.ua/index.php/konferentsiji/42-dvanadtsyata-vseukr-ajinska-praktichno-piznavalna-internet-konferentsiya/462-it-tekhnologiji-v-meditsini (дата звернення: 5.09.2019).

3. Computer-aided diagnosis. URL: https://en.wikipedia.org/wiki/Computer-aided_diagnosis (Last accessed: 5.09.2019).

4. Romanyuk S.O., Romanyuk O.N., Pavlov S.V., Pyvovar M.A., Usage of 3D images for genetic diseases diagnosis, на *Міжнарод. наук.-практ. конференції «Інформаційні технології і автоматизація»*, м. Одеса, с. 7-9, 2019.

5. Романюк О. Н., Пивовар М. А., Перун І. В., Чехместрук Р. Ю., Аналіз алгоритмів пошуку осі дзеркальної симетрії обличчя людини, на *Всеукр. наук.-техн. конференції «Комп'ютерні технології: інновації, проблеми, рішення»*, м. Житомир, 2019.

6. Романюк О. Н., Пивовар М. А., Модифікація методу Лоя для визначення осі симетрії обличчя людини, *Збірник матеріалів Міжнародної науково-практичної конференції «Електронні інформаційні ресурси в освіті і науці: створення, використання, доступ»*, Вінниця, 2019.

7. Medical diagnosis – Wikipedia. URL: https://en.wikipedia.org/wiki/Medical_diagnosis (Last accessed: 6.09.2019).

8. Diagnostically relevant facial gestalt information from ordinary photos | eLife. URL: https://elifesciences.org/articles/02020 (Last accessed: 7.09.2019).

9. ФГБНУ НЦПЗ. «Патология психического развития». URL: http://www.psychiatry.ru/lib/54/book-/36/chapter/12/ (дата звернення: 8.09.2019).

10. Phenotypic characteristics of patients with CHD9 Mutations. URL: https://www.researchgate.net/figure/Phenotypic-characteristics-of-patients-with-CHD8-Mutations-A-Common-facialfeatures_fig1_28740-4414 (Last accessed: 8.09.2019).

11. Facts about Down Syndrome | CDC. URL: https://www.cdc.gov/ncbddd/birthdefects/down-syndrome.html (Last accessed: 9.09.2019).

12. Van Der Burgt I., Brunner, H. Genetic heterogeneity in Noonan syndrome: Evidence for an autosomal recessive form. *American Journal of Medical Genetics*. 2000. pp. 46–51.

13. Alcohol effects a fetus. URL: https://theworldnews.net/za-news/alcohol-effects-a-fetus (Last accessed: 10.09.2019).

14. PACS1 syndrome – Genetics Home Reference. URL: https://ghr.nlm.nih.gov/condition/pacs1-syndrome (Last accessed: 10.09.2019).

15. Аномалии развития головного мозга: причины заболевания, основные симптомы, лечение и профилактика. URL: https://www.obozrevatel.com/health/bolezni/anomalii-razvitiya-golovnogo-mozga.htm (дата звернення: 10.09.2019).

16. Постолаки А. И. Симметрия и асимметрия в гармонии лица и зубных рядов. *Медицинские науки*. 2000.

17. Дубовик Е. И. Асимметрия лицевого черепа при различных его формах у взрослого человека. СПб., 2010. С. 19.

18. Большая пятёрка – Википедия. URL: https://ru.wikipedia.org/wiki/Большая-пятёрка-(психология) (дата звернення: 12.09.2019).

19. Fink B., Neave N. et al., Facial symmetry and judgements of attractiveness, health and personality. *Personality and Individual Differences 41*. 2006.

20. Представления об индивидуально-психологических особенностях человека по структурным особенностям его лица. URL: http://psyjournals.ru/exp-/2009/n4/26600_full.shtml (дата звернення: 13.09.2019).

21. Епифанцев Б. Н., Архипов А. А. Об информативности признака асимметрии лица в задачах распознавания операторов эргатических систем. *Автометрия*. 2015.

22. Liu Y., Schmidtetal K. L., Facial Asymmetry Quantification for Expression Invariant Human Identification. *Computer Vision and Image Understanding*. 2003.

23. Журавская А. В. Геометрический поиск симметричных объектов на цифровом изображении. МГУ имени М. В. Ломоносова. Москва. 2018.

24. Loy G., Eklundh JO. Detecting Symmetry and Symmetric Constellations of Features. *Lecture Notes in Computer Science*. Berlin. 2006.

25. David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Comp. Vis*. 2004.

26. Computer aided diagnosis. URL: http://capstone.utu.fi/en-computer-aided-diagnosis (Last accessed: 15.09.2019).

27. Graphical User Interface Based Computer Aided Diagnosis Tool of Human Brain Tumor Segmentation Through MRI and Validation. URL: https://www.semanticscholar.org/paper/Graphical-User-Interface-Based-Computer-Aided-Tool-Karnam/e1dd98134ced557643d71ceb6055-390234d3366b (Last accessed: 16.09.2019).

28. Identifying facial phenotypes of genetic disorders using deep learning. URL: https://www.nature.com-/articles/s41591-018-0279-0 (Last accessed: 17.09.2019).

29. Scale-invariant feature transform – Wikipedia. URL: https://en.wikip-edia.org/wiki/Scale-invariant_fea-ture_transform (Last accessed: 18.09.2019).

30. Bay H. et al., Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, vol. 110, no. 3 , 2008. pp. 346-359.

31. Gaussian blur – Wikipedia. URL: https://en.wikipedia.org/wiki/Ga-ussian_blur (Last accessed: 20.09.2019).

32. Масштабно-инвариантная трансформация признаков. URL: http-s://ru.wikipedia.org/wiki/Масштабно_инвариантная_трансформация_признаков (дата звернення: 24.09.2019).

33. Alcantarilla P. F. et al. KAZE features. *European Conference on Computer Vision*, Berlin, ECCV. 2012. pp. 214-227.

34. Гессиан функции – Википедия. URL: https://ru.wikipedia.org/wi-ki/Гессиан_функции (дата звернення: 30.09.2019).

35. Alcantarilla P. F. et al. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *British Machine Vision Conference*, Bristol, BMVC, 2013.

36. Rublee E. et al. ORB: An efficient alternative to SIFT or SURF. *IEEE International Conference on Computer Vision*, Barcelona, ICCV. 2011. pp. 2564-2571.

37. Детекторы и дескрипторы особых точек FAST, BRIEF, ORB. URL: https://habr.com/ru/post/414459 (дата звернення: 1.10.2019).

38. Forlenza L., Carton P., Accardo D., Fasano G., Moccia A.. Real Time Corner Detection for Miniaturized Electro-Optical Sensors Onboard Small Unmanned Aerial Systems. *Sensors (Basel)*. 2012. pp. 863-877.

39. Leutenegger S. et al. BRISK: Binary robust invariant scalable keypoints. *IEEE International Conference on Computer Vision*, Barcelona, ICCV, 2011, pp. 2548-2555.

40. Kolkur S., Kalbande D., Shimpi P. et al Human Skin Detection Using RGB, HSV and YCbCr Color model. *Advances in Intelligent Systems Research*, ICCASP. 2017. pp. 324-332.

41. Albiol, A., Torres, L., Delp E. Optimum color spaces for skin detection. *Proceedings of the International Conference on Image Processing (ICIP)*. 2001. pp.122-124.

42. HSV – Wikipedia. URL: https://en.wikipedia.org/wiki/HSV (Last accessed: 14.10.2019).

43. YCbCr – Википедия. URL: https://ru.wikipedia.org/wiki/YCbCr (дата звернення: 14.10.2019).

44. Романюк О. Н., Чорний А. В., Високопродуктивні методи та засоби зафарбовування тривимірних графічних об’єктів.. Вінниця, Україна: УНІВЕСУМ-Вінниця, 2006.

45. Ferry Q., Steinberg J. et al. Diagnostically relevant facial gestalt information from ordinary photos. *eLife Sciences Publications*. 2014.

46. Active appearance model – Wikipedia. URL: https://en.wikipedia.org/wiki/Active_appearance_model (Last accessed: 19.10.2019).

47. Таблица размеров головы ребенка по возрасту у мальчиков и девочек. URL: https://littleone.com/publicatio-n/0-5989-tablica-okruzhnosti-golovy-rebenka-po-mesyacam (дата звернення: 21.10.2019).

48. ВОЗ | Окружность головы-возраст. URL: https://www.who.int/childgrowth/standards/hc_for_age/ru (дата звернення: 21.10.2019).

49. Коридоры развития ребенка. URL: https://periscopes.ru/koridory-razvitiya-rebenka-tablica-devochki-razvitie-reb-nka.html (дата звернення: 22.10.2019).

50. Sagonas C., Zafeiriou S. A Semi-automatic Methodology for Facial Landmark Annotation. *Computer Vision and Pattern Recognition Workshops* (CVPRW). 2013. p. 896-903.

51. C++ – Энциклопедия языков программирования. URL: http://progopedia.ru/language/c-plus-plus (дата звернення: 30.10.2019).

52. Visual Basic .NET. URL: https://ru.wikipedia.org/wiki/Visual_Basic_.NET (дата звернення: 30.10.2019).

53. Введение в язык C# и .NET Framework. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework (дата звернення: 30.10.2019).

54. Windows Forms. URL: https://ru.wikipedia.org/wiki/Windows_Forms (дата звернення: 1.11.2019).

55. Что такое WPF? URL: https://docs.microsoft.com/ru-ru/visualstudio/designers/getting-started-with-wpf (дата звернення: 1.11.2019).

56. What’s a Universal Windows Platform (UWP) app? URL: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide (Last accessed: 1.11.2019).

57. Emgu CV: OpenCV in .NET. URL: http://www.emgu.com/wiki/index.php/Main_Page (Last accessed: 2.11.2019).

58. Entity Framework Overview | Microsoft Docs. URL: https://docs.microsoft.com/en-us/dotnet/framewor-k/data/adonet/ef/overview (Last accessed: 2.11.2019).

59. Microsoft Visual Studio. URL: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio (Last accessed: 3.11.2019).

60. MonoDevelop – Wikipedia. URL: https://ru.wikipedia.org/wiki/MonoDevelop (дата звернення: 2.11.2019).

61. Rider – Wikipedia. URL: https://ru.wikipedia.org/wiki/Rider (дата звернення: 2.11.2019).

62. Дейт К. Дж. Введение в системы баз данных - Introduction to Database Systems. 8-е изд. М.: Вильямс. 2005. С. 1328.

63. Романюк О. Н. Організація баз даних і знань. / О.Н. Романюк // Навчальний посібник. – Вінниця: УНІВЕРСУМ – Вінниця. 2003. С. 123.

64. Movel-View-ViewModel – Wikipedia. URL: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%-80%93viewmodel (Last accessed: 5.11.2019).

65. Хроматичне коло Іттена. URL: https://studopedia.su/5_27204_hromatichne-kolo-Ittena.html (дата звернення: 5.11.2019).

66. Software testing – Wikipedia. URL: https://en.wikipedia.org/wiki/Software_testing (Last accessed: 6.11.2019).

67. Бейзер Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер. 2004. С. 320.

68. Тестування програмного забезпечення. URL: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення (дата звернення: 7.11.2019.

69. Методичні рекомендації з комерціалізації розробок, створених в результаті науково-технічної діяльності – К.: Наказ Державного комітету України з питань науки, інновацій та інформатики (Лист № 1/06-4-97 від 13.09.2010 р.).

70. Козловський В. О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт. Вінниця: ВНТУ. 2012.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ д.т.н., проф. О.Н. Романюк

"____"_____2019 р.

Виконав:

_____студент гр.1ПІ-18м М.А. Пивовар

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування».

Галузь застосування – комп'ютерна медична діагностика.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності медичного експрес-діагностування за рахунок розробки нових та модифікації існуючих методів та засобів.

Призначення роботи – розробка методів і засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Romanyuk S.O., Romanyuk O.N., Pavlov S.V., Pyvovar M.A. Usage of 3D images for genetic diseases diagnosis, на *Міжнарод. наук.-практ. конференції «Інформаційні технології і автоматизація*», м. Одеса, с. 7-9, 2019.
2. Романюк О. Н., Пивовар М. А., Перун І. В., Чехместрук Р. Ю., Аналіз алгоритмів пошуку осі дзеркальної симетрії обличчя людини, на *Всеукр. наук.-техн. конференції «Комп'ютерні технології: інновації, проблеми, рішення»*, м. Житомир, 2019.
3. Романюк О. Н., Пивовар М. А., «Модифікація методу Лоя для визначення осі симетрії обличчя людини», *Збірник матеріалів Міжнародної науково-*

практичної конференції «Електронні інформаційні ресурси в освіті і науці: створення, використання, доступ», Вінниця, 2019.

4. Loy G., Eklundh JO. Detecting Symmetry and Symmetric Constellations of Features. *Lecture Notes in Computer Science*, vol. 3952, Berlin, 2006.

5. Ferry Q., Steinberg J. et al. Diagnostically relevant facial gestalt information from ordinary photos. *eLife Sciences Publications*. 2014.

5. Технічні вимоги

Кольоровий режим – TrueColor; метрологічні параметри голови дитини; розмір екрану – 1280x1024; вихідні методи для модифікації – метод Лоя для визначення осі симетрії обличчя людини; архів зображень облич дітей з генетичними захворюваннями; таблиці нормованих значень обхвату голови дитини залежно від віку.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз предметної галузі та постановка задачі розробки	04.09.2019 – 29.09.2019
2	Розробка модифікованого методу визначення симетричності обличчя людини	30.09.2019 – 26.10.2019
3	Розробка методів для проведення медичного експрес-діагностування	27.10.2019 – 3.11.2019
4	Розробка та тестування програмного додатку для медичного експрес-діагностування	4.11.2019 – 12.11.2019
5	Економічна частина	13.11.2019 – 17.11.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. Лістинг коду

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;

namespace Pyvovar_MKR
{
   public partial class App : Application
   {
      protected override void OnStartup(StartupEventArgs e)
      {
         base.OnStartup(e);

         ApplicationView app = new ApplicationView();
         ApplicationViewModel context = new ApplicationViewModel();
         app.DataContext = context;
         app.Show();
      }
   }
}
   public class ApplicationViewModel : ObservableObject
   {
      private ICommand _changePageCommand;
      private IPageViewModel _currentPageViewModel;
      private List<IPageViewModel> _pageViewModels;


      public ApplicationViewModel()
      {
         // створення усіх існуючих viewModel
         PageViewModels.Add(new MainViewModel());
         PageViewModels.Add(new CreatePatientViewModel());
         PageViewModels.Add(new SelectPatientViewModel());
         PageViewModels.Add(new DistancesNormsViewModel());
```

```
    PageViewModels.Add(new SymmetryNormsViewModel());

    // задання початкової сторінки
    CurrentPageViewModel = PageViewModels[0];
}


public ICommand ChangePageCommand
{
    get
    {
        if (_changePageCommand == null)
        {
            _changePageCommand = new RelayCommand(
                p => ChangeViewModel((IPageViewModel)p),
                p => p is IPageViewModel);
        }

        return _changePageCommand;
    }
}

public List<IPageViewModel> PageViewModels
{
    get
    {
        if (_pageViewModels == null)
            _pageViewModels = new List<IPageViewModel>();

        return _pageViewModels;
    }
}

public IPageViewModel CurrentPageViewModel
{
    get
    {
        return _currentPageViewModel;
    }
```

```
        set
        {
            if (_currentPageViewModel != value)
            {
                _currentPageViewModel = value;
                OnPropertyChanged("CurrentPageViewModel");
            }
        }
    }


    private void ChangeViewModel(IPageViewModel viewModel)
    {
        if (!PageViewModels.Contains(viewModel))
            PageViewModels.Add(viewModel);

        CurrentPageViewModel = PageViewModels
            .FirstOrDefault(vm => vm == viewModel);
    }
}
public class Patient
{
    public int PatientID { get; set; }
    public string PatientName { get; set; }
    public string PatientPhone { get; set; }
    public string PatientSex { get; set; }
    public string PatientEmail { get; set; }
    public DateTime PatientDateOfBirth { get; set; }
    public double PatientCircumferance { get; set; }
    public double PatientSymmetryLevel { get; set; }
    public string PatientDiagnosis { get; set; }
    public ICollection<Image> Image { get; set; }
    public ICollection<PatientDistance> PatientDistance { get; set; }
}
public class Image
{
    public int ImageID { get; set; }
    public string ImagePath { get; set; }
    public DateTime ImageDate { get; set; }
```

```
        public int ImageType { get; set; }
        public Patient Patient { get; set; }
    }
    public class Distance
    {
        public int DistanceID { get; set; }
        public string DistanceName { get; set; }
        public double DistanceValue { get; set; }
        public double DistanceMin { get; set; }
        public double DistanceMax { get; set; }
        public ICollection<PatientDistance> PatientDistance { get; set; }
    }
    public class PatientDistance
    {
        public Patient Patient { get; set; }
        public Distance Distance { get; set; }
    }
public static Image<Bgr, byte> DetectFacialLandmarks(Image<Bgr, byte> img)
{
    String facemarkFileName = "trainedModel.yaml";
    LoadModel(facemarkFileName);

    using (FacemarkLBFParams facemarkParam = new
CV.Face.FacemarkLBFParams())

    using (FacemarkLBF facemark = new CV.Face.FacemarkLBF(facemarkParam))
    using (VectorOfRect vr = new VectorOfRect(faceRegions.ToArray()))

    using (VectorOfVectorOfPointF landmarks = new VectorOfVectorOfPointF())
    {
        facemark.LoadModel(facemarkFileName);
        facemark.Fit(img, vr, landmarks);

        foreach (Rectangle face in faceRegions)
        {
            CvInvoke.Rectangle(img, face, new MCvScalar(0, 255, 0));
        }

        int len = landmarks.Size;
```

```
        for (int i = 0; i < landmarks.Size; i++)
        {
            using (VectorOfPointF vpf = landmarks[i])
                FaceInvoke.DrawFacemarks(img, vpf, new MCvScalar(255, 0, 0));
        }

    }
    return img;
}
public static double CalculateSymmetry(Image<Bgr, byte> image)
{
    double symmetryLevel = 0;
    ORBDetector orb = new ORBDetector();

    Image<Bgr, byte> mimage = new Image<Bgr, byte>(image.Width, image.Height);
    CvInvoke.Flip(image, mimage, 0);

    orb.DetectAndCompute(image, null);

    return symmetryLevel;
}
public static Image<Bgr, byte> RemoveBackground(Image<Bgr, byte> source)
{
    Image<Gray, byte> mask = new Image<Gray, byte>(source.Width + 2,
source.Height + 2);
    MCvConnectedComp comp = new MCvConnectedComp();

    //центр зображення в якості початкової точки
    Point point1 = new Point(source.Width / 2, source.Height / 2);

    CvInvoke.cvFloodFill(source, point1, new MCvScalar(255), new MCvScalar(50,
30, 10),
                                        new MCvScalar(50, 30, 10), out comp, 8 | (1 << 17) |
(255 << 8), mask);

    Image<Gray, byte> reducedMask = mask.Copy(new Rectangle(1, 1, source.Width,
source.Height));
    Image<Gray, byte> morphImg = new Image<Gray, byte>(source.Size);
```

```
   StructuringElementEx element = new StructuringElementEx(3, 3, 1, 1,
Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_CROSS);

   CvInvoke.cvMorphologyEx(reducedMask, morphImg, IntPtr.Zero, element,
CV_MORPH_OP.CV_MOP_CLOSE, 4);
   Image<Bgr, Byte> imageROI = source.Copy(morphImg);

   return imageROI;
}
public static void FindMatch(Mat modelImage, Mat observedImage, out
VectorOfKeyPoint modelKeyPoints, out VectorOfKeyPoint observedKeyPoints,
VectorOfVectorOfDMatch matches, out Mat mask, out Mat homography)
{
   int k = 2;
   double uniquenessThreshold = 0.80;
   homography = null;

   modelKeyPoints = new VectorOfKeyPoint();
   observedKeyPoints = new VectorOfKeyPoint();

   using (UMat uModelImage = modelImage.GetUMat(AccessType.Read))
   using (UMat uObservedImage = observedImage.GetUMat(AccessType.Read))
   {
      var featureDetector = new ORBDetector(9000);
      Mat modelDescriptors = new Mat();

      featureDetector.DetectAndCompute(uModelImage, null, modelKeyPoints,
modelDescriptors, false);
      Mat observedDescriptors = new Mat();

      featureDetector.DetectAndCompute(uObservedImage, null, observedKeyPoints,
observedDescriptors, false);
      using (var matcher = new BFMatcher(DistanceType.Hamming, false))
      {
         matcher.Add(modelDescriptors);

         matcher.KnnMatch(observedDescriptors, matches, k, null);
         mask = new Mat(matches.Size, 1, DepthType.Cv8U, 1);
         mask.SetTo(new MCvScalar(255));
```

```
            Features2DToolbox.VoteForUniqueness(matches, uniquenessThreshold,
mask);

            int nonZeroCount = CvInvoke.CountNonZero(mask);
            if (nonZeroCount >= 4)
            {
               nonZeroCount =
Features2DToolbox.VoteForSizeAndOrientation(modelKeyPoints,
observedKeyPoints,
                  matches, mask, 1.5, 20);
               if (nonZeroCount >= 4)
                  homography =
Features2DToolbox.GetHomographyMatrixFromMatchedFeatures(modelKeyPoints,
                     observedKeyPoints, matches, mask, 2);
            }
         }
      }
   }
}
public static Mat DetectSkin(string imagePath)
{
   Mat result;
   Mat img = CvInvoke.Imread(imagePath,
Emgu.CV.CvEnum.ImreadModes.AnyColor);

   Mat img_HSV, HSV_mask, yCrCb_mask;
   CvInvoke.CvtColor(img, img_HSV,
Emgu.CV.CvEnum.ColorConversion.Rgb2Hsv);

   CvInvoke.InRange(img_HSV, new ScalarArray(new MCvScalar(0, 15, 0)), new
ScalarArray(new MCvScalar(17, 170, 255)), HSV_mask);
   CvInvoke.MorphologyEx(HSV_mask, HSV_mask,
Emgu.CV.CvEnum.MorphOp.Open, 1);

   Mat img_YcrCb = CvInvoke.CvtColor(img,
Emgu.CV.CvEnum.ColorConversion.YCrCb2Rgb);

   CvInvoke.InRange(img_YcrCb, new ScalarArray(new MCvScalar(0, 15, 0)), new
ScalarArray(new MCvScalar(17, 170, 255)), yCrCb_mask);
```

```
    CvInvoke.MorphologyEx(yCrCb_mask, yCrCb_mask,
Emgu.CV.CvEnum.MorphOp.Open, 1);

    Mat globalMask = CvInvoke.BitwiseAnd(yCrCb_mask, HSV_mask);
    globalMask = CvInvoke.MedianBlur(globalMask, 3);
    globalMask = CvInvoke.MorphologyEx(globalMask, globalMask,
Emgu.CV.CvEnum.MorphOp.Open, 1);

    result = CvInvoke.BitwiseNot(globalMask);
    return result;
}
public static double AngleWithXaxis(Point p1, Point p2)
{
    int x = p1.X - p2.X;
    int y = p1.Y - p2.Y;

    if(x == 0)
    {
        return Math.PI / 2;
    }
    double angle = Math.Atan(y / x);
    if(angle < 0)
    {
        angle += Math.PI;
    }
    return angle;
}
public static Point DetectMidpoint(Point p1, Point p2)
{
    Point resultPoint = new Point();
    resultPoint.X = (p1.X + p2.X) / 2;
    resultPoint.Y = (p1.Y + p2.Y) / 2;

    return resultPoint;
}
public static bool VeryClose(Point a, Point b, double tol = 4.0)
{
    return Math.Sqrt(Math.Pow(a.X - b.X, 2) + Math.Pow(a.Y - b.Y, 2)) < tol;
}
```

```
public static double Reisfeld(double phi, double phj, double theta)
{
   return 1 - Math.Cos(phi + phj - 2 * theta);
}

public static double SFunction(double si, double sj, int sigma = 1)
{
   double q = ((0 - Math.Abs(si - sj)) / (sigma * (si + sj)));
   return Math.Exp(Math.Pow(q, 2));
}

public static List<Point> DetectSymmetryAxis(Image<Bgr, byte> image)
{
   Mat mimage = CvInvoke.Flip(image, 2);
   ORBDetector detector = new ORBDetector();

   IOutputArray des1, des2;
   double theta, r, mIJ;

   VectorOfKeyPoint kp1 = detector.DetectAndCompute(image, des1, false);
   VectorOfKeyPoint kp2 = detector.DetectAndCompute(mimage, des2, false);
   ConvertDegreesToRadians(kp1);
   ConvertDegreesToRadians(kp2);
   BFMatcher matcher = new BFMatcher(DistanceType.Hamming2);
   VectorOfVectorOfDMatch matches = matcher.KnnMatch(des1, des2, 2);

   Array houghr, houghth, weights = Array.Clear(matches, 0, matches.Length);
   List<VectorOfDMatch> good = new List<VectorOfDMatch>();
   int i = 0;
   foreach (VectorOfDMatch match in matches)
   {
      MKeyPoint point = kp1[match.queryIdx];
      MKeyPoint mirpoint = kp1[match.trainIdx];
      MKeyPoint mirpoint2 = kp2[match.trainIdx];

      mirpoint2.Angle = Math.PI - mirpoint2.Angle;
      mirpoint.Angle = Math.PI - mirpoint.Angle;
```

```
        if (mirpoint.Angle < 0.0) mirpoint.Angle += 2 * Math.PI;
        if (mirpoint2.Angle < 0.0) mirpoint2.Angle += 2 * Math.PI;

        mirpoint.Point = (mimage.Shape[1] - mirpoint.Point[0], mirpoint.Point[1]);
        if (VeryClose(point.Point, mirpoint.Point))
        {
            mirpoint = mirpoint2;
            result.Add(match2);
        }
        else { good.Add(match); }
        theta = AngleWithXaxis(point.Point, mirpoint.Point);
        Point c = DetectMidpoint(point.Point, mirpoint.Point);

        r = c.X * Math.Cos(theta + c.Y) * Math.Sin(theta);
        mIJ = Reisfeld(point.Angle, mirpoint.Angle, theta) * SFunction(point.Size,
mirpoint.Size);
        houghr[i] = r;
        houghth[i] = theta;
        weights[i] = mIJ;
        i += 1;
        good = good.OrderBy(o => o.Distance).ToList();
    }
    List<Point> result = new List<Point>();
    foreach (Point point in good)
    {
        if ((Math.PI / 4 < theta) && (theta < 3 * Math.PI / 4))
        {
            Point rPoint = new Point();
            rPoint.X = Convert.ToInt32(r - point.Y * Math.Sin(theta) / Math.Cos(theta));
            rPoint.Y = Convert.ToInt32(r - point.X * Math.Cos(theta) / Math.Sin(theta));
            result.Add(rPoint);
        }
    }
    return result;
}
public static double GetSymmetryLevel(List<Pixel> area1, List<Pixel> area2)
{
    double NMSE = 0;
    var values = area1.Zip(area2, (a1, a2) => new { A1 = a1, A2 = a2 });
```

```
    int rx = 0, gx = 0, bx = 0, a1Total = 0;
    foreach (var value in values)
    {
        rx += Math.Pow(value.A1.R - value.A2.R, 2);
        gx += Math.Pow(value.A1.G - value.A2.G, 2);
        bx += Math.Pow(value.A1.B - value.A2.B, 2);
        a1Total = Math.Pow(value.A1.R, 2) + Math.Pow(value.A1.G, 2) +
Math.Pow(value.A1.B, 2);
    }
    NMSE = (rx + gx + bx) / a1Total;

    return NMSE;
}
public static List<Pixel> GetPixelsArray(string imagePath)
{
    Bitmap img = new Bitmap(imagePath);
    List<Pixel> pixels = new List<Pixel>();
    for (int i = 0; i < img.Width; i++)
    {
        for (int j = 0; j < img.Height; j++)
        {
            Color color = img.GetPixel(i, j);
            Pixel pixel = new Pixel();
            pixel.R = color.R;
            pixel.G = color.G;
            pixel.B = color.B;
            pixels.Add(pixel);
        }
    }
    return pixels;
}
public static int AnalyzeCircumference(double value, double age, int sex)
{
    double x3 = Math.Pow(age, 3);
    double x2 = Math.Pow(age, 2);
    double x = age;
    // 1 - хлопець, 2 - дівчина
    if(sex == 1)
```

```
    {
        double SD3 = 0.0082 * x3 - 0.237 * x2 + 2.8406 * x + 35.897;
        double nSD3 = 0.0081 * x3 - 0.2457 * x2 + 2.9261 * x + 28.518;
        double SD2 = 0.0082 * x3 - 0.2395 * x2 + 2.8611 * x + 34.651;
        double nSD2 = 0.008 * x3 - 0.2428 * x2 + 2.9114 * x + 29.696;
        double SD1 = 0.0082 * x3 - 0.2413 * x2 + 2.8912 * x + 33.359;
        double nSD1 = 0.0081 * x3 - 0.2415 * x2 + 2.8916 * x + 30.971;

        //1 - повністю здоровий, 2 - незначне відхилення, 3 - серйозне відхилення
        if (value >= nSD1 && value <= SD1) return 1;
        if (value > SD2 && value < SD3 && value < nSD2 && value > nSD3) return 2;
        if (value >= SD3 || value <= nSD3) return 3;
    }
    else
    {
        double SD3 = 0.0083 * x3 - 0.24 * x2 + 2.8443 * x + 35.113;
        double nSD3 = 0.0081 * x3 - 0.2308 * x2 + 2.6587 * x + 28.257;
        double SD2 = 0.0083 * x3 - 0.2382 * x2 + 2.8109 * x + 33.959;
        double nSD2 = 0.0081 * x3 - 0.2313 * x2 + 2.6817 * x + 29.428;
        double SD1 = 0.008 * x3 - 0.2293 * x2 + 2.723 * x + 32.935;
        double nSD1 = 0.008 * x3 - 0.2308 * x2 + 2.6944 * x + 30.599;

        if (value >= nSD1 && value <= SD1) return 1;
        if (value > SD2 && value < SD3 && value < nSD2 && value > nSD3) return 2;
        if (value >= SD3 || value <= nSD3) return 3;
    }
    return 0;
}

private Image<Bgr, byte> ToGrayscaleImage(Image<Bgr, byte> inputImage)
{
    UMat uimage = new UMat();
    CvInvoke.CvtColor(inputImage, uimage, ColorConversion.Bgr2Gray);
    var outputImage = uimage.ToImage<Bgr, Byte>();
    return outputImage;
}

private Image<Bgr, byte> ToCannyImage(Image<Bgr, byte> inputImage)
{
```

```
    var outputImage = inputImage.Canny(400, 20);
    return outputImage.ToUMat().ToImage<Bgr, byte>();
}


private Image<Bgr, byte> GetContours(Image<Bgr, byte> inputImage)
{
    double cannyThreshold = 180.0;
    double circleAccumulatorThreshold = 120;

    contourOutput = originalImage;
    UMat matInput = inputImage.ToUMat();
    matInput.ConvertTo(matInput, DepthType.Cv8U);

    CvInvoke.CvtColor(matInput, matInput, ColorConversion.Bgr2Gray);
    circles = CvInvoke.HoughCircles(matInput, HoughType.Gradient, 2.0, 20.0,
cannyThreshold, circleAccumulatorThreshold, 5);

    CvInvoke.FindContours(matInput, contours, null, RetrType.List,
ChainApproxMethod.ChainApproxSimple);
    CvInvoke.DrawContours(contourOutput, contours, -1, new MCvScalar(0, 255, 0),
1);
    return contourOutput;
}

public class Matcher
{
    public int Threshold = 4;

    public MatchResult FindMatch(string path)
    {
        var mat = new Mat(path, LoadImageType.Grayscale);
        return FindMatch(mat);
    }

    public MatchResult FindMatch(Mat queryMat)
    {
        try
        {
```

```
        var queryDescriptors = DescriptorManager.ExtractDescriptor(queryMat);
        const int Knn = 11;
        using (var indices = new Mat(new Size(queryDescriptors.Rows, 2),
DepthType.Cv32F, 2))
        using (var dists = new Mat(new Size(queryDescriptors.Rows, 2),
DepthType.Cv32F, 2))
        {

            IndexContext.CurrentFlannIndex.KnnSearch(queryDescriptors, indices,
dists, Knn, 32);
            var result = new Dictionary<Models.Image, QueryHit>();

            for (int i = 0; i < indices.Rows; i++)
            {
               if (dists.GetFloatValue(i, 0) < (0.6 * dists.GetFloatValue(i, 1)))
               {
                  var pos = indices.GetIntValue(i, 0);
                  if (IndexContext.CurrentMappingIndex.ContainsKey(pos))
                  {
                     var img = IndexContext.CurrentMappingIndex[pos];
                     if (!result.ContainsKey(img))
                     {
                        result.Add(img, new QueryHit { Image = img });
                     }
                     result[img].Hits++;
                  }
               }
            }
            return new MatchResult
            {
               Matches = result.Where(x => x.Value.Hits >
Threshold).OrderByDescending(x => x.Value.Hits).ToDictionary(x => x.Key, y =>
y.Value),
            };



        }
    }
```

```
        catch (Exception e)
        {

            return null;
        }
    }
}
public static void CheckAndDownloadFile(String fileName, String url)
{
    if (!File.Exists(fileName))
    {
        String fileUrl = url + fileName;
        Trace.WriteLine("downloading file from:" + fileUrl + " to: " + fileName);
        System.Net.WebClient downloadClient = new System.Net.WebClient();
        try
        {
            downloadClient.DownloadFile(fileUrl, fileName);
        }
        catch
        {
            File.Delete(fileName);
            throw;
        }
    }
}
public static int GenerateDatabase() {

    _FaceRecognition = FaceRecognition.Create(directory);

    const string extractPath = "studyImages";
    var zips = new[]
    {
        new{ Zip = "studyImages1.zip",    IsImage = true,  Directory = "studyImages1"
},
        new{ Zip = "studyImages2.zip",    IsImage = true,  Directory = "studyImages2"
},
        new{ Zip = "studyImages3.zip",    IsImage = true,  Directory = "studyImages3"
},
```

```
        new{ Zip = "studyImages4.zip",    IsImage = true,  Directory = "studyImages4"
}
    };

    Directory.CreateDirectory(extractPath);

    foreach (var zip in zips)
    {
        if (!Directory.Exists(Path.Combine(extractPath, zip.Directory)))
            ZipFile.ExtractToDirectory(zip.Zip, extractPath);
    }

    var annotation = zips.FirstOrDefault(arg => !arg.IsImage);
    var imageZips = zips.Where(arg => arg.IsImage).ToArray();
    if (annotation == null)
        return -1;

    var images = new List<Image>();
    foreach (var file in Directory.EnumerateFiles(Path.Combine(extractPath,
annotation.Directory)))
    {

        var txt = File.ReadAllLines(file);
        var filename = txt[0];
        var jpg = $"{filename}.jpg";
        foreach (var imageZip in imageZips)
        {
            var found = false;
            var path = Path.Combine(Path.Combine(extractPath, imageZip.Directory,
jpg));
            if (File.Exists(path))
            {
                found = true;
                using (var fi = FaceRecognition.LoadImageFile(path))
                {
                    var locations = _FaceRecognition.FaceLocations(fi, 1,
Model.Hog).ToArray();
                    else
                    {
```

```
                var location = locations.First();
                var parts = new List<Part>();
                for (var i = 1; i < txt.Length; i++)
                {
                    var tmp = txt[i].Split(',').Select(s =>
s.Trim()).Select(float.Parse).Select(s => (int)s).ToArray();
                    parts.Add(new Part { X = tmp[0], Y = tmp[1], Name = $"{i - 1}"
});
                }

                var image = new Image
                {
                    File = Path.Combine(imageZip.Directory, jpg),
                    Box = new Box
                    {
                        Left = location.Left - padding,
                        Top = location.Top - padding,
                        Width = location.Right - location.Left + 1 + padding * 2,
                        Height = location.Bottom - location.Top + 1 + padding * 2,
                        Part = parts.ToArray()
                    }
                };

                using (var bitmap = System.Drawing.Image.FromFile(path))
                {
                    var b = image.Box;
                    using (var g = Graphics.FromImage(bitmap))
                    {
                        using (var p = new Pen(Color.Red, bitmap.Width / 400f))
                            g.DrawRectangle(p, b.Left, b.Top, b.Width, b.Height);

                        foreach (var part in b.Part)
                            g.FillEllipse(Brushes.GreenYellow, part.X, part.Y, 5, 5);
                    }

                    var result = Path.Combine(extractPath, "Result");
                    Directory.CreateDirectory(result);

                    bitmap.Save(Path.Combine(result, jpg), ImageFormat.Jpeg);
```

```
                }

                images.Add(image);
            }
        }
    }

    if (found)
        break;
  }
}
var dataset = new Dataset
{
  Name = "Multi-PIE dataset",
  Images = images.ToArray()
};

var settings = new XmlWriterSettings();
using (var sw = new StreamWriter(Path.Combine(extractPath,
"trainedModel.yaml"), false, new System.Text.UTF8Encoding(false)))
using (var writer = XmlWriter.Create(sw, settings))
{
  writer.WriteProcessingInstruction("xml-stylesheet", @"type=""text/xsl""
href=""image_metadata_stylesheet.xsl""");
  var serializer = new XmlSerializer(typeof(Dataset));
  serializer.Serialize(writer, dataset);
}
return 0;
}
public class DescriptorManager
{

  private static ORBDetector _detector;
  public static ORBDetector Detector
      => _detector ?? (_detector = new ORBDetector(numberOfFeatures: 700,
scaleFactor: 1.2F, nLevels: 10, edgeThreshold: 0));

  public static Mat ExtractDescriptor(Models.Image image)
  {
```

```
    if (!string.IsNullOrWhiteSpace(image.LocalPath))
    {
        using (var mat = new Mat(image.LocalPath, LoadImageType.Grayscale))
        {
            //mat.ResizeKeepAspect(640, 480);
            return ExtractDescriptor(mat);
        }
    }
    return null;
}

public static Mat ExtractDescriptor(Mat mat)
{
    var descriptors = new Mat();
    var keyPoints = new VectorOfKeyPoint();
    Detector.DetectAndCompute(mat, null, keyPoints, descriptors, false);
    descriptors.ConvertTo(descriptors, DepthType.Cv8U);
    return descriptors;
}
public static Mat ConcatDescriptors(IEnumerable<Mat> descriptors)
{
    var mat = descriptors.FirstOrDefault();
    if (mat == null) return null;
    descriptors = descriptors.Skip(1);
    foreach (var descriptor in descriptors)
    {
        var next = new Mat();
        CvInvoke.VConcat(mat, descriptor, next);
        next.ConvertTo(mat, DepthType.Cv8U);
    }
    mat.ConvertTo(mat, DepthType.Cv8U);
    return mat;
}
```

Додаток В. Ілюстративний матеріал

Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, д. т. н., професор __ _____ О. Н. Романюк

Рецензент, к. т. н., доцент кафедри КН _____ _____ В. В. Колодний

Нормоконтроль, д. т. н., професор _____ О. Н. Романюк

Виконавець, студент гр.1ПІ-18м _____ М. А. Пивовар

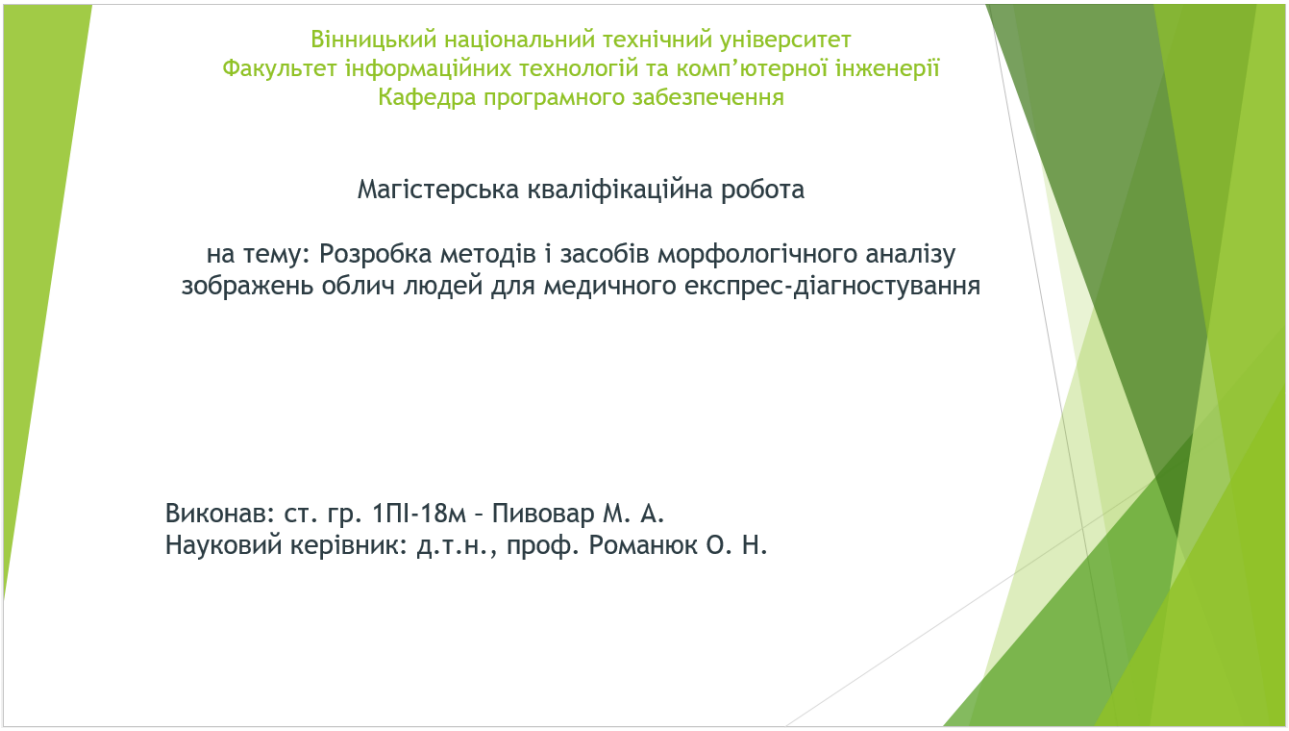


Рисунок В.1 – Слайд 1. Тема, автор та керівник магістерської кваліфікаційної роботи

Рисунок В.2 – Слайд 2. Галузі застосування зображень облич людей в медицині

Мета, об’єкт, предмет та завдання роботи

Метою роботи є підвищення ефективності медичного експрес-діагностування за рахунок розробки нових та модифікації існуючих методів та засобів морфологічного аналізу зображень облич людей.

Об’єктом дослідження є процес морфологічного аналізу зображень облич людей в медичному експрес-діагностуванні

Предметом дослідження є методи та засоби аналізу 2D-зображень

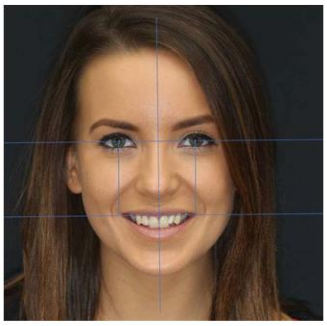
Завдання, що розв’язувались у роботі:

- провести аналіз існуючих методів та засобів морфологічного аналізу зображень облич людей для медичного експрес-діагностування з метою пошуку способів підвищення їх ефективності;
- запропонувати нові:
  - методи підвищення продуктивності визначення осі симетрії обличчя людини;
  - методи підвищення швидкості визначення типу генетичних захворювань на основі аналізу візуальних характеристик обличчя людини;
- отримати аналітичні залежності обхвату голови дитини від її віку з урахуванням допустимого інтервалу;
- розробити програмний додаток на основі запропонованих методів;
- провести тестування розробленого програмного додатку.

Рисунок В.3 – Слайд 3. Мета, об’єкт, предмет та завдання роботи

Симетрія обличчя людини в діагностиці

Асиметрія обличчя є важливим фактором індивідуальної краси та може слугувати симптомом багатьох хвороб, пов’язаних з аномаліями лицевого нерву та кісток черепа, порушенням мімічних м’язів, змінами м’яких тканин та інших.



Дослідження, проведене Б. Фінком показало, що людям з низьким ступенем асиметрії обличчя, оточуючі частіше присвоюють позитивні якості, ніж людям з сильно несиметричним обличчям.

Рисунок В.4 – Слайд 4. Симетрія обличчя людини в діагностиці

Модифікація методу Лоя для визначення осі симетрії обличчя людини

Одним із найпоширеніших методів визначення осі симетрії обличчя людини є метод Лоя, який базується на алгоритмі визначення ключових точок SIFT.

Запропоновано модифікацію методу Лоя за рахунок заміни алгоритму визначення ключових точок SIFT на ORB, що дозволить значно підвищити швидкість та точність роботи методу визначення осі симетрії.

Середні значення результатів роботи алгоритмів визначення ключових точок

Алгоритм	Кількість знайдених точок	Тривалість роботи, с
SIFT	3424.9	0.2665
SURF	4143.1	0.1847
KAZE	1586.5	0.2820
AKAZE	1743.2	0.0994
ORB	9754.3	0.0393
BRISK	6375.8	0.1695

Рисунок В.5 – Слайд 5. Модифікація методу Лоя для визначення осі симетрії обличчя людини

Метод визначення симетричності обличчя людини

Алгоритм Колкура для виявлення шкіри

$$0 \le H \le 17, 15 \le S \le 170, 0 \le V \le 255,$$
$$0 \le Y \le 255, 135 \le Cr \le 180, 85 \le Cb \le 135.$$

$$Y = 0.299R + 0.287G + 0.11B.$$
$$Cr = R - Y.$$
$$Cb = B - Y.$$

Середньоквадратична похибка для порівняння

$$NMSE = \frac{\sum_i (R_1(i) - R_2(i))^2 + (G_1(i) - G_2(i))^2 + (B_1(i) - B_2(i))^2}{\sum_i R_1(i)^2 + G_1(i)^2 + B_1(i)^2}$$

Рисунок В.6 – Слайд 6. Метод визначення симетричності обличчя людини

Рисунок В.7 – Слайд 7. Визначення рівня асиметричності

Рисунок В.8 – Слайд 8. Діагностування генетичних захворювань на основі аналізу морфологічних характеристик обличчя людини

Рисунок В.9 – Слайд 9. Ознаки генетичних захворювань відповідно до запропонованої моделі

Обхват голови дівчат від народження до п'яти років

	SD -3	SD -2	SD -1	SD 1	SD 2	SD 3
Новонароджена	30,3	31,5	32,7	35,1	36,2	37,4
1 місяць	33,0	34,2	35,4	37,7	38,9	40,1
2 місяці	34,6	35,8	37,0	39,5	40,7	41,9
3 місяці	35,8	37,1	38,3	40,8	42,0	43,3
4 місяці	36,8	38,1	39,3	41,8	43,1	44,4
5 місяців	37,6	38,9	40,2	42,7	44,0	45,3
6 місяців	38,3	39,6	40,9	43,5	44,8	46,1
7 місяців	38,9	40,2	41,5	44,1	45,5	46,8
8 місяців	39,4	40,7	42,0	44,7	46,0	47,4
9 місяців	39,8	41,2	42,5	45,2	46,5	47,8
10 місяців	40,2	41,5	42,9	45,6	46,9	48,3
11 місяців	40,5	41,9	43,2	45,9	47,3	48,6
12 місяців	40,8	42,2	43,5	46,3	47,6	49,0
24 місяці	43,0	44,4	45,8	48,6	50,0	51,4
36 місяців	44,3	45,7	47,1	49,9	51,3	52,7
48 місяців	45,1	46,5	47,9	50,8	52,2	53,6
60 місяців	45,7	47,1	48,5	51,3	52,8	54,2

Обхват голови хлопців від народження до п'яти років

	SD -3	SD -2	SD -1	SD 1	SD 2	SD 3
Новонароджений	30,7	31,9	33,2	35,7	37,0	38,3
1 місяць	33,8	34,9	36,1	38,4	39,6	40,8
2 місяці	35,6	36,8	38,0	40,3	41,5	42,6
3 місяці	37,0	38,1	39,3	41,7	42,9	44,1
4 місяці	38,0	39,2	40,4	42,8	44,0	45,2
5 місяців	38,9	40,1	41,4	43,8	45,0	46,2
6 місяців	39,7	40,9	42,1	44,6	45,8	47,0
7 місяців	40,3	41,5	42,7	45,2	46,4	47,7
8 місяців	40,8	42,0	43,3	45,8	47,0	48,3
9 місяців	41,2	42,5	43,7	46,3	47,5	48,8
10 місяців	41,6	42,9	44,1	46,7	47,9	49,2
11 місяців	41,9	43,2	44,5	47,0	48,3	49,6
12 місяців	42,2	43,5	44,8	47,4	48,6	49,9
24 місяці	44,2	45,5	46,9	49,6	51,0	52,3
36 місяців	45,2	46,6	48,0	50,9	52,3	53,7
48 місяців	45,8	47,3	48,7	51,7	53,1	54,6
60 місяців	46,3	47,7	49,2	52,2	53,7	55,2

Рисунок В.10 – Слайд 10. Статистичні дані про нормовані значення обхвату голови дітей залежно від віку

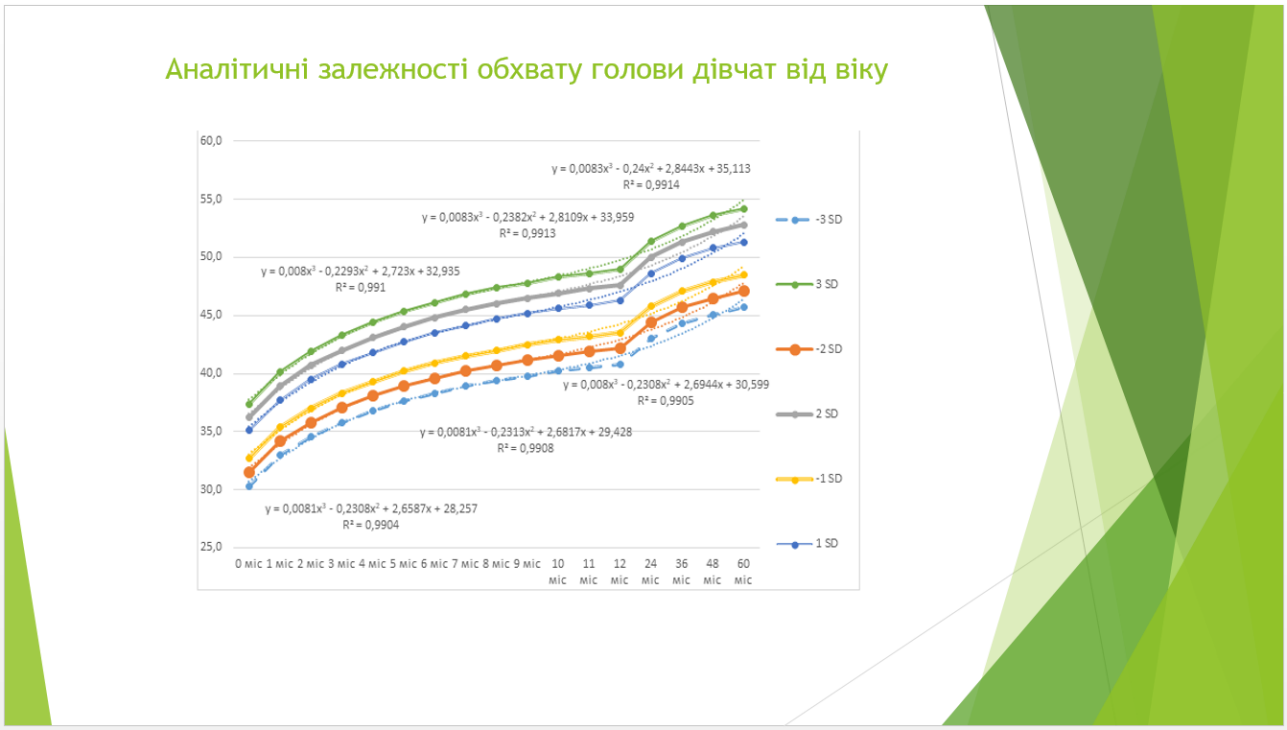


Рисунок В.11 – Слайд 11. Аналітичні залежності обхвату голови дівчат від віку

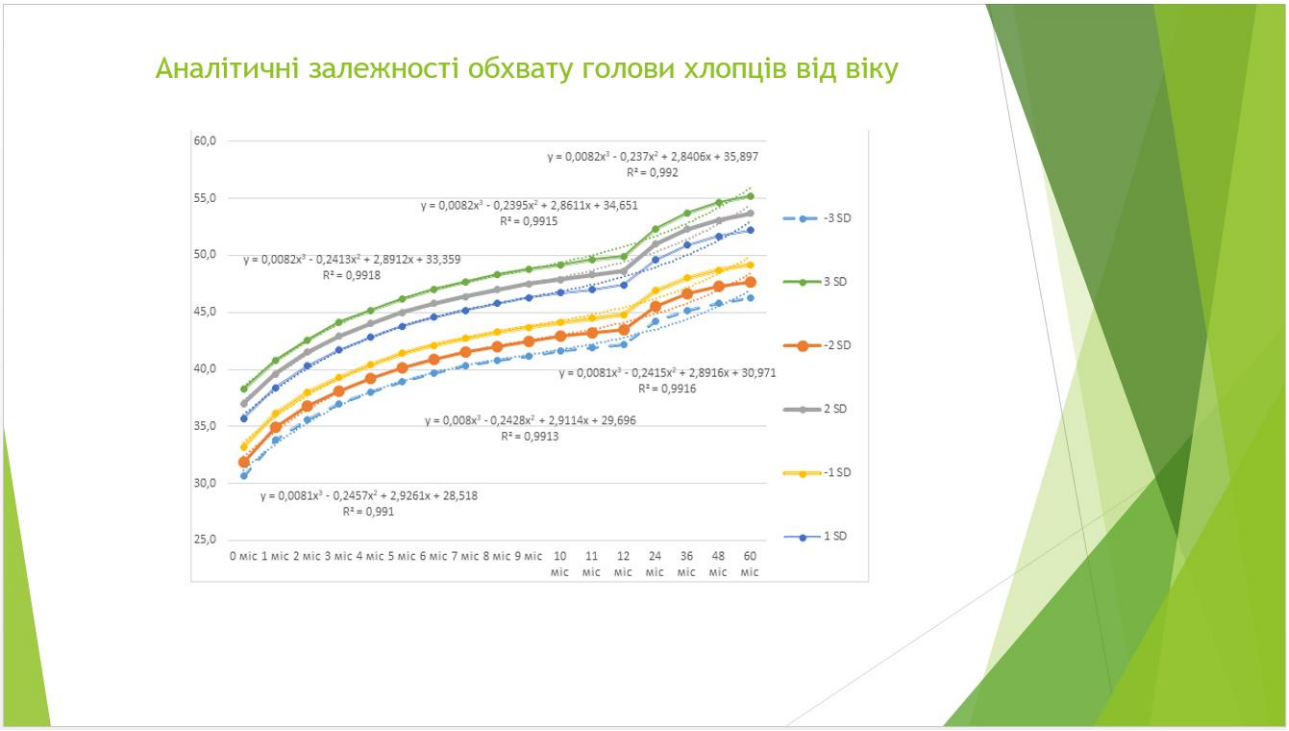


Рисунок В.12 – Слайд 12. Аналітичні залежності обхвату голови хлопців від віку

Рисунок В.13 – Слайд 13. Блок-схеми алгоритмів роботи розроблених методів

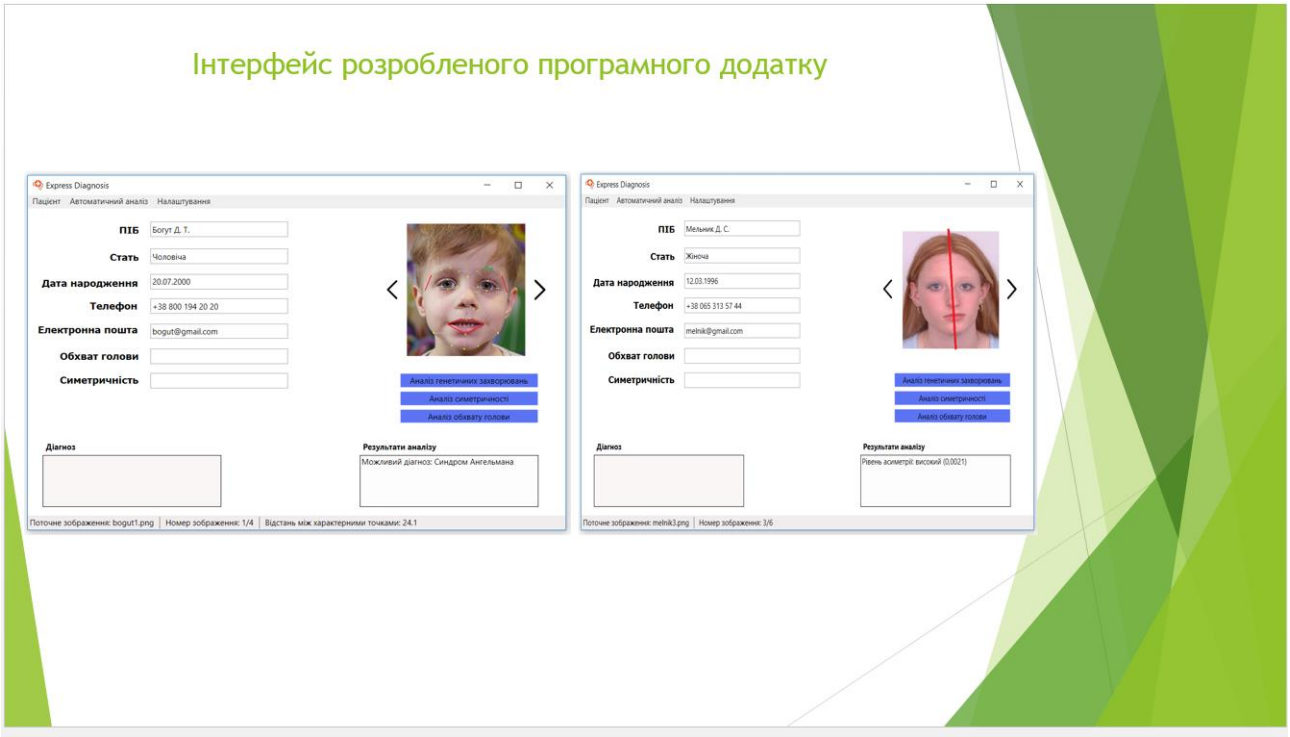


Рисунок В.14 – Слайд 14. Інтерфейс розробленого програмного додатку

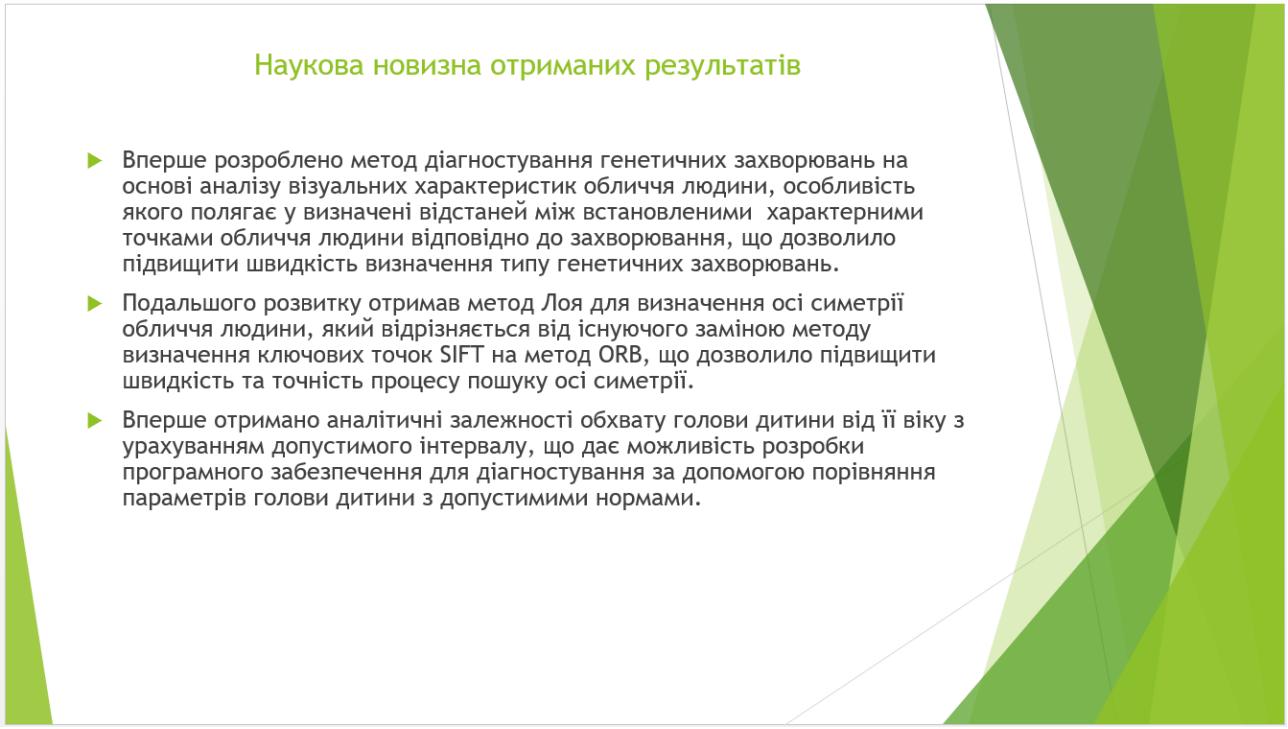


Рисунок В.15 – Слайд 15. Наукова новизна отриманих результатів

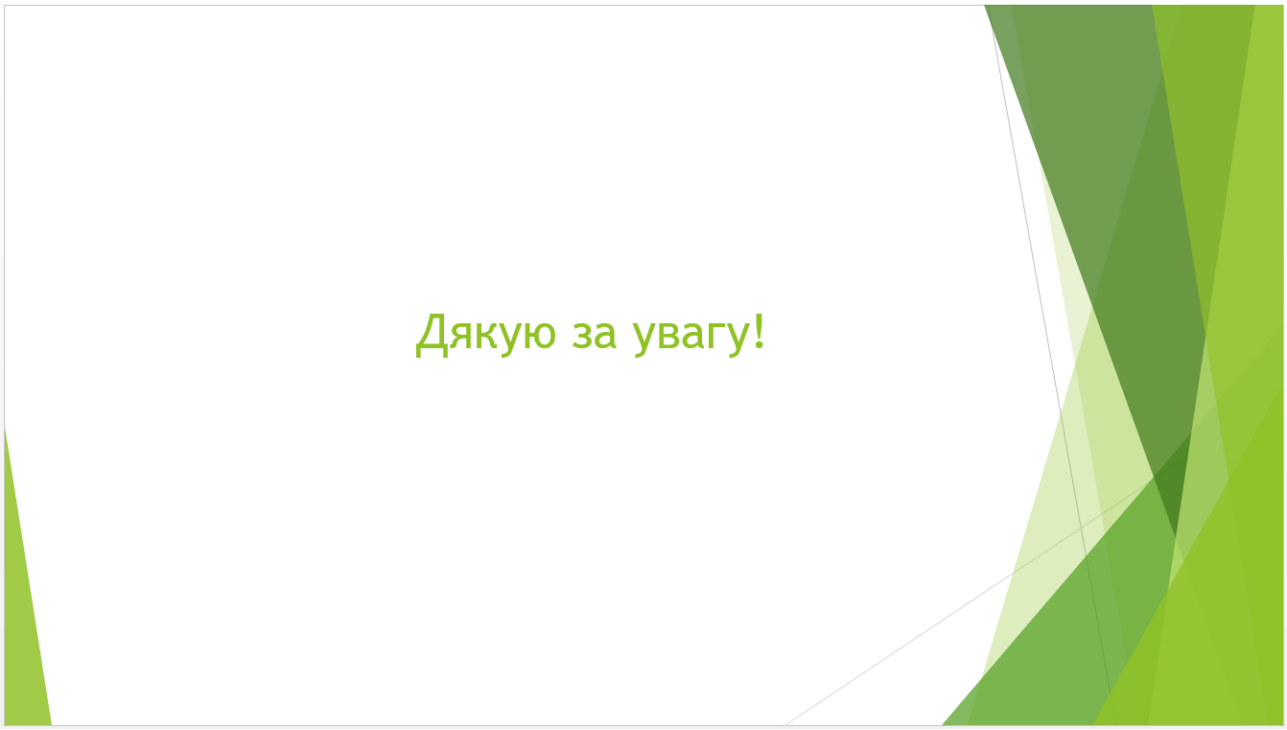


Рисунок В.16 – Слайд 16. Дякую за увагу