Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ РЕКОНСТРУКЦІЇ ТРИВИМІРНИХ ЗОБРАЖЕНЬ
RESEARCH AND SOFTWARE IMPLEMENTATION OF 3D IMAGE RECONSTRUCTION METHODS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-222м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

ДЕНИСЕНКО В.В.
(ПРІЗВИЩЕ та ініціали)
Керівник КОЦУР М.І.
(ПРІЗВИЩЕ та ініціали)
Рецензент СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)

2023

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДЕНИСЕНКО Владислави Вадимівни
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів реконструкції тривимірних зображень. Research and Software Implementation of 3D Image Reconstruction Methods

керівник проєкту (роботи) к.т.н., доцент, КОЦУР Михайло Ігорович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 14 " листопада 2023 року № 437
2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЦУР М.І., професор		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 08 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження	8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тижні	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка) _____ Владислава ДЕНИСЕНКО
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Михайло КОЦУР
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 92 с., 3 табл., 26 рис., 3 дод., 28 джерел.

PYTHON, PYCHARM, РЕКОНСТРУКЦІЯ ЗОБРАЖЕНЬ, МОВА ПРОГРАМУВАННЯ, ТРИВИМІРНІ ЗОБРАЖЕННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об’єкт дослідження – процес реконструкції тривимірних зображень.

Предмет дослідження – програмні засоби для реконструкції тривимірних зображень.

Метою роботи є дослідження та програмна реалізація методів реконструкції тривимірних зображень.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm.

Результати. Розроблено програмне забезпечення для реконструкції тривимірних зображень за допомогою мови програмування Python та середовища розробки PyCharm.

Практична цінність роботи полягає у розробці програмного забезпечення, що реалізує реконструкції тривимірних зображень.

Висновки. Виконано проєктування програмного забезпечення для реконструкції тривимірних зображень. Розроблено програмне забезпечення для реконструкції тривимірних зображень. Здійснено тестування розробленого програмного забезпечення для реконструкції тривимірних зображень.

Галузь використання – медицина, системи контролю, системи біометричної аутентифікації.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 92 pages, 3 tables, 26 figures, 3 appendixes, 28 sources.

PYTHON, PYCHARM, IMAGE RECONSTRUCTION, PROGRAMMING LANGUAGE, THREE-DIMENSIONAL IMAGES, SOFTWARE.

The object of research is the process of reconstruction of three-dimensional images.

The subject of the research is software tools for reconstructing three – dimensional images.

The aim of the work is to study and programmatically implement methods for reconstructing three-dimensional images.

Materials, methods, and technical tools: Python programming language, PyCharm development environment.

Results. Software for reconstructing three-dimensional images using the Python programming language and the PyCharm development environment has been developed.

The practical value of the work lies in the development of software that implements reconstruction of three-dimensional images.

Conclusions. Software for reconstruction of three-dimensional images was designed. Software for reconstruction of three-dimensional images has been developed. The developed software for reconstruction of three-dimensional images was tested.

Scope of use-medicine, control systems, biometric authentication systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

2D – двовимірне;

3D – тривимірне;

AR – Artificial Reality;

CPD – Coherent Point Drift;

GMM – Gaussian Mixture Model;

ICP – Iterative Closest Points;

IDE – Integrated Development Environment;

JVM – віртуальній машині Java;

MVS – Multi-View Stereo;

PDF – Probability Density Function;

АОК – автоматизований оптичний контроль;

КВМ – координатно-вимірювальних машини;

ПЗ – програмне забезпечення.

ВСТУП

Дана робота присвячена приватній задачі з області тривимірної реконструкції зображень: відновленню тривимірної моделі обличчя людини [1]-[3]. Вхідними даними для завдання є кольорові плоскі зображення реконструйованого особи, на виході потрібно отримати тривимірну модель [1]-[3].

Щоб встановити тривимірну модель, потрібно пройти наступні етапи:

– знайти на представлених плоских зображеннях загальні об'єкти: точки, краю і т. д.;

– за знайденими відповідностями між ними встановити їх положення в тривимірному просторі;

– об'єднати тривимірні об'єкти в модель;

– при необхідності застосувати до моделі будь-які алгоритми для поліпшення її якості.

Перші три етапи загальні і можуть застосовуватися для відновлення довільних об'єктів, тоді як останній етап істотно залежить від типу відновлюваного об'єкта [1]-[3].

Людське обличчя – якраз той тип об'єктів, для якого особливо потрібно четвертий етап. Загальні методи реконструкції, застосовані до такого своєрідного об'єкту, як людське обличчя, дають незадовільні результати. Причини в тому, що не враховуються апріорно відомі особливості об'єкта: деякі фрагменти (щоки, лоб) являють собою рівні поверхні, мають бідну структуру і можуть описуватися невеликим числом даних; інші ж (очі, ніс, вуха, губи), навпаки, влаштовані складно і їх потрібно описувати більш детально [1]-[3]. Крім того, про людське обличчя є маса відомостей з анатомії, що дозволяє ввести явні параметри і таким чином отримувати більш точні реконструкції.

У роботі дається докладний огляд методів, що застосовуються на етапах а)-в), огляд існуючих підходів до побудови моделей осіб, і детально розбираються два алгоритми реєстрації осіб, що реалізують етап г) [1]-[3].

Актуальність роботи випливає з великих можливих областей застосування розглянутих алгоритмів. Це такі області, як віртуальна реальність (в будь-якому з її проявів: комп'ютерні ігри, створення тривимірної графіки для фільмів, доповнена реальність і т.д.), системи ідентифікації (наприклад, охоронні системи підприємств) [1]-[3].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Тривимірна реконструкція

Методи реконструкції тривимірних (3D) зображень стали потужними інструментами в різних областях, що дозволяють перетворювати двовимірні (2D) дані в складні об'ємні уявлення [4]-[6]. Сфери застосування цих методів різноманітні і охоплюють такі сфери, як медична візуалізація, комп'ютерний зір, Археологія, виробництво тощо. У цьому всебічному дослідженні розглядаються завдання, що виконуються методами реконструкції 3D-зображень, і методи, що використовуються в різних областях [4]-[6].

Медична візуалізація: отримання інформації на основі об'ємних даних. Реконструкція 3D-зображень є невід'ємною частиною перетворення послідовностей 2D-медичних зображень, таких як знімки комп'ютерної томографії (КТ) і магнітно-резонансної томографії (МРТ), в об'ємні представлення [4]-[6]. Це використовується для:

– планування операції – хірурги використовують 3D-реконструкції для детального передопераційного планування, що дозволяє їм візуалізувати складні анатомічні структури в трьох вимірах;

– діагностики захворювань – лікарі використовують 3D-візуалізацію для точної діагностики, отримуючи всебічне уявлення про внутрішні органи, пухлини і аномалії [4]-[6];

– оцінка лікування – моніторинг прогресу лікування полегшується шляхом порівняння 3D-зображень з часом, що дозволяє медичним працівникам оцінювати ефективність втручань [4]-[6];

– віртуальне дослідження – дослідники та викладачі використовують 3D-реконструкції для створення віртуальних анатомічних моделей для освітніх цілей та інтерактивного дослідження [4]-[6].

Реалізація цих задач виглядає наступним чином:

– реконструкція КТ – при комп'ютерній томографії складні алгоритми реконструюють зрізи поперечного перерізу в 3D обсяги, забезпечуючи детальне представлення внутрішніх структур [4]-[6];

– об'ємний рендеринг МРТ – передові методи МРТ дозволяють реконструювати 3D-зображення шляхом об'єднання даних з декількох зрізів, отримуючи детальну інформацію про м'які тканини [4]-[6];

– об'ємний рендеринг ультразвуку – 3D-реконструкції за даними ультразвуку покращують візуалізацію під час таких процедур, як візуалізація плода, підвищуючи точність діагностики [4]-[6];

– мультимодальне злиття – інтеграція даних з різних методів візуалізації, таких як КТ та МРТ, підвищує повноту та точність 3D-реконструкцій [4]-[6].

Комп'ютерний зір: розшифровка 3D-структури за 2D-зображеннями. 3D-реконструкція в комп'ютерному зорі передбачає виведення тривимірної структури об'єктів або сцен з 2D-зображень або відеопослідовностей [4]-[6].

Це використовується для:

– розпізнавання об'єктів – 3D-реконструкції допомагають у розпізнаванні об'єктів, надаючи додаткову інформацію про глибину, покращуючи здатність розрізняти об'єкти [4]-[6];

– доповненої реальності (artificial reality – AR) – програми AR використовують 3D-реконструкції для плавного накладання віртуальних об'єктів на реальне середовище, створюючи захоплюючий досвід користувачів [4]-[6];

– автономної навігації – роботизовані системи використовують 3D-реконструкції для просторового сприйняття, дозволяючи роботам і автономним транспортним засобам орієнтуватися в складних середовищах;

– розуміння сцени – аналіз 3D-структури сцени сприяє кращому розумінню сцени, підтримуючи програми в робототехніці, спостереженні та віртуальній реальності [4]-[6].

Реалізуються такі задачі наступним чином:

– структура руху – методи структури руху реконструюють тривимірні структури, аналізуючи рух камери або датчика з часом, використовуючи відповідність об'єктів між зображеннями [4]-[6];

– стереовізія – використовуючи два або більше зображень з різних точок зору, стереовізійні методи тріангулюють точки для оцінки глибини та реконструкції 3D-сцен;

– технології визначення глибини – датчики глибини, такі як лідар і камери з фіксацією часу польоту, надають пряму інформацію про глибину, полегшуючи точну 3D-реконструкцію;

– фотограмметрія – фотограмметричні методи передбачають отримання тривимірної інформації з фотографій, які часто використовуються в таких додатках, як Геодезія та документування культурної спадщини [4]-[6].

Тривимірне моделювання поверхні: з'єднання реального і віртуального світів. 3D-моделювання поверхонь фокусується на створенні детальних уявлень поверхонь з 2D-зображень або хмар точок [4]-[6].

Автоматизоване проєктування – інженери та дизайнери використовують 3D-моделі поверхонь для проєктування виробів та конструкцій у віртуальному середовищі.

Індустрія розваг використовує 3D-моделювання поверхні для створення анімованих персонажів, навколишнього середовища та спецефектів у фільмах та відеоіграх [4]-[6].

Промисловий дизайн: дизайнери використовують 3D-моделі для створення прототипів і доопрацювання промислових продуктів перед фізичним виробництвом.

Віртуальна реальність – програми віртуальної реальності покладаються на 3D-моделі поверхні, щоб занурити користувачів у реалістичне віртуальне середовище.

Алгоритми реконструкції поверхні: ці алгоритми перетворюють хмари точок у гладкі 3D-поверхні, фіксуючи геометрію об'єктів [4]-[6].

Моделювання полігональної сітки: представлення поверхонь у вигляді полігональних сіток є поширеним методом в 3D-моделюванні, що дозволяє ефективно зберігати дані про поверхні і маніпулювати ними [4]-[6].

Реєстрація хмар точок: вирівнювання та об'єднання хмар точок з декількох джерел сприяють більш повним 3D-моделям поверхні.

Відображення текстур: додавання текстур до 3D-поверхонь підвищує візуальний реалізм, роблячи моделі більш привабливими та деталізованими.

Дистанційне зондування: відображення особливостей землі в трьох вимірах. При дистанційному зондуванні 3D-реконструкція включає в себе створення цифрових моделей рельєфу і зйомку рельєфу земної поверхні.

Моніторинг навколишнього середовища: 3D-реконструкції місцевості допомагають відстежувати зміни в навколишньому середовищі, такі як вирубка лісів, землекористування та стихійні лиха [4]-[6].

Містобудування: містобудівники використовують 3D-моделі для візуалізації міських ландшафтів, планування інфраструктурних проєктів та моделювання впливу запропонованих розробок.

Реагування на стихійні лиха: 3D-реконструкції підтримують зусилля з реагування на стихійні лиха, надаючи детальну інформацію про постраждалих районах і допомагаючи в розподілі ресурсів [4]-[6].

Картографія: створення точних топографічних карт ґрунтується на точних 3D-реконструкціях земної поверхні.

Стереофотограмметрія: подібно до стереозображення, методи стереофотограмметрії використовують перекриваються аерофотознімки для реконструкції місцевості в 3D [4]-[6].

Виявлення світла і визначення дальності: технологія виявлення світла і визначення дальності забезпечує точні дані про висоту з високою роздільною здатністю, сприяючи детальної 3D-реконструкції.

Радар із синтезованою апертурою: такі дані, отримані від радарних систем, використовуються в 3D-реконструкції для картографування місцевості та виявлення змін [4]-[6].

Мультиспектральна візуалізація: Об'єднання даних з різних спектральних діапазонів розширює інформацію, доступну для 3D-реконструкції, підвищуючи точність моделей місцевості [4]-[6].

Археологія та культурна спадщина: збереження минулого в цифровому вигляді. 3D-реконструкція в археології передбачає створення цифрових моделей археологічних пам'яток, артефактів і об'єктів культурної спадщини. Цифрове збереження артефактів та історичних об'єктів дозволяє документувати їх та зберігати для майбутніх поколінь [4]-[6]:

– археологи використовують 3D-реконструкції для аналізу структури, геометрії та деталей артефактів, сприяючи археологічним дослідженням;

– планування реставрації: 3D-моделі допомагають планувати відновлення пошкоджених або занепадаючих об'єктів культурної спадщини та споруд;

– віртуальні музеї: 3D-реконструкції сприяють створенню віртуальних музеїв, надаючи громадськості доступні платформи для вивчення культурної спадщини [4]-[6].

Реалізується це у вигляді:

– фотограмметричної документації – фотограмметрія широко використовується для документування археологічних об'єктів та артефактів, отримання детальної 3D-інформації з фотографій;

– лазерного сканування – високоточні технології лазерного сканування створюють деталізовані 3D-моделі артефактів і структур, зберігаючи дрібні деталі [4]-[6].

– Multi-View Stereo (MVS): алгоритми MVS реконструюють 3D-моделі шляхом аналізу декількох видів об'єкта, створюючи деталізовані і текстуровані уявлення [4]-[6].

– цифрових візуалізації та 3D-друку – створення 3D-моделей полегшує використання 3D-друку для відтворення артефактів або створення фізичних копій для навчальних цілей.

Промислова Метрологія: забезпечення точності при виробництві. Тривимірна реконструкція в промисловій метрології передбачає вимірювання розмірів і контроль якості у виробничих процесах [4]-[6].

Це використовується для:

– контролю якості – 3D-реконструкції використовуються для перевірки виготовлених деталей, щоб переконатися, що вони відповідають заданим розмірам і стандартам якості [4]-[6];

– зворотного проєктування – створення 3D-моделей існуючих деталей або виробів підтримує процеси зворотного проєктування для поліпшення дизайну або тиражування;

– оптимізації складальної лінії – аналіз 3D-реконструкцій компонентів допомагає оптимізувати процеси складання і забезпечити точну підгонку [4]-[6];

– неруйнівного контролю –методи 3D-реконструкції дозволяють проводити неруйнівний контроль, зменшуючи потребу у фізичному розбиранні [4]-[6].

Реалізується у вигляді:

– сканування структурованим світлом – проєктування структурованих світлових візерунків на поверхні дозволяє виконувати точні 3D-реконструкції, особливо в ситуаціях, що вимагають високої точності [4]-[6];

– координатно-вимірювальних машини (КВМ) – об'єднання даних КВМ з 3D-реконструкціями підвищує точність вимірювань в промисловій метрології;

– КТ на виробництві – КТ використовується для 3D-реконструкції на виробництві для огляду внутрішніх структур і виявлення дефектів [4]-[6];

– автоматизований оптичний контроль (АОК) – системи АОК використовують 3D-реконструкції для огляду поверхонь і виявлення дефектів на виготовлених виробах.

Біомеханіка: аналіз руху та структури в біологічній сфері. Тривимірна реконструкція в біомеханіці передбачає аналіз руху та структури біологічних об'єктів, таких як людське тіло або тварини [4]-[6].

Біомеханічні дослідження: дослідники використовують 3D-реконструкції для аналізу механіки руху, розуміння сил та взаємодій у біологічних системах.

1.2 Аналіз існуючих рішень

Для реконструкції 3D-зображень в різних областях широко використовується різне програмне забезпечення (ПЗ). Популярність цих інструментів часто залежить від конкретних вимог програми, характеру вхідних даних та бажаного рівня точності.

Ці програмні рішення підходять для широкого спектру застосувань, від медичної візуалізації до комп'ютерного зору та промислової метрології. Вибір конкретного інструменту часто залежить від конкретних потреб користувача та характеристик оброблюваних даних. Дослідники та практики можуть вибирати ПЗ на основі таких факторів, як простота використання, підтримка спільноти, доступність певних алгоритмів та можливості інтеграції з іншими інструментами чи робочими процесами.

Розглянемо деякі з найбільш популярних програмних рішень для реконструкції 3D-зображень:

1.2.1 Застосунок ImageJ

ImageJ – це програма обробки зображень на базі Java, розроблена в Національному інституті охорони здоров'я та лабораторії оптичного та обчислювального приладобудування (Університет Вісконсіна) [7], [8]. Перша версія, ImageJ 1.x, розроблена у відкритому доступі, в той час як ImageJ2 і пов'язані з ним проєкти SciJava, ImgLib2 і SCIFIO ліцензовані по дозвільної

ліцензії BSD-2. ImageJ був розроблений з відкритою архітектурою, яка забезпечує розширюваність за допомогою плагінів Java та записуваних макросів. Користувацьке отримання, плагіни для аналізу та обробки можуть бути розроблені за допомогою вбудованого редактора ImageJ та компілятора Java [7], [8]. Написані користувачем плагіни дозволяють вирішувати багато завдань обробки і аналізу зображень, від тривимірної візуалізації живих клітин до радіологічної обробки зображень, порівняння даних декількох систем візуалізації до автоматизованих гематологічних систем. Архітектура плагінів ImageJ та вбудоване середовище розробки зробили його популярною платформою для навчання обробки зображень [7], [8].

ImageJ можна запускати як онлайн-аплет, завантажувану програму або на будь-якому комп'ютері з віртуальною машиною Java 5 або новішої версії. Завантажувані дистрибутиви доступні для Microsoft Windows, класичної Mac OS, macOS, Linux. Вихідний код ImageJ знаходиться у вільному доступі на вебсайті [7], [8].

ImageJ може відображати, редагувати, аналізувати, обробляти, зберігати та друкувати 8-розрядні кольорові та сірі зображення, 16-розрядні цілі числа та 32-розрядні зображення з плаваючою комою [7], [8]. Застосунок може читати багато форматів файлів зображень, включаючи TIFF, PNG, GIF, JPEG, BMP, DICOM та FITS, а також формати raw. ImageJ підтримує стеки зображень, серії зображень, які спільно використовують одне вікно, і є багатопотоковим, тому трудомісткі операції можуть виконуватися паралельно на багатопроцесорному обладнанні. с може обчислювати статистику площі та значень пікселів для вибраних користувачем об'єктів та об'єктів з пороговим значенням інтенсивності. Він може вимірювати відстані та кути. Він може створювати гістограми щільності та графіки профілю лінії [7], [8]. Він підтримує стандартні функції обробки зображень, такі як логічні та арифметичні операції між зображеннями, маніпулювання контрастом, згортка, аналіз Фур'є, різкість, згладжування, виявлення країв та фільтрація медіани. Застосунок виконує геометричні перетворення, такі як масштабування,

поворот і перевороти. Програма підтримує будь-яку кількість зображень одночасно, обмежену лише доступною пам'яттю [7], [8].

Скріншот ImageJ представлено на рис.1.1 [8].

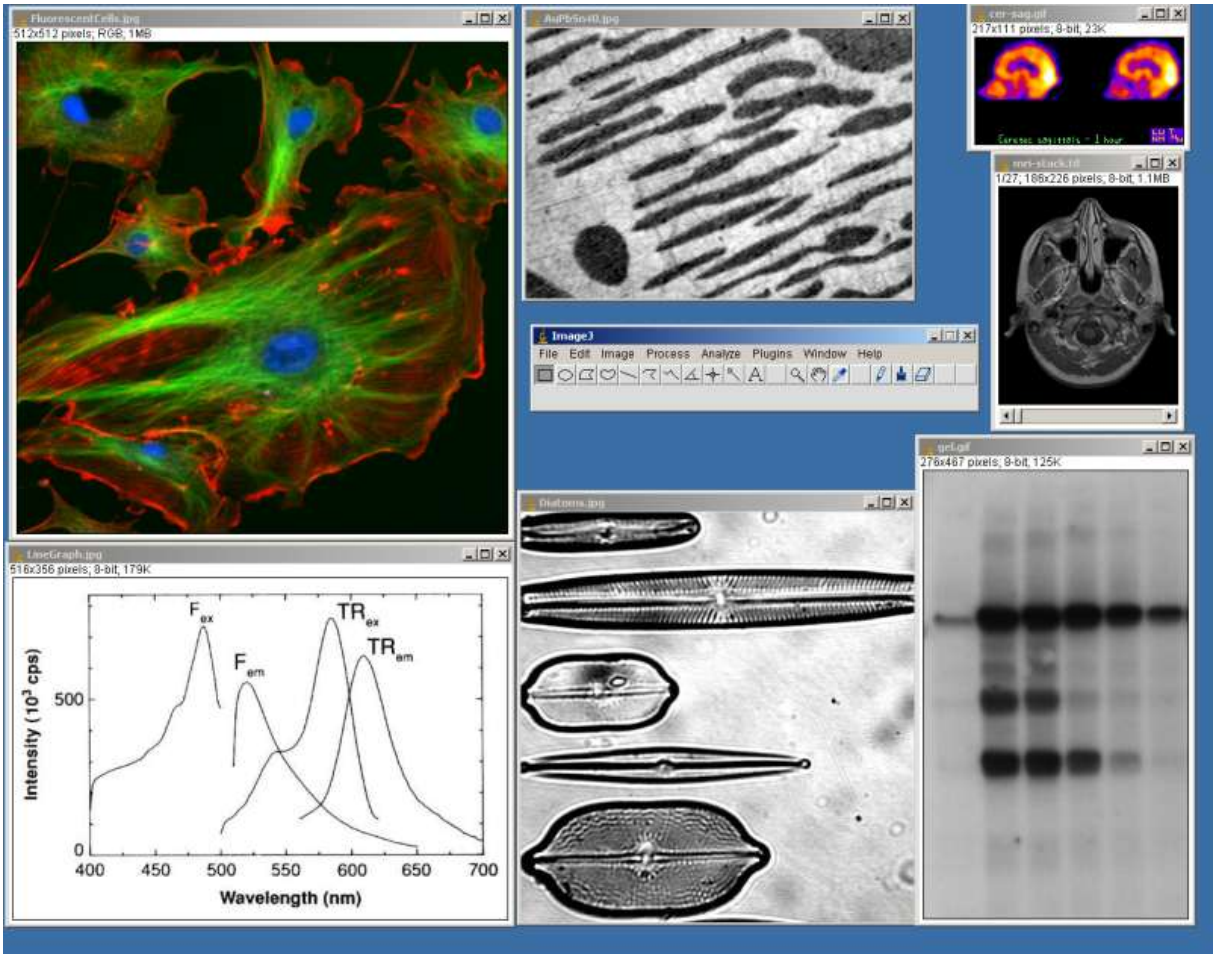

Рисунок 1.1 – Скріншот ImageJ [8]

1.2.2 Програмний пакет 3D Slicer

3D Slicer – це безкоштовний програмний пакет з відкритим вихідним кодом для аналізу зображень і наукової візуалізації [9], [10]. 3D Slicer використовується в різних галузях медицини, включаючи аутизм, розсіяний склероз, системний червоний вовчак, рак передміхурової залози, рак легенів, рак молочної залози, шизофренію, ортопедичну біомеханіку, серцево-судинні захворювання та нейрохірургію [9], [10].

3D Slice – це безкоштовне ПЗ з відкритим кодом, яке є гнучкою модульною платформою для аналізу та візуалізації зображень. 3D Slicer розширений, щоб забезпечити розробку як інтерактивних, так і пакетних інструментів обробки для різних додатків [9], [10].

3D Slicer забезпечує реєстрацію зображень, обробку дифузійної трактографії, інтерфейс до зовнішніх пристроїв для підтримки наведення зображення і об'ємний рендеринг з підтримкою GPU, серед інших можливостей. 3D Slicer має модульну організацію, яка дозволяє додавати нові функціональні можливості та надає ряд загальних функцій, недоступних у конкуруючих інструментах [9], [10].

Можливості інтерактивної візуалізації 3D Slicer включають в себе можливість відображення довільно орієнтованих фрагментів зображення, побудова моделей поверхні по мітках зображень і апаратно прискорений об'ємний рендеринг. 3D Slicer також підтримує багатий набір функцій анотації (фідуціарні та вимірювальні віджети, настроювані кольорові карти) [9], [10].

Можливості 3D Slicer включають:

– обробка зображень спеціалізованих медичних форматів та читання/запис багатьох інших форматів;

– інтерактивна візуалізація об'ємних воксельних зображень, полігональних сіток і об'ємних рендерів;

– ручне редагування;

– об'єднання і спільна реєстрація даних з використанням жорстких і нежорстких алгоритмів;

– автоматична сегментація зображень;

– аналіз та візуалізація даних тензорної дифузії зображень;

– відстеження пристроїв для процедур, керованих зображеннями.

3D Slicer скомпільовано для використання на декількох обчислювальних платформах, включаючи Windows, Linux та macOS [9], [10].

3D Slicer розповсюджується під безкоштовною ліцензією з відкритим кодом. Ліцензія не містить обмежень на використання програмного

забезпечення в академічних або комерційних проєктах. Однак не стверджується, що програмне забезпечення корисне для будь-якого конкретного завдання. Користувач несе повну відповідальність за дотримання місцевих норм і приписів. Пристрій для нарізки не був офіційно схвалений для клінічного застосування Управлінням з контролю за продуктами і ліками США або будь-яким іншим регулюючим органом в інших країнах [9], [10].

Скріншот програмного пакету 3D Slicer представлено на рис.1.2 [10].

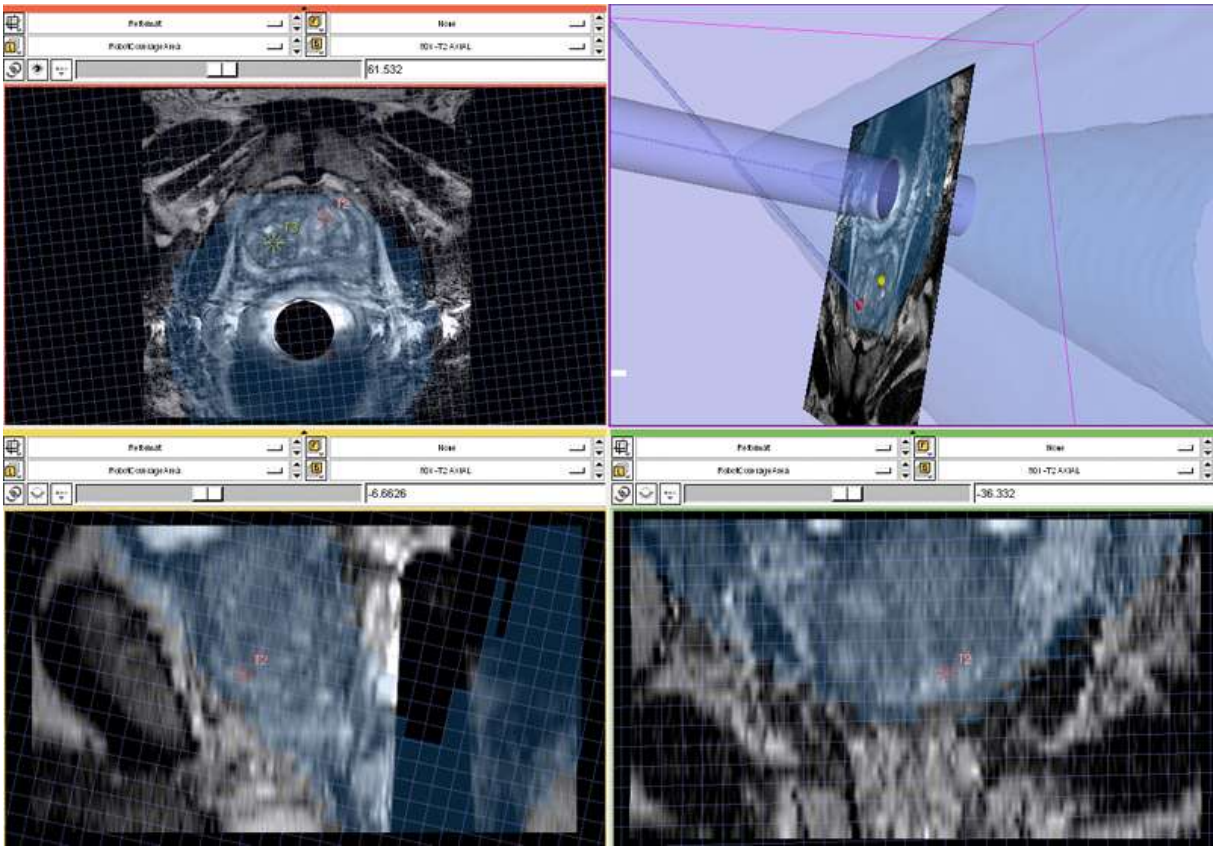

Рисунок 1.2 – Скріншот програмного пакету 3D Slicer [10]

1.2.3 Програма ParaView

ParaView – це кросплатформна програма з відкритим кодом для інтерактивної наукової візуалізації [11], [12]. Він має архітектуру клієнт-сервер для полегшення віддаленої візуалізації наборів даних і генерує моделі рівня деталізації для підтримки інтерактивної частоти кадрів для великих наборів даних. Це програма, створена поверх бібліотек Visualization Toolkit.

ParaView – це програма, призначена для паралелізації даних на мультикомп'ютерах та кластерах із спільною або розподіленою пам'яттю. Її також можна запускати як застосунок на одному комп'ютері [11], [12].

ParaView – це багатоплатформна програма для аналізу та візуалізації даних з відкритим кодом. Застосунок ParaView відомий і використовується в багатьох різних спільнотах для аналізу та візуалізації наборів наукових даних. Він може бути використаний для створення візуалізацій для аналізу даних за допомогою якісних та кількісних методів. Дослідження даних можна проводити інтерактивно в 3D або програмно, використовуючи можливості пакетної обробки ParaView [11], [12].

ParaView був розроблений для аналізу надзвичайно великих наборів даних з використанням обчислювальних ресурсів розподіленої пам'яті. Його можна запускати на суперкомп'ютерах для аналізу наборів даних тераскейла, а також на ноутбуках для обробки даних меншого розміру [11], [12].

ParaView – це прикладний фреймворк, а також готовий застосунок. База коду ParaView спроєктована таким чином, що всі її компоненти можуть бути повторно використані для швидкої розробки вертикальних застосунків. Така гнучкість дозволяє розробникам ParaView швидко розробляти додатки, що володіють специфічною функціональністю для конкретної предметної області [11], [12].

ParaView працює на паралельних та однопроцесорних системах з розподіленою та спільною пам'яттю. Він був успішно протестований на Windows, macOS, Linux, IBM Blue Gene, Cray XT3 та різних робочих станціях Unix, кластерах та суперкомп'ютерах. Під капотом ParaView використовує Visualization Toolkit як механізм обробки даних та візуалізації та має інтерфейс користувача, написаний за допомогою Qt [11], [12].

Цілі команди ParaView включають наступне:

– розроблення мультиплатформного застосунку для візуалізації з відкритим кодом;

– підтримка моделей розподілених обчислень для обробки великих наборів даних;

– створення відкритого, гнучкого та інтуїтивно зрозумілого інтерфейсу користувача;

– розроблення розширюваної архітектури на основі відкритих стандартів.

Скріншот використання ParaView представлено на рис.1.3 [12].

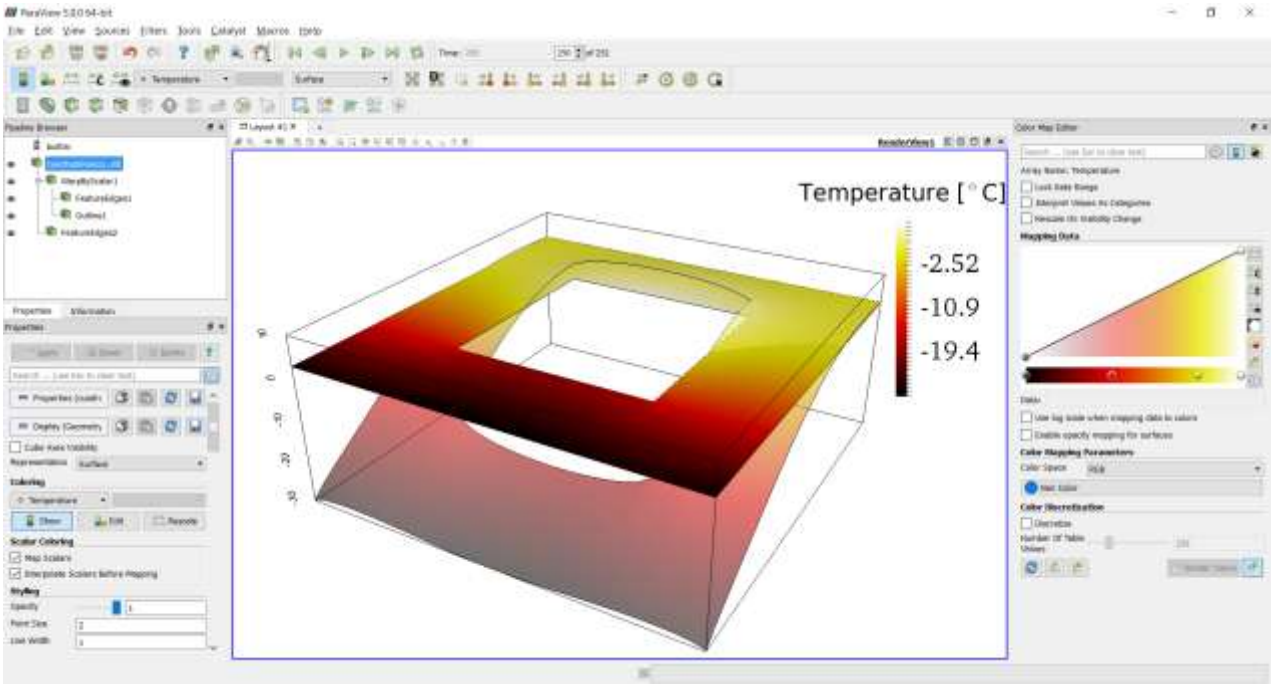

Рисунок 1.3 – Скріншот використання ParaView [12]

1.2.4 Порівняння існуючих рішень

Порівняльну характеристику ПЗ для реконструкції тривимірних зображень наведено у таблиці 1.1.

Порівняльна таблиця надає детальний огляд ключових функцій і характеристик. Користувачам слід враховувати конкретні вимоги і переваги при виборі ПЗ. Відтак, ParaView є більш узагальненим та зосередженим на науковій та інженерній візуалізації, тоді як ImageJ та 3D Slicer більше орієнтовані на природничі науки та медичну візуалізацію відповідно.

Таблиця 1.1 – Порівняння методів для пошуку спільнот користувачів у соціальних мережах

	ImageJ	3D Slicer	ParaView
Спрямованість застосунків	Науки про життя, мікроскопія	Медична візуалізація, наукова візуалізація	Наукова та інженерна візуалізація
Відкритий вихідний код	Так	Так	Так
Інтерфейс користувача	Зручний у використанні, особливо для біологів	Великий, розроблений спеціально для медичних дослідників	Багатофункціональна, підходить для наукових робочих процесів
Розширення / плагіни	Доступно багато плагінів	Велика колекція модулів і розширень	Підтримує плагіни та сценарії користувачів
3D візуалізація	Підтримує 3D візуалізацію	Спеціалізується на медичній 3D-візуалізації	Розширені можливості 3D-візуалізації
Підтримка спільноти	Активна спільнота, особливо в галузі наук про життя	Активна спільнота, особливо в галузі медицини	Активна спільнота з різноманітною базою користувачів
Підтримка сценаріїв	Так (макромова, плагіни можуть бути написані за сценарієм)	Так (Python, сценарії для Slicer)	Так (Python, сценарії користувачів)
Спеціалізовані інструменти	Плагіни для конкретних додатків у галузі наук про життя	Модулі для аналізу медичних зображень	Універсальний з акцентом на великі набори даних
Формати даних	Підтримує різні формати зображень	Підтримує формати медичних зображень (DICOM, NIfTI)	Підтримує різні формати даних
Простота використання	Зручний для користувача з простим інтерфейсом	Багатофункціональний, може мати більш круту криву навчання	Багатофункціональний, вимагає деякого навчання
Застосування в дослідженнях	Зазвичай використовується в наукових дослідженнях	Широко використовується в медичних дослідженнях та освіті	Застосовується в наукових дослідженнях та інженерії
Паралельна обробка	Обмежений	Обмежений	Підтримує паралельну обробку великих наборів даних

3D Slicer розроблений спеціально для медичних дослідників зі спеціалізованими модулями, що робить його комплексним рішенням для аналізу медичних зображень. ParaView чудово справляється з великими наборами даних та паралельною обробкою, що робить його придатним для складних наукових та інженерних візуалізацій.

Зрештою, вибір між цими інструментами залежить від конкретного домену користувача, уподобань та характеру завдань 3D-реконструкції, які він має намір виконати.

Проте, хоча ImageJ, 3D Slicer і ParaView є потужними і широко використовуваними програмними рішеннями для різних завдань обробки зображень і візуалізації, вони усі мають певні обмеження або недоліки. ImageJ, хоча і підтримує базову 3D-візуалізацію, може не пропонувати той самий рівень розширених функцій 3D-візуалізації, що й деякі інші спеціалізовані інструменти. Більш того, обробка дуже великих наборів даних або декількох вимірювань може призвести до обмежень пам'яті, що вплине на продуктивність та зручність використання. Ну а загалом, застосунок може бути ресурсоємним, особливо при роботі з великими наборами медичних даних, що може вплинути на продуктивність на менш потужному обладнанні. 3D Slicer та ParaView в першу ж чергу призначені для наукової та інженерної візуалізації, і користувачі з інших областей можуть вважати їх функції менш безпосередньо застосовними до їхніх потреб.

Саме тому актуальною задачею є розробка нових методів та засобів для реконструкції тривимірних зображень, які б надавали універсальний функціонал з реконструкції зображень, які в подальшому могли б використовуватися в різних прикладних галузях та задачах, з більш оптимізованим використанням обчислювальних потужностей.

1.3 Постановка задачі

Метою роботи є дослідження та розробка методів реконструкції тривимірних зображень.

У роботі розробляється система реконструкції тривимірної моделі людського обличчя по його двовимірним зображенням. Наведемо кілька прикладів можливих областей застосування розроблюваної системи:

– ідентифікація людини за фрагментами відеозапису і знімків систем спостереження, що може знайти широке застосування в комплексах ПЗ для охоронних систем;

– побудова віртуального альтер-его користувача, що може бути використане в системах віртуальної реальності як розважального, так і професійного призначення;

– створення високоякісних моделей особи людини без необхідності їх ручного доопрацювання дизайнером і фахівцем з 3D-моделювання.

У кожній з наведених областей застосування розроблюваної системи дозволить зменшити або навіть виключити використання дорогої праці фахівців, тим самим знизивши витрати і підвищивши економічну ефективність.

1.4 Висновки за розділом 1

Визначено, що реконструкція тривимірних зображень є важливою задачею та широко використовується у різних сферах людської діяльності.

Проаналізовано основні програмні рішення, що реалізують вирішення цієї задачі. Серед усіх існуючих рішень є недоліки.

Тому актуальною є задача розробки нового підходу у вигляді методу та програмної реалізації для реконструкції тривимірних зображень.

Сформульовано перелік функціональних вимог до ПЗ.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

Наука про дані (Data science) – міждисциплінарна галузь, яка поєднує знання домену, навички програмування та статистичні знання для отримання інформації з даних, в останні роки спостерігала сплеск популярності [13]-[15]. Серед багатьох доступних мов програмування Python виділяється найширшим вибором інструментів для розробки, пов'язаної з наукою про дані. Мова Python стала фактичною мовою науки про дані, розглянемо це на прикладі її універсальності, екосистеми, простоти використання та підтримки спільноти [13]-[15].

Універсальність. Python – це мова програмування загального призначення, яка забезпечує універсальну основу для роботи з даними. Її синтаксис чіткий і читабельний, роблять її доступною як для початківців, так і для досвідчених програмістів. Така чіткість синтаксису прискорює процес розробки, сприяючи плавному переходу від створення коду до аналізу [13]-[15].

Широка застосовність. Універсальність Python виходить за рамки науки про дані. Вона використовується у веброзробці, штучному інтелекті, машинному навчанні, автоматизації, наукових обчисленнях тощо. Ця широка застосовність робить Python ідеальним вибором для приватних осіб та організацій, які шукають єдину мову для різних проєктів [13]-[15].

Сумісність. Python перевершує інші мови та інструменти щодо сумісності. Інтеграція з такими мовами, як C та C++, є плавною, що дозволяє виконувати завдання, важливі для продуктивності, на цих мовах, використовуючи Python для логіки високого рівня. Більше того, Python може взаємодіяти з бібліотеками та інструментами, написаними такими мовами, як Java та R, підвищуючи його сумісність [13]-[15].

Багата екосистема: багато бібліотек та фреймворків. NumPy та SciPy – основи наукових обчислень. Бібліотеки NumPy та SciPy формують основу наукових обчислювальних можливостей Python. NumPy забезпечує підтримку великих багатовимірних масивів і матриць, А також набір математичних функцій для роботи з цими масивами. SciPy розширює ці можливості, пропонуючи додаткові функціональні можливості для оптимізації, обробки сигналів та статистичних операцій [13]-[15].

Pandas – простіша обробка даних. Pandas, потужна бібліотека для маніпулювання даними, полегшує обробку та аналіз структурованих даних. У ній представлені дві фундаментальні структури даних, DataFrame і Series, які дозволяють інтуїтивно маніпулювати наборами даних. Pandas відмінно справляється з такими завданнями, як очищення, об'єднання, зміна форми і агрегування даних, оптимізуючи етап підготовки даних для аналізу [13]-[15].

Matplotlib та Seaborn: майстерність візуалізації. Можливості візуалізації в Python збагачені Matplotlib та Seaborn. Matplotlib, велика бібліотека побудови графіків, пропонує безліч настроюваних діаграм і графіків. Побудований поверх Matplotlib, Seaborn забезпечує інтерфейс високого рівня для створення естетично привабливих статистичних візуалізацій. Разом вони дозволяють фахівцям з обробки даних ефективно передавати інформацію [13]-[15].

Scikit-Learn – спрощене машинне навчання. Scikit-Learn, бібліотека машинного навчання для Python, інкапсулює великий набір алгоритмів для таких завдань, як класифікація, регресія, кластеризація та зменшення розмірності. Його стандартизований інтерфейс полегшує впровадження та оцінку моделей машинного навчання, що робить його наріжним каменем в інструментарії фахівців з обробки даних і практиків машинного навчання [13]-[15].

TensorFlow та PyTorch: домінування глибокого навчання. Python панує над глибоким навчанням завдяки TensorFlow та PyTorch. Розроблений Google,

TensorFlow відомий своєю гнучкістю та масштабованістю при побудові та навчанні глибоких нейронних мереж. PyTorch, Підтримуваний Facebook, відомий своїм динамічним обчислювальним графіком, що забезпечує більш інтуїтивний підхід до розробки моделей глибокого навчання. Обидві бібліотеки отримали широке поширення та підтримку дослідницьких та промислових спільнот [13]-[15].

Statsmodels – статистичне моделювання на Python. Для статистичного моделювання та перевірки гіпотез Statsmodels є безцінною бібліотекою. Вона пропонує повний набір статистичних моделей, тестів гіпотез та інструментів для дослідження статистичних даних. За допомогою Statsmodels науковці даних можуть проводити ретельний Статистичний аналіз, щоб зробити значущі висновки зі своїх наборів даних [13]-[15].

Блокноти Jupyter – інтерактивна наука про дані. Блокноти Jupyter ілюструють прихильність Python до інтерактивних обчислень. Ці блокноти забезпечують інтерактивне середовище з можливістю спільного використання, в якому співіснують код, візуалізації та розповідний текст. Блокноти Jupyter широко використовуються для дослідницького аналізу даних, прототипування моделей машинного навчання та створення навчального контенту [13]-[15].

Інтеграція з інструментами обробки великих даних. Python легко інтегрується з інструментами та фреймворками для роботи з великими даними, такими як Apache Spark та Apache Hadoop. PySpark, API Python для Apache Spark, дозволяє науковцям даних використовувати можливості розподілених обчислень для широкомасштабної обробки та аналізу даних [13]-[15].

Простота використання: усунення бар'єрів для входу. Читабельність і стислість. Синтаксис Python підкреслює читабельність і лаконічність – філософію дизайну, втілену в Python. Така читабельність сприяє спільній роботі, оскільки код часто читається кількома зацікавленими сторонами

протягом усього його життєвого циклу. Використання відступів замість фігурних дужок для роздільників блоків підвищує ясність коду [13]-[15].

Об’ємна документація. Python може похвалитися вичерпною документацією як основної мови, так і багатьох її бібліотек. Наявність зрозумілої і доступної документації прискорює процес навчання для початківців і служить цінним довідником для досвідчених практиків [13]-[15].

Швидке прототипування. Динамічне введення тексту та інтерпретований характер Python сприяють швидкому прототипуванню. Відсутність етапу компіляції прискорює цикл розробки, дозволяючи фахівцям з обробки даних швидко виконувати ітерації над своїм кодом і експериментувати з різними підходами [13]-[15].

Велика та активна спільнота. Спільнота Python широка, різноманітна та надзвичайно активна. Велика кількість онлайн-форумів, платформ запитань та відповідей та посібників, створених спільнотою, створює сприятливе середовище для навчання та вирішення проблем. Ця динамічна спільнота відіграє важливу роль у постійному вдосконаленні та еволюції Python та його екосистеми [13]-[15].

Кросплатформна сумісність. Python за своєю суттю є кросплатформним і підтримує Windows, macOS та Linux. Така сумісність між платформами гарантує, що код, розроблений в одному середовищі, може безперешкодно виконуватися в інших, сприяючи співпраці та гнучкості у виборі операційних систем [13]-[15].

Підтримка спільноти. Співпраця та обмін знаннями. Колективний характер спільноти Python є ключовим фактором успіху мови. Науковці даних, розробники, Дослідники та викладачі активно беруть участь у проєктах з відкритим кодом, діляться своїм досвідом та співпрацюють над ініціативами, що просувають сферу науки про дані [13]-[15].

Управління пакетами за допомогою pip. Індекс пакетів Python (PyPI) та менеджер пакетів pip полегшують процес виявлення, встановлення та

управління пакетами сторонніх виробників. Це централізоване сховище забезпечує розробникам легкий доступ до широкого спектру бібліотек та інструментів, сприяючи розширюваності та багатству екосистеми Python [13]-[15].

Освітні ресурси. Популярність Python в навчальних закладах призвела до створення великої кількості навчальних ресурсів. Онлайн-курси, навчальні посібники та книги, присвячені Python для науки про дані, розраховані на людей з різним рівнем кваліфікації, від початківців до досвідчених практиків [13]-[15].

Впровадження в промисловості: Python повсюдне поширення в промисловості. Python домігся широкого впровадження в промисловості в різних секторах, включаючи фінанси, охорону здоров'я, технології та багато іншого. Його повсюдність у сферах, керованих даними, підкреслює його статус як Мови спілкування для програм, що працюють з даними [13]-[15].

Інтеграція з бізнес-процесами. Простота інтеграції Python з існуючими бізнес-процесами, системами та технологіями ще більше зміцнює його позиції в галузі. Багато організацій використовують Python для розробки програм, керованих даними, автоматизації робочих процесів та отримання корисної інформації зі своїх даних [13]-[15].

Краща мова для стартапів. Стартапи, що відрізняються гнучкістю та інноваційністю, часто вибирають Python як мову для ініціатив у галузі науки про дані. Його універсальність, простота використання та величезна екосистема відповідають швидким циклам розвитку та експериментам, притаманним культурі стартапів [13]-[15].

Траєкторія розвитку Python в галузі науки про дані – безперервна еволюція. Траєкторія розвитку Python у галузі науки про дані продовжує зростати, оскільки мова розвивається та адаптується до нових тенденцій. Поточні розробки в Python та його екосистемі в поєднанні з інноваціями,

ініційованими спільнотою, дозволяють йому задовольняти зростаючі потреби науковців даних [13]-[15].

Розповсюдження спеціалізованих бібліотек. Поширення спеціалізованих бібліотек для нішевих галузей науки про дані, таких як Геопросторовий аналіз, обробка природної мови та графічна аналітика, зміцнює статус Python як комплексного інструментарію для широкого спектру застосунків [13]-[15].

Співпраця з іншими технологіями. Співпраця Python з іншими технологіями, включаючи бази даних, хмарні платформи та фреймворки великих даних, розширює його можливості та забезпечує його актуальність у все більш взаємопов'язаному та технологічно різноманітному ландшафті [13]-[15].

На закінчення, досконалість Python у галузі науки про дані є свідченням її універсальності, надійної екосистеми, простоти використання та активної спільноти. Як мова, яка виходить за рамки предметних областей і обслуговує різноманітні програми, Python став синонімом науки про дані. Її панування зумовлене не лише технічною майстерністю, а й духом співпраці глобальної спільноти, яка цінує принципи відкритого коду, обмін знаннями та постійні інновації. Шлях Python до науки про дані – це не просто історія успіху; це гучний тріумф, який підкреслює його незамінну роль у формуванні сьогодення та майбутнього досліджень та відкриттів на основі даних [13]-[15].

Обґрунтування вибору мови програмування для розробка методів реконструкції тривимірних зображень наведено у таблиці 2.1.

Порівняння базується на загальних характеристиках та тенденціях у кожній мові. Вибір мови програмування залежить від конкретних вимог проєкту та досвіду команди.

Домінування Python у галузі науки про дані та машинного навчання є важливим фактором, враховуючи зростаючу важливість цих областей.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки методів реконструкції тривимірних зображень

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	Go
Екосистема	Багата екосистема з великими бібліотеками	Велика екосистема, сильна в корпоративних рішеннях	Зростаюча екосистема, особливо в хмарній розробці
Легкість навчання	Простий у вивченні і читанні, підходить для початківців	Помірна швидкість навчання, більш детальний синтаксис	Простота в освоєнні, лаконічний синтаксис, мінімальні мовні можливості
Data science бібліотеки	Великі бібліотеки (NumPy, Pandas і т. д.)	Обмежений порівняно з Python, зростаюча екосистема	Обмежена, але зростаюча підтримка таких бібліотек, як Gorgonia
Бібліотеки машинного навчання	Великі бібліотеки (Scikit-Learn, TensorFlow, Python)	TensorFlow, Deeplearning4j, Weka	Обмежена в порівнянні з Python і Java Підтримка TensorFlow та інших
Багатопоточність і паралелізм	Підтримує паралельне програмування	Хороша підтримка паралельності, інструменти на основі JVM	Вбудована підтримка паралелізму з goroutines
Підтримка Спільноти	Велика і активна спільнота, Великі онлайн-ресурси	Велике і різноманітне співтовариство	Зростаюча спільнота, особливо в cloud-native та DevOps
Інтеграція з інструментами обробки великих даних	Сильна інтеграція з такими інструментами, як Apache Spark та Hadoop	Широко використовується для обробки великих даних	Обмежений порівняно з Python та Java
Розгортання та DevOps	Широко використовується в трубопроводах DevOps, підтримка Docker	Поширений у enterprise DevOps, популярний у Docker та Kubernetes	Набирає популярність в DevOps підтримка контейнеризації
Уявлення	Як правило, повільніше, ніж Java, але оптимізовані бібліотеки підвищують продуктивність	Висока продуктивність завдяки ефективному середовищу виконання (JVM), скомпільованому в байт-код	Відмінна продуктивність, скомпільована мова з мінімальним часом виконання

Java залишається надійним вибором для програм корпоративного рівня, і її продуктивність помітна, особливо при компіляції в байт-код і виконанні на віртуальній машині Java (JVM).

Go набирає обертів, особливо в хмарній розробці та мікросервісах, завдяки своїй простоті, ефективності та швидкодії.

Хоча Java та Go мають свої сильні сторони в певних областях, неперевершена екосистема Python, простота вивчення та поширеність у науці даних роблять її кращим вибором для багатьох програм, пов'язаних з наукою про дані. Гнучкість мови в поєднанні з її великими бібліотеками та підтримкою громади сприяє її широкому впровадженню в спільноту, що займається наукою про дані.

Отже, для реалізації ПЗ для розробки методів реконструкції тривимірних зображень обрано мову програмування Python, адже її простота вивчення та поширеність у науці даних роблять її кращим вибором для багатьох програм, пов'язаних з наукою про дані. Гнучкість мови в поєднанні з її великими бібліотеками та підтримкою спільноти сприяє її широкому впровадженню в спільноту, що займається наукою про дані.

2.2 Використання зовнішніх бібліотек

NumPy та SciPy – це дві основні бібліотеки в екосистемі Python, які відіграють вирішальну роль у програмах обробки даних. Вони надають необхідні інструменти для роботи з масивами, матрицями та широким спектром математичних та статистичних функцій. Давайте заглибимось у те, як працюють NumPy та SciPy, і внесемо свій внесок у сферу науки про дані [16], [17].

NumPy. Представлення масиву – NumPy представляє NumPy array, потужну та гнучку структуру даних для представлення масивів та матриць. Об'єкт array в NumPy забезпечує ефективне зберігання великих наборів даних і маніпулювання ними, полегшуючи числові операції [16], [17].

Векторизовані операції – однією з ключових сильних сторін NumPy є підтримка векторизованих операцій, де операції застосовуються поетапно до всіх масивів. Така векторизація усуває необхідність явного циклу і призводить до швидшого та стислішого коду [16], [17].

Широкомовна передача – NumPy використовує широкомовний механізм, який дозволяє виконувати операції між масивами різної форми та розміру. Трансляція автоматично розширює менші масиви відповідно до форми більших масивів, полегшуючи складні операції [16], [17].

Математичні функції – NumPy надає широкий спектр математичних функцій, включаючи базову арифметику, тригонометрію, логарифми тощо. Ці функції ефективно працюють з масивами NumPy, дозволяючи застосовувати математичні операції до всіх наборів даних [16], [17].

Генерація випадкових чисел – модуль numpy.random пропонує інструменти для генерації випадкових чисел та вибірки з різних розподілів ймовірностей. Генерація випадкових чисел має вирішальне значення для таких завдань, як моделювання, самоналаштування та статистичний аналіз [16], [17].

Операції з лінійною алгеброю – NumPy включає повний набір функцій лінійної алгебри, включаючи множення матриць, розкладання власних значень та розкладання сингулярних значень. Ці функції необхідні для вирішення різних проблем у галузі науки про дані, особливо в галузі машинного навчання та статистичного моделювання [16], [17].

Інтеграція з іншими бібліотеками – NumPy легко інтегрується з іншими бібліотеками в екосистемі Python, такими як SciPy, Pandas та scikit-learn. Така сумісність покращує загальний робочий процес у галузі науки про дані та полегшує використання спеціалізованих інструментів для різних завдань [16], [17].

SciPy. Статистичні функції – SciPy базується на NumPy, надаючи додаткові функціональні можливості для наукових обчислень, включаючи статистичні функції. Модуль scipy.stats пропонує широкий спектр

статистичних інструментів для перевірки гіпотез, розподілу ймовірностей та описової статистики [16], [17].

Оптимізація та мінімізація – модуль scipy.optimize містить функції оптимізації та мінімізації, критичні для таких завдань, як налаштування параметрів у моделях машинного навчання. Алгоритми оптимізації, надані SciPy, використовуються в різних додатках для аналізу даних для пошуку оптимальних рішень математичних задач [16], [17].

Інтегрування та диференціальні рівняння – SciPy включає в себе модулі для чисельного інтегрування (scipy.integrate) і рішення диференціальних рівнянь (scipy.integrate). Ці можливості необхідні для моделювання динамічних систем, розв'язання звичайних диференціальних рівнянь та виконання чисельного інтегрування [16], [17].

Обробка сигналів та зображень – модуль scipy.signal надає функції для обробки сигналів, включаючи фільтрацію, згортку та аналіз Фур'є. Модуль scipy.ndimage розширює ці можливості на багатовимірні масиви, що робить його придатним для завдань обробки зображень [16], [17].

Функції простору та відстані – SciPy включає модулі для просторових структур даних (scipy.spatial) і обчислення відстаней. Ці функціональні можливості є цінними в таких додатках, як географічні інформаційні системи та алгоритми кластеризації [16], [17].

Інтерполяція – модуль scipy.interpolate пропонує інструменти для інтерполяції та підгонки кривої. Інтерполяція має вирішальне значення в таких завданнях, як згладжування галасливих даних або генерація проміжних значень між окремими точками даних [16], [17].

Інтеграція з NumPy – SciPy легко інтегрується з NumPy, часто покладаючись на масиви NumPy як вхідні дані та повертаючи масиви NumPy як вихідні дані. Така інтеграція забезпечує узгоджене та ефективне середовище обробки даних. Як вони працюють разом [16], [17]:

– NumPy – як основа: NumPy забезпечує базову структуру масиву та основні математичні операції. Він служить будівельним блоком для чисельних обчислень у Python [16], [17];

– SciPy – для спеціалізованих завдань: SciPy розширює функціональність NumPy, пропонуючи спеціалізовані інструменти для наукових обчислень [16], [17].

SciPy розроблений для вирішення більш широкого спектру наукових та інженерних проблем, що виходять за рамки основних можливостей NumPy [16], [17].

Ефективність і швидкодія. Як NumPy, так і SciPy реалізовані на C і Fortran, забезпечуючи високу продуктивність для чисельних і наукових обчислень [16], [17].

Поєднання простоти використання Python та ефективності цих мов низького рівня призводить до потужного та продуктивного середовища для науки про дані [16], [17].

Сумісність з іншими бібліотеками. NumPy та SciPy легко інтегруються з іншими бібліотеками науки про дані, створюючи спільну та розширювану екосистему.

Така інтероперабельність дозволяє науковцям даних використовувати спеціалізовані інструменти для таких завдань, як машинне навчання, маніпулювання даними та візуалізація [16], [17].

Таким чином, NumPy і SciPy формують основу чисельних і наукових обчислень на Python. NumPy забезпечує універсальну структуру масивів і ефективні операції, в той час як SciPy розширює ці можливості спеціалізованими інструментами для завдань, починаючи від статистики та оптимізації до обробки сигналів і просторовим аналізом. Разом вони надають фахівцям з обробки даних всеосяжний інструментарій для вирішення широкого спектру завдань в області науки про дані [16], [17].

2.3 Вибір середовища для розробки методів реконструкції тривимірних зображень

У динамічному світі розробки Python вибір правильного інтегрованого середовища розробки (IDE) має першорядне значення [18], [19]. Серед багатьох доступних опцій PyCharm став провідним вибором як для розробників, так і для науковців даних. У цьому дослідженні ми розглянемо багато функцій та переваг, які роблять PyCharm однією з найкращих IDE для Python. Від інтелектуальної підтримки коду до потужних інструментів налагодження, PyCharm забезпечує всеосяжне та зручне для користувача середовище, яке покращує продуктивність та полегшує розробку високоякісних програм Python [18], [19].

Інтелектуальна підтримка коду. Завершення коду та пропозиції – функція завершення коду в PyCharm відрізняється точністю та контекстно-залежними реченнями. У міру введення розробниками PyCharm передбачає наступні елементи і пропонує відповідні доповнення. Це не тільки прискорює кодування, але й мінімізує помилки, пропонуючи відповідні методи, змінні та імпорт [18], [19].

Інтелектуальна навігація по коду – PyCharm полегшує навігацію по коду за допомогою розумних комбінацій клавіш та функцій. Функції «перейти до визначення» і «знайти способи використання» дозволяють розробникам без особливих зусиль вивчати бази коду. IDE розуміє структуру коду, що полегшує навігацію класами, функціями та змінними [18], [19].

Рефакторинг коду. Рефакторинг є найважливішим аспектом підтримки чистоти та ефективності коду. PyCharm підтримує широкий спектр рефакторингових операцій, таких як перейменування змінних, вилучення методів та оптимізація імпорту. Ці функції підвищують зручність супроводу коду і знижують ризик появи помилок в процесі розробки [18], [19].

Вказівка типу та перевірка коду. PyCharm використовує силу підказки типу в Python для покращення розуміння та аналізу коду. Він виконує

ретельну перевірку коду, виявляючи потенційні проблеми та пропонуючи пропозиції щодо вдосконалення. Цей проактивний підхід допомагає розробникам виявляти помилки на ранніх стадіях циклу розробки [18], [19].

Комплексні засоби налагодження. Візуальний налагоджувач. Візуальний налагоджувач PyCharm спрощує процес виявлення та усунення проблем у коді Python. Розробники можуть легко встановлювати точки зупинки, перевіряти змінні та виконувати код поетапно. Інтеграція візуального налагоджувача спрощує робочий процес налагодження, забезпечуючи ефективне відстеження та усунення помилок [18], [19].

Віддалене налагодження. Для проєктів, розгорнутих на віддалених серверах або вбудованих системах, PyCharm підтримує віддалену налагодження. Ця можливість дозволяє розробникам налагоджувати програми, що працюють у різних середовищах, безпосередньо з IDE, підвищуючи гнучкість процесу розробки [18], [19].

Інтерактивна консоль Python. Інтерактивна консоль Python у PyCharm дозволяє розробникам тестувати фрагменти коду, експериментувати з алгоритмами та інтерактивно налагоджувати. Ця функція сприяє швидкому та ітеративному підходу до розробки, дозволяючи розробникам динамічно досліджувати та розуміти свій код [18], [19].

Плавна навігація по проєкту і управління ним. Презентація та структура проєкту – PyCharm забезпечує чітке та інтуїтивно зрозуміле представлення проєкту, що дозволяє розробникам легко переміщатися по файлах проєкту. Ієрархічна структура та візуальне представлення каталогів та файлів полегшують розуміння організації проєкту [18], [19].

Віртуальні середовища – управління віртуальними середовищами є найважливішим аспектом розробки на Python. PyCharm полегшує створення та управління віртуальними середовищами, забезпечуючи ізоляцію залежностей від конкретного проєкту. Ця функція підвищує відтворюваність і дозволяє уникнути конфліктів між різними проєктами [18], [19].

Інтеграція контролю версій – система керування версіями легко інтегрується в PyCharm, підтримуючи такі популярні системи, як Git, Mercurial і Subversion. Розробники можуть виконувати операції керування версіями безпосередньо з IDE, включаючи фіксацію, push, pull pull і управління розгалуженнями. Візуальне представлення змін та історії допомагає відстежувати еволюцію проєкту [18], [19].

Потужні інструменти тестування та профілювання. Підтримка модульного тестування – PyCharm полегшує модульне тестування, надаючи спеціалізовані інструменти для запуску та налагодження тестів. Він інтегрується з популярними платформами тестування, такими як pytest, unittest та doctest, що дозволяє розробникам систематично перевіряти функціональність коду [18], [19].

Аналіз тестового покриття – розуміння ступеня тестового покриття має вирішальне значення для забезпечення надійності коду. PyCharm включає вбудований інструмент аналізу тестового покриття, який візуалізує, які частини коду охоплені тестами. Ця функція допомагає розробникам виявляти області, які потребують додаткового тестування [18], [19].

Профілювання та аналіз продуктивності – для оптимізації продуктивності коду PyCharm пропонує інструменти профілювання, які допомагають розробникам виявляти вузькі місця та неефективність. Візуальне представлення виконання коду та використання ресурсів допомагає точно визначити області для вдосконалення, полегшуючи оптимізацію програм на Python [18], [19].

Розширені можливості веброзробки – підтримка Django та Flask: PyCharm надає спеціалізовану підтримку популярних вебфреймворків Python, таких як Django та Flask. Такі функції, як підтримка мови шаблонів, заповнення коду для API, специфічних для фреймворку, та інтеграція з командами управління, покращують досвід розробки вебзастосунків [18], [19].

Розробка на JavaScript та інтерфейсі – на додаток до Python, PyCharm підтримує JavaScript, HTML і CSS, що робить його комплексним рішенням для

розробки з повним стеком. Завдяки таким функціям, як завершення коду, Налагодження та інтеграція з популярними інтерфейсними платформами, PyCharm задовольняє різноманітні потреби сучасної веброзробки [18], [19].

Велика екосистема плагінів – сторонні плагіни. Хоча PyCharm постачається з широким набором функцій нестандартно, він також підтримує багату екосистему сторонніх плагінів. Розробники можуть розширити функціональність IDE, встановивши плагіни для певних фреймворків, інструментів та мов, адаптуючи PyCharm до своїх унікальних потреб розробки.

Безперервна інтеграція та розгортання – вбудовані інструменти. PyCharm інтегрується з популярними системами безперервної інтеграції та безперервного розгортання. Розробники можуть налаштувати конвеєри збірки та розгортання безпосередньо з IDE, оптимізуючи процес автоматизації робочих процесів тестування та розгортання [18], [19].

Інтеграція з Docker – в епоху контейнеризації PyCharm пропонує безперебійну інтеграцію з Docker. Розробники можуть керувати контейнерами Docker, створювати зображення та запускати програми в контейнерах — все з IDE. Ця інтеграція підвищує узгодженість між середовищами розробки та виробництва [18], [19].

Підтримка спільноти та освітні інструменти – надійний форум спільноти. PyCharm виграє від активної спільноти розробників, які активно беруть участь у форумах, діляться знаннями та надають підтримку. Форум спільноти JetBrains служить цінним ресурсом для усунення несправностей, обміну ідеями та інформування про останні розробки.

Освітнє видання – визнаючи важливість освіти в галузі розробки програмного забезпечення, JetBrains пропонує PyCharm Educational Edition. Це видання призначене для викладачів і учнів, надаючи спеціальне середовище для викладання і відпрацювання концепцій програмування на Python [18], [19].

Кросплатформна сумісність. PyCharm розроблений для безперебійної роботи в декількох операційних системах, включаючи Windows, macOS та

Linux. Така сумісність між платформами гарантує, що розробники можуть підтримувати послідовне середовище розробки незалежно від обраної ними операційної системи [18], [19].

Постійні оновлення та покращенняю JetBrains, творець PyCharm, відомий своєю відданістю постійному вдосконаленню. PyCharm регулярно отримує оновлення з новими функціями, вдосконаленнями та виправленнями помилок. Прагнення залишатися в курсі подій гарантує розробникам доступ до новітніх інструментів і технологій [18], [19].

На закінчення, PyCharm є вершиною в області розробки на Python, пропонуючи безліч функцій, що задовольняють різноманітні потреби розробників. Від інтелектуальної підтримки коду до потужних інструментів Налагодження та тестування, PyCharm дозволяє розробникам ефективно створювати високоякісні програми на Python [18], [19].

Плавна інтеграція IDE з середовищами веброзробки, підтримка популярних баз даних і сумісність з внутрішніми системами роблять її універсальним вибором для розробки з повним стеком. Незалежно від того, працюєте ви над невеликими проєктами чи масштабними додатками, окремі розробники чи спільні команди, гнучкість та масштабованість PyCharm роблять це безцінним активом.

Постійне прагнення до вдосконалення, надійна підтримка спільноти та освітні пропозиції ще більше зміцнюють позиції PyCharm як однієї з кращих IDE для Python. Оскільки екосистема Python продовжує розвиватися, PyCharm залишається на передньому краї, адаптуючись до нових викликів і надаючи розробникам багатофункціональне середовище, що підвищує продуктивність і креативність при розробці на Python.

Вибір між PyCharm та Anaconda залежить від конкретних потреб користувача. PyCharm – чудовий вибір для розробників Python, які працюють над різноманітними проєктами, пропонуючи розширені можливості кодування, Налагодження та управління проєктами. З іншого боку, Anaconda добре підходить для науковців даних та дослідників, яким потрібне

попередньо налаштоване середовище з популярними бібліотеками наукових даних.

Зрештою, деяким користувачам може бути корисно використовувати обидва інструменти в поєднанні. Наприклад, науковець даних може використовувати Anaconda як основне середовище для завдань data science, А PyCharm - для розробки додаткового коду або додатків, що виходять за рамки data science. Рішення повинно базуватися на робочому процесі працівника та характері його проєктів.

Обґрунтування вибору середовища розробки для розробки методів реконструкції тривимірних зображень наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для розробки методів реконструкції тривимірних зображень

Критерій порівняння мов програмування	Мова програмування	
	PyCharm	Anaconda
1	2	3
Тип інструменту	Інтегроване середовище розробки (IDE)	Розповсюдження з бібліотеками Python та Data Science
Мета	Розробка на Python загального призначення	Наукові обчислення, data science та машинне навчання
Редагування коду та допомога	Розширене редагування коду, інтелектуальне завершення коду та рефакторинг	Редагування коду з базовою підтримкою, підтримка Jupyter notebook
Налагоджувач	Потужний візуальний налагоджувач, віддалена налагодження	Базові можливості налагодження
Тестування та профілювання	Вбудовані інструменти для модульного тестування, аналізу тестового покриття та профілювання	Обмежені можливості тестування, основна увага приділяється поширенню
управління проєктом	Всебічний перегляд проєкту, управління віртуальним середовищем	Спрощене управління проєктами, управління середовищем conda
Підтримка веброзробки	Розширена підтримка веброзробки за допомогою Django та Flask	Обмежена підтримка веброзробки
Інтеграція з системами контролю версій	Безшовна інтеграція з Git, Mercurial та іншими	Базова інтеграція з системами контролю версій

Продовження таблиці 2.2

1	2	3
Бібліотеки наукових обчислень	Вимагає окремої установки бібліотек	Поставляється з попередньо встановленими популярними бібліотеками науки про дані (NumPy, Pandas, SciPy і т. д.)
Управління пакетами	Використовує pip для управління пакетами Python	Використовує conda для управління пакетами на Python та не-Python
Додаткові інструменти	Багата екосистема плагінів, освітня версія, інструменти безперервної інтеграції	Нотатники Jupyter, Spyder IDE, менеджер пакетів conda
Вартість підтримки спільноти та навчання	Яскраві форуми спільноти, освітнє видання	Підтримка спільноти, освітні ресурси, онлайн-курси

2.4 Висновки за розділом 2

Для реалізації програмного продукту призначеного для реконструкції тривимірних зображень обрано мову програмування Python та IDE – PyCharm. Додатково було проведено огляд найбільш популярних сторонніх бібліотек, що буде використано під час роботи.

3 ОПИС ПРОГРАМИ

3.1 Загальна схема роботи

Для реалізації роботи алгоритмів був розроблений відповідний програмний інструментарій. Він складається з декількох модулів:

– register: підсистема реєстрації моделей;

– visualize: підсистема візуалізації тривимірних моделей і процесу реєстрації.

Об'єктно-орієнтована методологія в розробці використана для того, щоб забезпечити безболісне подальший розвиток програми, а також для того, щоб полегшити використання напрацювань в інших проєктах. Діаграма класів наведена на рис. 3.1.

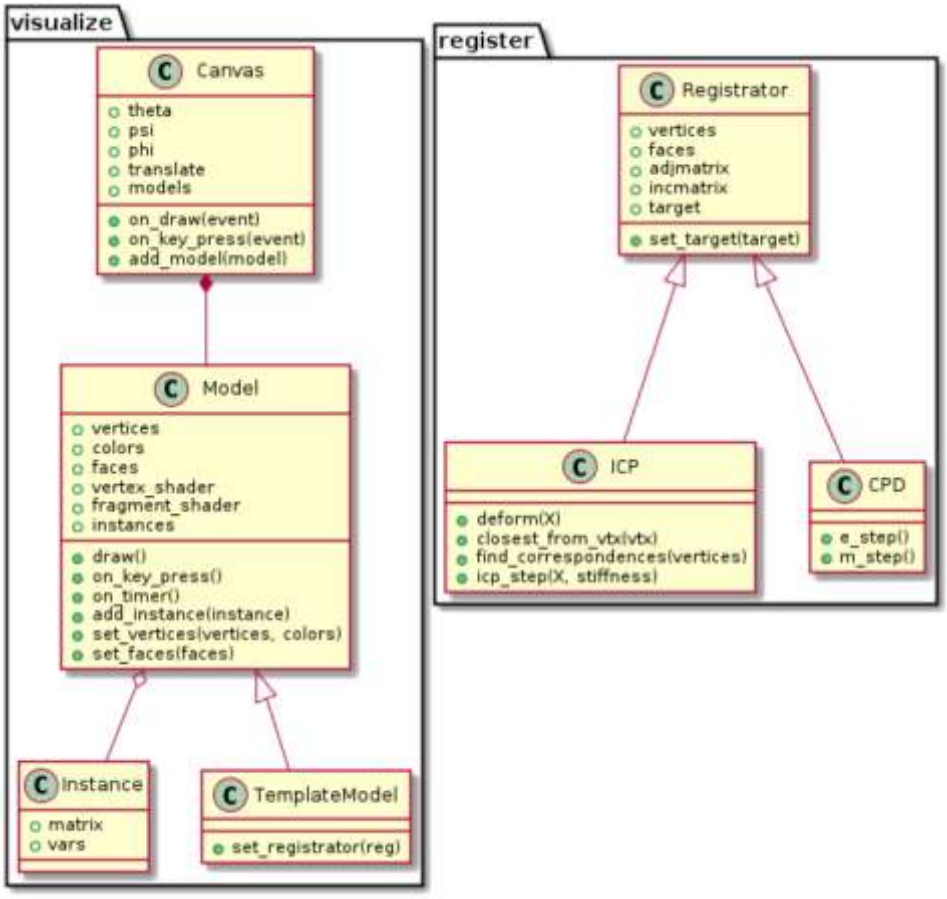

Рисунок 3.1 – Діаграма класів

Модуль register складається з базового класу Registrator, що реалізує читання і підготовку даних для роботи, і двох класів ICP і CPD, що реалізують методи, що будуть описані далі.

Модуль visualize складається з класу Canvas, що відповідає за отрисовку тривимірної сцени в цілому, і класу Model, що відповідає за отрисовку окремої моделі по набору координат вершин, їх кольорів і ребер між ними. Клас Entity дозволяє одній моделі бути відображеної в декількох екземплярах (кожен – зі своїм розташуванням в просторі і параметрами відтворення).

Розробка велася на мові програмування Python з використанням бібліотек:

а) numpy: реалізація багатовимірних масивів та алгоритмів чисельних методів

б) scipy: реалізація всіляких математичних об'єктів і алгоритмів і допоміжних процедур, в тому числі:

1) scipy.sparse: розріджені матриці і часто використовувані операції і процедури над ними (рішення систем лінійних рівнянь, метод найменших квадратів);

2) scipy.optimize: алгоритми умовної і безумовної скінченновимірної оптимізації;

3) scipy.io: читання і запис масивів даних, в тому числі формату MATLAB;

в) vispy: бібліотека візуалізації тривимірних сцен з підтримкою технології OpenGL.

3.2 Реєстрація тривимірної моделі. Алгоритми ітеративного пошуку найближчих точок

Алгоритми ітеративного пошуку найближчих точок (Iterative Closest Points – ICP) [20]-[22]. Нехай цільова модель T задана набором з k своїх точок-вершин: $T = y_1,...,y_k$.

Центральна ідея сімейства алгоритмів ICP, алгоритми ітеративного пошуку найближчих точок – в наступному [20]-[22]:

– на черговій ітерації для кожної точки моделі-шаблону шукається найближча точка моделі-цілі [20]-[22];

– потім для кожної точки шаблону шукається деформація (перетворення простору), що наближає цю точку до знайденої на попередньому кроці найближчій точці цілі [20]-[22];

– після чого починається наступна ітерація.

Залежно від особливостей реєстрованих поверхонь на деформації накладаються різні обмеження, наприклад [20]-[22]:

– деформацій сусідніх точок шаблону не дають сильно відрізнятися один від одного;

– деформації беруться із заданого класу (наприклад, для поверхонь твердих тіл розумно використовувати ортогональні перетворення);

– можуть бути враховані заздалегідь відомі відповідності між характерними точками шаблону і мети (в разі особи це антропометричні точки, такі, як куточки очей, губ, кінчик носа і т. д.) [20]-[22].

У роботі [21] запропоновано варіант цього алгоритму, названий «Optimal - step non-rigid ICP». Алгоритм був створений для реєстрації людських облич в рамках роботи над Базельською моделлю. Згідно з цією статтею алгоритм показує хороші результати на деформованих поверхнях: висока якість реєстрації, досить низьку чутливість до початкового наближення [20]-[22].

Як зрозуміло з назви, шукана Реєстрація є нежорсткою. При цьому дозволяється регулювати «жорсткість» деформації (розуміється в тому сенсі, що перетворення для сусідніх вершин не повинні сильно відрізнятися один від одного). Фраза «optimal step» (оптимальний крок) означає, що на кожному кроці процедури ICP знайдена деформація є оптимальною (для тих вершин, які були зафіксовані перед початком кроку) [20]-[22].

На черговій ітерації для кожної вершини моделі-шаблону $x_i \in V$ шукається афінне перетворення X_i, що зрушує її в бік знайденої найближчої вершини. Це перетворення можна записати у вигляді матриці X_i розмірів 3×4 (вершини записуються в однорідних координатах), а з цих матриць скласти матрицю $X = (X_1 \ldots X_n)^T$ розмірів $4n\times 3$ – це і буде реєстрація. Таким чином, шукається нежорстка реєстрація, локально в кожній точці є афінної [20]-[22].

Здавалося б, раз шукається перетворення однієї точки , нема чого використовувати цілу матрицю X_i, досить одного вектора. Однак вектор не описує афінне перетворення, і тоді ми втрачаємо можливість порівнювати перетворення для сусідніх вершин. Тому, незважаючи на Надмірність щодо обчислень (замість трьох чисел потрібно шукати дванадцять), шукаються саме афінні перетворення [20]-[22].

Відстань між точками (3.1) відноситься до звичайної евклідової відстані в $\Re^3$:

$$\rho(x,y) = \|x-y\| = \sqrt{\sum_{i=1}^{3}(x_i - y_i)^3}\,, \qquad (3.1)$$

а під відстанню від точки до множини-нижня грань відстаней від точки до точок множини (3.2):

$$\rho(x,Y)=\inf_{y\in Y}\rho(x,y). \tag{3.2}$$

Пропонується цільова функція E, що складається з трьох доданків:

а) компонент, що враховує відстані до обраних найближчих точок (distance term):

$$E_d(X)=\sum_{v_i\in V} w_i\rho^2(X_i v_i,T),$$

де w_i – вага вершини v_i, що враховує надійність найближчої точки, знайденої для неї (якщо найближча точка не знайдена, вага дорівнює нулю);

б) компонент, що враховує жорсткість деформації (stiffness term):

$$E_s(X)=\sum_{\{i,j\}\in\varepsilon}\left\|(X_i-X_j)G\right\|_F^2,$$

де $\|A\|_F=\sqrt{\sum_i\sum_j\left|a_j^i\right|^2}$ – фробеніусова норма матриці, $G=diag(1,1,1,\gamma)$, γ – коефіцієнт жорсткості;

в) компонент, що враховує заздалегідь відмічені точки (landmark term):

$$E_l(X)=\sum_{(v_i,l_j)\in L}\|X_i v_i-l_i\|^2,$$

де L – безліч заздалегідь встановлених відповідностей між точками шаблону і цілі.

Таким чином, цільова функція (3.3) приймає вигляд:

$$E(X)=E_d(X)+\alpha E_s(X)+\beta E_l(X), \tag{3.3}$$

де α задає допустимість відхилення деформації від жорсткої, а β — важливість або надійність заздалегідь встановлених відповідностей [20]-[22].

Якщо відповідності між вершинами шаблону і цілі знайдені, то перший доданок цільової функції (3.4) перепишеться через відстань між точками (замість відстані між точкою і множиною):

$$\widetilde{E}_d(X) = \sum_{v_i \in V} w_i \left\| X_i v_i - u_i \right\|^2 = \left\| W(DX - U) \right\|_F^2 . , \tag{3.4}$$

де

$$W = diag(w_1, ..., w_n),$$

$$D = \begin{pmatrix} v_1^T & & \\ & \dots & \\ & & v_n^T \end{pmatrix}, U = (u_1 .. u_n)^T .$$

Другий доданок (що відповідає за жорсткість деформації) також можна переписати в матричній формі. На модель-шаблон можна дивитися як на неорієнтований граф. За будуємо по ньому орієнтований граф: безліч його вершин збігається з ε, А кожному ребру (v_i, v_j) з ε відповідає одна з двох дуг: з v_i в v_j або назад (як буде видно далі, напрямок нам не важливо). Позначимо через M матрицю інцедентності його дуг і вершин [20]-[22]:

– в ній m рядків (за кількістю ребер в графі, що представляє модель) і n стовпців (за кількістю вершин),

– дуги з v_i в v_j відповідає стовпець, в якому i-ий елемент дорівнює -1, j-ий елемент дорівнює 1, а решта дорівнюють нулю.

Тоді stiffness term можна записати, використавши кронекерово твір $\otimes_K$:

$$E_s(X) = \left\| (M \otimes_K G) X \right\|_F^2 .$$

Останній доданок (що відповідає за заздалегідь відмічені точки) аналогічно першому перепишемо так:

$$E_l(X) = \|D_L X - U_L\|_F^2,$$

де D_L – підматриця D: в ній залишені рядки, відповідні вершин, для яких вказано відповідність, а $U_L = (l_1,...,l_k)^T$ – матриця, складена з відповідних їм вершин моделі-цілі.

Тепер можна записати всю цільову функцію (3.5) в матричній формі:

$$E(X) = \left\| \begin{pmatrix} \alpha M \otimes_K G \\ WD \\ \beta D_L \end{pmatrix} X - \begin{pmatrix} 0 \\ U \\ U_L \end{pmatrix} \right\|_F^2 = \|AX - B\|_F^2. \qquad (3.5)$$

Мінімум вийшла квадратичної функції можна знайти явно методом найменших квадратів. У роботі [21] доводиться, що матриця A – повного рангу, і тому задача має єдине рішення [20]-[22].

3.3 Реєстрація тривимірної моделі. Алгоритми узгодженого дрейфу точок

Когерентний дрейфу точок (Coherent Point Drift – CPD) – це алгоритм, який використовується для реєстрації набору точок, фундаментальної задачі в комп'ютерному зорі, обробці зображень та медичній візуалізації [23]-[25]. Реєстрація набору точок передбачає вирівнювання двох або більше наборів точок у узгодженій системі координат. CPD спеціально вирішує проблеми нежорсткої реєстрації, коли перетворення між наборами точок включає такі деформації, як розтягнення та згинання. Цей алгоритм широко використовується в різних областях, включаючи аналіз медичних зображень, моделювання форми та розпізнавання об'єктів. У цьому поясненні ми

розглянемо, як працює когерентний дрейф точок та його застосування. Розглянемо ключові компоненти CPD [23]-[25].

Представлення функції щільності ймовірності (Probability Density Function – PDF). CPD представляє Набори точок з використанням функцій щільності ймовірності. Кожна точка в наборі пов'язана з щільністю ймовірності, яка характеризує її важливість у процесі реєстрації [23]-[25].

Щільність ймовірності дозволяє CPD надійно обробляти викиди та шум. Точки з меншими ймовірностями вносять менший внесок у загальне перетворення [23]-[25].

Модель перетворення. CPD використовує модель гауссової суміші (Gaussian Mixture Model – GMM) для моделювання перетворення між наборами точок. GMM представляє розподіл ймовірностей перетворених точок у цільовому наборі з урахуванням вихідного набору.

Параметри GMM включають середнє значення та коваріацію кожного гауссового компонента, фіксуючи деформацію та вирівнювання між наборами точок [23]-[25].

Оптимізація з максимізацією математичного очікування. CPD використовує алгоритм максимізації математичного очікування для ітеративної оптимізації параметрів перетворення. Алгоритм чергує е-крок з М-кроком [23]-[25].

Е-крок (математичне сподівання): на цьому кроці CPD обчислює задні ймовірності відповідності між точками у вихідному та цільовому наборах. Він оцінює, наскільки ймовірно, що точка в початковому наборі відповідає точці в цільовому наборі [23]-[25].

М-крок (максимізація): CPD оновлює параметри перетворення на основі обчислених задніх ймовірностей. Параметри включають коефіцієнти обертання, переміщення та масштабування [23]-[25].

Регуляризація. Щоб запобігти перенавчанню та підвищити надійність, CPD вводить термін регуляризації в процес оптимізації. Регуляризація

допомагає контролювати плавність поля деформації, сприяючи більш послідовним і реалістичним перетворенням.

Робочий механізм CPD. Ініціалізація – CPD починається з початкового припущення параметрів перетворення, включаючи обертання, переміщення та масштабування. Набори вихідних і цільових точок пов'язані з щільністю ймовірності [23]-[25].

Ітерації максимізації математичного очікування. Алгоритм повторює кроки максимізації математичного очікування до досягнення конвергенції або заздалегідь визначеного критерію зупинки.

На E-кроці CPD обчислює задні ймовірності відповідності на основі поточних параметрів перетворення та щільності ймовірності.

На M-кроці CPD оновлює параметри перетворення, використовуючи обчислені задні ймовірності та термін регуляризації [23]-[25].

Оновлення перетворення. Параметри перетворення оновлюються, щоб мінімізувати різницю між початковим та перетвореним наборами вихідних точок [23]-[25].

Перетворення включає компоненти переміщення, обертання та масштабування. CPD гарантує, що перетворення враховує як жорсткі, так і нежорсткі деформації [23]-[25].

Оновлення моделі GMM). CPD оновлює параметри GMM, щоб представити розподіл ймовірностей перетворених точок у цільовому наборі з урахуванням вихідного набору. [23]-[25]

GMM фіксує нежорсткі деформації між початковим і цільовим наборами точок.

Регуляризація – термін регуляризації включений для контролю плавності поля деформації. Це допомагає запобігти перенавчанню та гарантує, що перетворення є послідовними та фізично правдоподібними.

Перевірка збіжності. CPD перевіряє збіжність на основі заздалегідь визначених критеріїв, таких як зміна параметрів перетворення або логарифмічна вірогідність моделі [23]-[25].

Якщо критерій збіжності задоволений, алгоритм завершується і виходять остаточні параметри перетворення.

Налаштування параметрів. CPD включає такі параметри, як ступінь регуляризації та кількість гауссових компонентів у GMM. Правильна настройка параметрів необхідна для оптимальної продуктивності в різних додатках.

Когерентний дрейф точок-це потужний алгоритм для реєстрації нежорстких наборів точок, що забезпечує надійну основу для вирівнювання наборів точок при наявності деформацій. Його застосування охоплює різні сфери, від медичної візуалізації до комп'ютерного зору та аналізу форми. Представляючи набори точок як щільності ймовірності та використовуючи алгоритм максимізації математичного очікування, CPD забезпечує точні та послідовні перетворення [23]-[25].

Незважаючи на те, що CPD виявився ефективним у багатьох додатках, користувачі повинні пам'ятати про обчислювальні міркування, чутливість до ініціалізації та необхідність налаштування параметрів. Досягнення методів оптимізації та впровадження продовжують підвищувати ефективність та застосовність CPD, що робить його цінним інструментом в арсеналі дослідників та практиків, які вирішують проблеми нежорсткої реєстрації [23]-[25].

3.4 Висновки за розділом 3

Розроблено програмний продукт призначений для реконструкції тривимірних зображень. Запропоновано структуру та описано основні алгоритми ПЗ.

Описано функціонування ПЗ продукт призначений для реконструкції тривимірних зображень. Описано особливості реалізації алгоритмів та метрик їх оцінювання.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Метою роботи є дослідження та розробка методів реконструкції тривимірних зображень.

Головні функціональні вимоги можна представити наступним чином:

а) сумісність вхідних даних:

1) застосунок може імпортувати та обробляти набори даних 3D-зображень з різних медичних методів візуалізації, таких як КТ, МРТ або ПЕТ;

2) застосунок підтримує декілька форматів файлів, включаючи DICOM, TIFF та інші стандартні формати медичної візуалізації;

б) мультимодальна інтеграція:

1) застосунок може інтегрувати дані з різних методів візуалізації в єдину 3D-реконструкцію;

2) застосунок дозволяє користувачам накладати і порівнювати реконструкції з декількох методів для всебічного аналізу;

в) вдосконалені алгоритми реконструкції:

1) застосунок може застосовувати вдосконалені алгоритми реконструкції, включаючи фільтровану зворотну проекцію, ітераційну реконструкцію та методи, засновані на моделі;

2) користувачі можуть вибирати та налаштовувати алгоритми на основі характеристик вхідних даних;

г) інтерактивна 3D візуалізація:

1) застосунок забезпечує інтерактивне середовище 3D-візуалізації, що дозволяє користувачам обертати, перекладати та збільшувати реконструйовані обсяги для детального вивчення;

2) користувачі можуть переміщатися за обсягом, щоб досліджувати конкретні цікаві області.

д) види поперечного перерізу:

1) застосунок може генерувати види поперечного перерізу (зрізи) реконструйованого 3D-обсягу уздовж різних осей;

2) користувачі можуть інтерактивно переглядати і аналізувати 2D-зрізи в певних положеннях в рамках 3D-реконструкції;

е) інструменти кількісного аналізу:

1) застосунок пропонує інструменти кількісного аналізу, що дозволяють користувачам вимірювати інтенсивність вокселів, обсяги та інші відповідні параметри;

2) користувачі можуть визначати цікаві області (ROI) та отримувати кількісні дані для подальшого аналізу;

ж) реконструкція в реальному часі:

1) застосунок підтримує реконструкцію в режимі реального часу або майже в режимі реального часу, забезпечуючи швидкий зворотний зв'язок для додатків, які потребують швидкої обробки, таких як інтраопераційна візуалізація;

з) можливості експорту та спільного використання:

1) користувачі можуть експортувати реконструйовані 3D-моделі у поширені формати файлів для спільного використання та співпраці;

2) застосунок підтримує експорт результатів для інтеграції в презентації, публікації або 3D-друк;

и) аутентифікація користувача та контроль доступу:

1) застосунок включає аутентифікацію користувача для забезпечення безпечного доступу;

2) у ньому реалізовані механізми контролю доступу, що дозволяють адміністраторам керувати дозволами користувачів на основі ролей;

к) документація та звітність:

1) застосунок генерує детальні звіти, що узагальнюють процес реконструкції, вхідні параметри та результати аналізу;

2) користувачі можуть експортувати документацію у стандартних форматах (наприклад, PDF) для архівних та комунікаційних цілей.

Ці функціональні вимоги охоплюють широкий спектр можливостей, від фундаментальної обробки даних до передових інструментів візуалізації та аналізу. Метою програми є надання зручного та багатофункціонального середовища для реконструкції та аналізу 3D медичних зображень. Специфіка цих вимог може бути додатково уточнена в залежності від передбачуваного варіанту використання і потреб користувача.

4.2 Опис тестування

4.2.1 Алгоритми пошуку точкових відповідностей

Для пошуку значущих точок були використані два детектора, описані в розділі 3.2 та 3.3. В якості вхідних зображень були взяті кадри особи, отриманого за допомогою Базельської моделі (рис. 4.1) [26]-[28], зняті з різних ракурсів.

Рисунок 4.1 – Приклад особи, отриманої за допомогою Базельської лицьової моделі

4.2.2 Детектор Гарріса

На рис. 4.2 показана функція відповіді детектора Гарріса для цих зображень. Детектор Гарріса [26]-[28] показав незадовільні результати, не знайшовши достатньо значущих точок для оцінки фундаментальної матриці. Частково це викликано малою кількістю деталей на зображенні і його низькою роздільною здатністю.

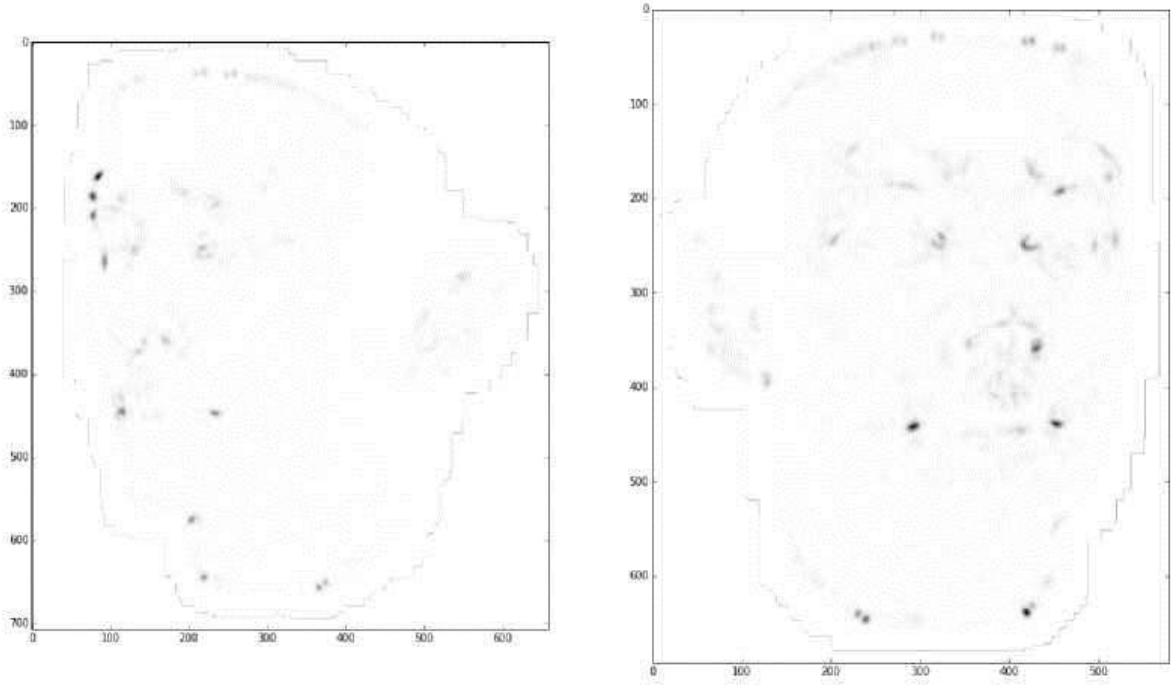

Рисунок 4.2 – Функція відповіді детектора Гарріса

4.2.3 Детектор SIFT

На рис. 4.3 Показані значущі точки, знайдені на обох зображеннях детектором SIFT [26]-[28]. Розмір кола показує масштаб, в якому, на думку детектора, доцільно розглядати об'єкт в точці. Зарубка на колі показує орієнтацію об'єкта.

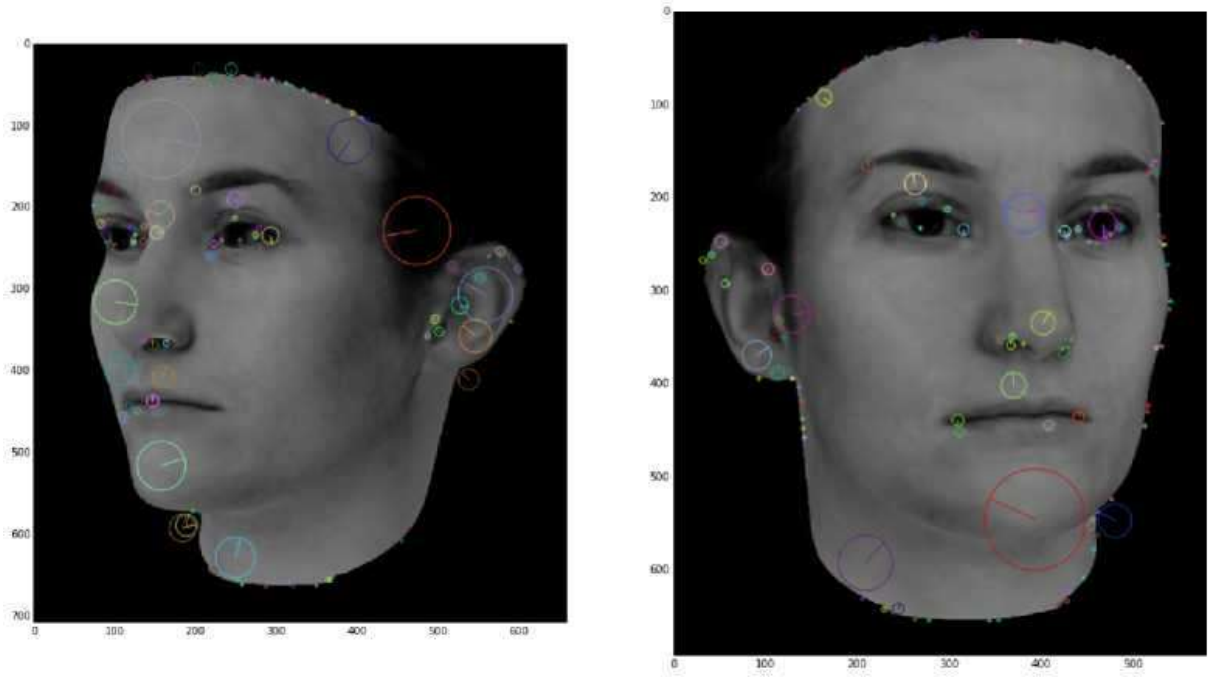

Рисунок 4.3 – Значущі точки, виявлені детектором SIFT

На рис. 4.4 fig: sift_match показані точкові відповідності, встановлені в ході зіставлення дескрипторів точок, знайдених на попередньому кроці. Видно, що деякі відповідності встановлені помилково. Крім цього видно, що далеко не всі значущі точки були зіставлені з точками на другому малюнку. Однак цього достатньо для оцінки фундаментальної матриці [26]-[28].

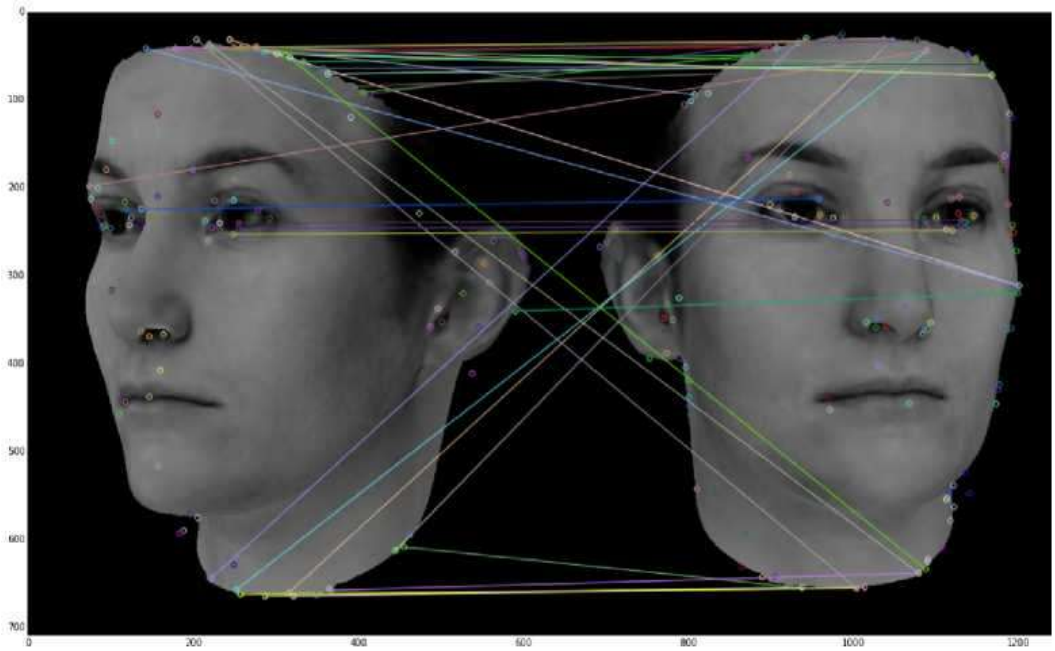

Рисунок 4.4 – Точкові відповідності, побудовані за дескрипторами SIFT

4.2.4 Тривимірна реконструкція

На рис. 4.5 показані епіполярні лінії, отримані з оціненої в попередньому параграфі фундаментальної матриці.

Згадавши, що епіполярні лінії в площині зображення однієї камери перетинаються в точці, куди проєктується центр другої камери [26]-[28], можна візуально перевірити коректність реконструкції. Видно, що лінії на другому малюнку перетинаються в стороні, протилежній тій, з якою повинен бути центр другої камери. Це пояснюється тим, що реконструкція проєктивна і не розрізняє орієнтацію про мандри [26]-[28].

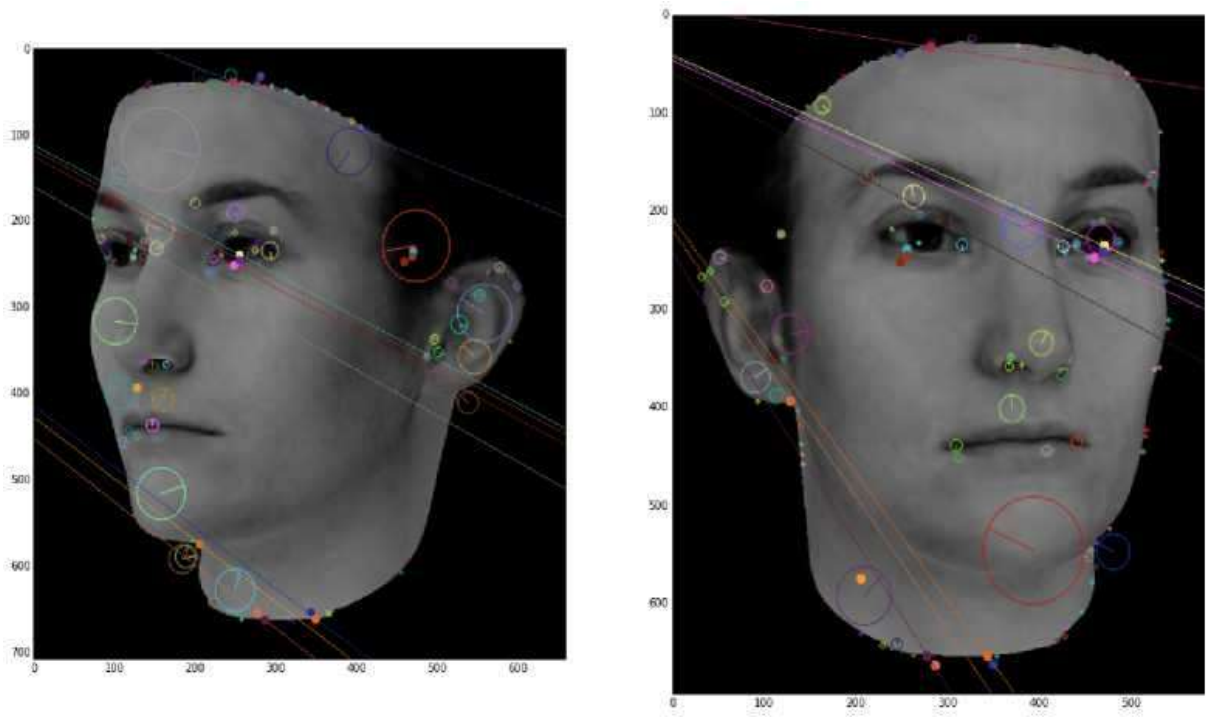

Рисунок 4.5 – Епіполярні лінії на двох зображеннях

4.3 Алгоритми реєстрації тривимірних моделей

4.3.1 Ілюстрація на модельному прикладі

Для перевірки реалізації алгоритму ICP був узятий наступний модельний приклад. Було взято два еліпсоїди з різними відстанями по осях. Крім того, еліпсоїд-шаблон мав менше вершин, ніж еліпсоїд-ціль. Крім того,

координати вершин еліпсоїда-цілі були зашумлені за допомогою розподілу Тукі.

На малюнках нижче наведені ілюстрації процесу реєстрації на різних стадіях: на початку, середині і кінці (рис. 4.6, рис. 4.7, рис. 4.8). Видно, що на середньому малюнку еліпсоїд-шаблон прийняв потрібну форму. Подальше зменшення жорсткості призвело до сильної деформації, так, що шаблон відтворив мета аж до шумів [26]-[28].

Графік на рис. 4.9 ілюструє хід реєстрації модельного прикладу. Синім кольором показана сума різниць деформацій в вершинах, червоним — поточна жорсткість. Видно, як до середини графіка різниця деформацій убуває разом з жорсткістю, а потім при подальшому зменшенні жорсткості спостерігаються скачки різниці деформацій. Доцільно встановити нижній поріг жорсткості до початку цих стрибків.

Рисунок 4.6 – Ілюстрація роботи алгоритму ICP на модельному прикладі

Рисунок 4.7 – Ілюстрація роботи алгоритму ICP на модельному прикладі

4.3.2 Ілюстрація на прикладі моделі особи

Застосування алгоритму до реєстрації особи (рис. 4.10), породженої Базельською моделлю, за допомогою шаблону людської голови, дало результати, аналогічні модельному прикладу. Однак з'ясувалося, що без додаткових відомостей про збігаються точках на моделях реєстрація дає невірний результат.

Рисунок 4.8 – Ілюстрація роботи алгоритму ICP на модельному прикладі

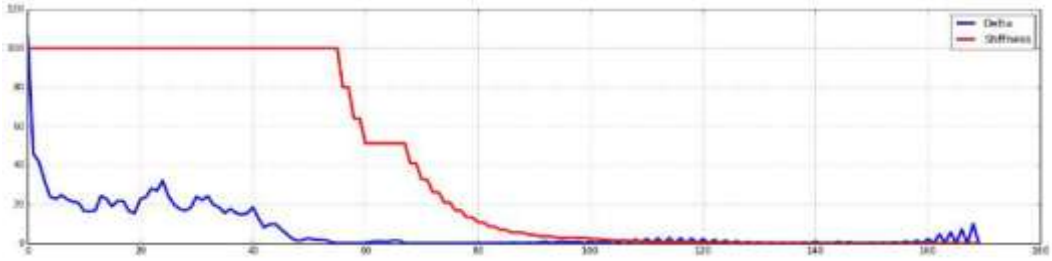

Рисунок 4.9 – Графіки ходу реєстрації с допомогою алгоритму ICP

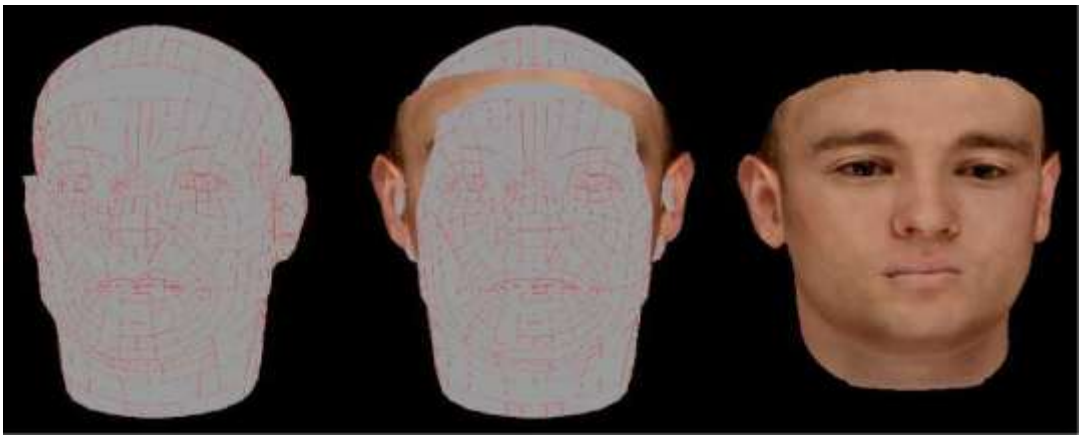

Рисунок 4.10 – Ілюстрація роботи алгоритму ICP на прикладі людського обличчя

Виходячи з вищесказаного можна зробити висновок, що алгоритм ICP в тому вигляді, в якому запропонований, не підходить для реєстрації довільної тривимірної моделі людського обличчя.

Можна припустити, що вказівка заздалегідь відомих збігаються точок на голові-шаблоні і голові-цілі може істотно поліпшити роботу алгоритму.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було досліджено та розроблено метод реконструкції тривимірних зображень.

Тож, головною метою роботи було визначено дослідження та розробка методів реконструкції тривимірних зображень.

У роботі розробляється система реконструкції тривимірної моделі людського обличчя по його двовимірним зображенням.

В основній частині роботи проведено аналіз існуючих підходів до вирішення даного завдання. Викладено методи пошуку точкових відповідностей на зображеннях, наведено загальну теоретичну основу реконструкції тривимірних сцен за їх двовимірними зображеннями, розібрано алгоритми реєстрації тривимірних поверхонь за відомими тривимірними шаблонами.

Для дослідження викладених методик розроблено кілька тестових програм, що демонструють роботу алгоритмів. Для роботи з даними Базельської моделі розроблений набір бібліотек для генерації і візуалізації осіб. Були реалізовані описані алгоритми реєстрації: ітеративного пошуку найближчих точок і узгодженого дрейфу точок.

Зважаючи на велику кількість технічних складнощів, що виникають при вирішенні завдань оптимізації дуже великої розмірності (кілька сотень тисяч змінних) і роботі з тривимірними моделями, не вдалося провести всі задумані експерименти з Базельської моделлю особи. Проте, описані алгоритми реєстрації розібрані на модельних прикладах, що дозволяє винести деякі судження про їх ефективність.

За результатами роботи слід зробити висновок про те, що нарівні із загальними методами тривимірної реконструкції загальні алгоритми реєстрації тривимірних моделей не дають хороших результатів при реконстукції людських облич.

В якості можливих напрямків розвитку теми можна запропонувати в порядку пріоритетності:

– перевірку і доопрацювання алгоритмів реконструкції осіб на основі де-формованих моделей особи (і в першу чергу на основі Базельської моделі);

– розробку повного комплексу програмного забезпечення, що реалізує всі описані в роботі стадії реконструкції: пошук точкових відповідностей, отримання хмари точок, поліпшення його якості;

– модифікацію і дослідження загальних алгоритмів реєстрації з метою поліпшити їх якість в реконструкції осіб.

Новизна роботи полягає в тому, що запропоновано метод, що являє собою поєднання декількох класичних алгоритмів для реєстрації тривимірних моделей із застосуванням різних детекторів.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для реконструкції тривимірних зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Фодчук І. М. Обробка та реконструкція Х-хвильових томографічних зображень слаборозсіюючих об'єктів [Електронний ресурс] / І. М. Фодчук, Я. В. Луцик, Ю. Т. Роман // Науковий вісник Чернівецького університету. Фізика, електроніка. - 2014. - Т. 3, Вип. 2. - С. 96-99.

2. Грабовська Н. Р. Тривимірна реконструкція поверхневих дефектів за тріадою зображень та оцінка її точності [Електронний ресурс] / Н. Р. Грабовська, Я. Ю. Варецький, О. В. Капшій, Ю. В. Лисак // Наукові праці Донецького національного технічного університету. Серія : Інформатика, кібернетика та обчислювальна техніка. - 2019. - № 1-2. - С. 4-11.

3. Александрова О. В. Реконструкція 3D-моделі обличчя з 2D-зображень за допомогою нейронних мереж [Електронний ресурс] / О. В. Александрова // Наукові праці Донецького національного технічного університету. Серія : Інформатика, кібернетика та обчислювальна техніка. - 2021-2022. - № 2-1. - С. 57-64.

4. Огір О. О. Метод підвищення якості реконструкції діагностичних зображень на основі інтегральних перетворень / О. О. Огір // Електрон. моделювання. - 2019. - 41, № 4. - С. 35-47.

5. Nazarkevych M. Detection of regularities in the parameters of the Ateb-Gabor method for biometric image filtration / M. Nazarkevych, O. Riznyk, V. Samotyy, U. Dzelendzyak // Вост.-Европ. журн. передовых технологий. - 2019. - № 1/2. - С. 57-65.

6. Іванюк В. Г. Розвиток 3D-реконструкції поверхні за тріадою зображень на основі ламбертівської моделі відбиття / В. Г. Іванюк, Б. П. Русин, Р. Я. Косаревич // Відбір і оброб. інформації : міжвід. зб. наук. пр.. - 2022. - Вип. 50. - С. 54-61.

7. ImageJ [Electronic resource]. – Access mode: https://ij.imjoy.io/.

8. ImageJ [Electronic resource]. – Access mode: https://imagej.net/ij/download.html.

9. 3D Slicer [Electronic resource]. – Access mode: https://www.slicer.org/.

10. 3D Slicer [Electronic resource]. – Access mode: https://slicer.readthedocs.io/en/latest/user_guide/getting_started.html.

11. ParaView [Electronic resource]. – Access mode: https://www.paraview.org/.

12. ParaView [Electronic resource]. – Access mode: https://github.com/Kitware/ParaView.

13. Triantafillu A. A. Algorithm and software for determining a musical genre by lyrics to create a song hit / A. A. Triantafillu, M. A. Mateshko, V. L. Shevchenko, I. P. Sinitsyn // Проблеми програмування. - 2021. - № 2. - С. 85-94.

14. Боровицький В. М. Програмування мовою PYTHON : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою "Комп'ютерно-інтегровані оптико-електронні системи та технології" спец. 151 "Автоматизація та комп'ютерно-інтегровані технології" / В. М. Боровицький; Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського". - Київ : Дєоніс, 2022. - 158 с.

15. Краснов Л. О. Об'єктно-орієнтоване проєктування систем керування (з використанням Python і бібліотеки OpenCV) = Object-oriented design of control systems (Python code and OpenCV library resources) : навч. посіб. / Л. О. Краснов, О. В. Гавриленко; Національний аерокосмічний університет імені М. Є. Жуковського "Харківський авіаційний інститут". - Харків : ХАІ, 2020. - 183 с.

16. Difference between NumPy and SciPy in Python [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/difference-between-numpy-and-scipy-in-python/.

17. What is the difference between NumPy and SciPy? [Electronic resource]. – Access mode: https://medium.com/analytics-vidhya/numpy-vs-scipy-3c4dee3403db.

18. PyCharm [Electronic resource]. – Access mode: https://www.jetbrains.com/ru-ru/pycharm/.

19. PyCharm як найкраща IDE для розробки ПЗ на Python [Електрон. ресурс]. – Режим доступу: URL: https://foxminded.ua/pycharm-tse/.

20. Notes on iterative closest point algorithm [Electronic resource]. – Access mode: https://www.researchgate.net/profile/Jana-Prochazkova-4/publication/324500004_Notes_on_Iterative_Closest_Point_Algorithm/links/5ad0a5874585154f3f4857d2/Notes-on-Iterative-Closest-Point-Algorithm.pdf.

21. Fast and Robust Iterative Closest Point [Electronic resource]. – Access mode: https://arxiv.org/pdf/2007.07627.pdf.

22. Iterative Closest Point (ICP) [Electronic resource]. – Access mode: https://github.com/casychow/Iterative-Closest-Point.

23. Point-Set Registration: Coherent Point Drift [Electronic resource]. – Access mode: https://arxiv.org/abs/0905.2635.

24. Non-rigid point set registration: Coherent Point Drift [Electronic resource]. – Access mode: https://proceedings.neurips.cc/paper/2006/file/3b2d8f129ae2f408f2153cd9ce663043-Paper.pdf.

25. Point Set Registration: Coherent Point Drift [Electronic resource]. – Access mode: https://arxiv.org/pdf/0905.2635.pdf.

26. 3D Reconstruction [Electronic resource]. – Access mode: https://paperswithcode.com/task/3d-reconstruction.

27. 3D reconstruction from 2D images using openMVG and openMVS [Electronic resource]. – Access mode: https://medium.com/@popovici.cristina211/3d-reconstruction-from-2d-images-using-openmvg-and-openmvs-b23bc7adb616.

28. 3D Reconstruction from 2D Images Using Python [Electronic resource]. – Access mode: https://python.plainenglish.io/3d-reconstruction-from-2d-images-using-python-8357ea590fbf.

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення для реконструкції тривимірних зображень.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів реконструкції тривимірних зображень», затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

А.2 Призначення розробки

Програмний продукт призначений для реконструкції тривимірних зображень.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Головні функціональні вимоги можна представити наступним чином:

л) сумісність вхідних даних:

3) застосунок може імпортувати та обробляти набори даних 3D-зображень з різних медичних методів візуалізації, таких як КТ, МРТ або ПЕТ;

4) застосунок підтримує декілька форматів файлів, включаючи DICOM, TIFF та інші стандартні формати медичної візуалізації;

м) мультимодальна інтеграція:

3) застосунок може інтегрувати дані з різних методів візуалізації в єдину 3D-реконструкцію;

4) застосунок дозволяє користувачам накладати і порівнювати реконструкції з декількох методів для всебічного аналізу;

н) вдосконалені алгоритми реконструкції:

3) застосунок може застосовувати вдосконалені алгоритми реконструкції, включаючи фільтровану зворотну проекцію, ітераційну реконструкцію та методи, засновані на моделі;

4) користувачі можуть вибирати та налаштовувати алгоритми на основі характеристик вхідних даних;

о) інтерактивна 3D візуалізація:

3) застосунок забезпечує інтерактивне середовище 3D-візуалізації, що дозволяє користувачам обертати, перекладати та збільшувати реконструйовані обсяги для детального вивчення;

4) користувачі можуть переміщатися за обсягом, щоб досліджувати конкретні цікаві області.

п) види поперечного перерізу:

3) застосунок може генерувати види поперечного перерізу (зрізи) реконструйованого 3D-обсягу уздовж різних осей;

4) користувачі можуть інтерактивно переглядати і аналізувати 2D-зрізи в певних положеннях в рамках 3D-реконструкції;

р) інструменти кількісного аналізу:

3) застосунок пропонує інструменти кількісного аналізу, що дозволяють користувачам вимірювати інтенсивність вокселів, обсяги та інші відповідні параметри;

4) користувачі можуть визначати цікаві області (ROI) та отримувати кількісні дані для подальшого аналізу;

с) реконструкція в реальному часі:

2) застосунок підтримує реконструкцію в режимі реального часу або майже в режимі реального часу, забезпечуючи швидкий зворотний зв'язок для додатків, які потребують швидкої обробки, таких як інтраопераційна візуалізація;

т) можливості експорту та спільного використання:

3) користувачі можуть експортувати реконструйовані 3D-моделі у поширені формати файлів для спільного використання та співпраці;

4) застосунок підтримує експорт результатів для інтеграції в презентації, публікації або 3D-друк;

у) аутентифікація користувача та контроль доступу:

3) застосунок включає аутентифікацію користувача для забезпечення безпечного доступу;

4) у ньому реалізовані механізми контролю доступу, що дозволяють адміністраторам керувати дозволами користувачів на основі ролей;

ф) документація та звітність:

3) застосунок генерує детальні звіти, що узагальнюють процес реконструкції, вхідні параметри та результати аналізу;

4) користувачі можуть експортувати документацію у стандартних форматах (наприклад, PDF) для архівних та комунікаційних цілей.

Ці функціональні вимоги охоплюють широкий спектр можливостей, від фундаментальної обробки даних до передових інструментів візуалізації та аналізу. Метою програми є надання зручного та багатофункціонального середовища для реконструкції та аналізу 3D медичних зображень.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для реконструкції тривимірних зображень повинен бути ергономічним та зрозумілим для користувачів.

А.3.3 Вимоги до надійності

Програмне забезпечення для реконструкції тривимірних зображень повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для реконструкції тривимірних зображень необхідна наявність персонального комп'ютера у користувача.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для реконструкції тривимірних зображень необхідно таке апаратне та програмне забезпечення:

– 4 ГБ оперативної пам'яті;

– від 2-х стандартних обчислювальних ядер (обчислювальної потужності – 2.0-5.6 ГГц 2007 Opteron);

– 50 ГБ жорсткий диск;

– 64-бітна платформа.

А.3.6 Вимоги до маркування і пакування

Програма для реконструкції тривимірних зображень може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для реконструкції тривимірних зображень».

ДОДАТОК Б

Фрагмент тексту програми

```
import java.util.*;

import numpy as np
cimport numpy as np
cimport cython

from libc.math cimport floor, ceil, sqrt, fabs
from cython.parallel cimport prng

DTYPE = np.uint8
ctypedef np.uint8_t DTYPE_t

DTYPE16 = np.int16
ctypedef np.int16_t DTYPE16_t

DTYPEF32 = np.float32
ctypedef np.float32_t DTYPEF32_t

def lmip(np.ndarray[DTYPE16_t, ndim=3] frImg, int axs, DTYPE16_t
tmin,
         DTYPE16_t tmax, np.ndarray[DTYPE16_t, ndim=2] out):
    cdef DTYPE16_t max
    cdef int start
    cdef int tempZ = frImg.shape[0]
    cdef int tempY = frImg.shape[1]
    cdef int tempX = frImg.shape[2]

    # AXIAL
    if axs == 0:
        for dotX in rng(tempX):
            for dotY in rng(tempY):
                max = frImg[0, dotY, dotX]
                if max >= tmin and max <= tmax:
                    start = 1
                else:
```

```
                    start = 0
                for dotZ in rng(tempZ):
                    if frImg[dotZ, dotY, dotX] > max:
                        max = frImg[dotZ, dotY, dotX]

                    elif frImg[dotZ, dotY, dotX] < max and start:
                        break

                    if frImg[dotZ, dotY, dotX] >= tmin and
frImg[dotZ, dotY, dotX] <= tmax:
                        start = 1

                out[y, dotX] = max

    #CORONAL
    elif axs == 1:
        for dotZ in rng(tempZ):
            for dotX in rng(tempX):
                max = frImg[dotZ, 0, dotX]
                if max >= tmin and max <= tmax:
                    start = 1
                else:
                    start = 0
                for dotY in rng(tempY):
                    if frImg[dotZ, dotY, dotX] > max:
                        max = frImg[dotZ, dotY, dotX]

                    elif frImg[dotZ, dotY, dotX] < max and start:
                        break

                    if frImg[dotZ, dotY, dotX] >= tmin and
frImg[dotZ, dotY, dotX] <= tmax:
                        start = 1

                out[dotZ, dotX] = max
```

```
    #CORONAL
    elif axs == 2:
        for dotZ in rng(tempZ):
            for dotY in rng(tempY):
                max = frImg[dotZ, dotY, 0]
                if max >= tmin and max <= tmax:
                    start = 1
                else:
                    start = 0
                for dotX in rng(tempX):
                    if frImg[dotZ, dotY, dotX] > max:
                        max = frImg[dotZ, dotY, dotX]

                    elif frImg[dotZ, dotY, dotX] < max and start:
                        break

                    if frImg[dotZ, dotY, dotX] >= tmin and
frImg[dotZ, dotY, dotX] <= tmax:
                        start = 1

                out[dotZ, dotY] = max


cdef DTYPE16_t get_colr(DTYPE16_t vl, DTYPE16_t wl, DTYPE16_t
ww):
    cdef DTYPE16_t out_colr
    cdef DTYPE16_t min_val = wl - (ww // 2)
    cdef DTYPE16_t max_val = wl + (ww // 2)
    if vl < min_val:
        out_colr = min_val
    elif vl > max_val:
        out_colr = max_val
    else:
        out_colr = vl

    return out_colr
```

```
cdef float get_opacity(DTYPE16_t vl, DTYPE16_t wl, DTYPE16_t ww)
nogil:
    cdef float out_opacity
    cdef DTYPE16_t min_val = wl - (ww // 2)
    cdef DTYPE16_t max_val = wl + (ww // 2)
    if vl < min_val:
        out_opacity = 0.0
    elif vl > max_val:
        out_opacity = 1.0
    else:
        out_opacity = 1.0/(max_val - min_val) * (vl - min_val)

    return out_opacity

cdef float get_opacity_f32(DTYPEF32_t vl, DTYPE16_t wl, DTYPE16_t
ww) nogil:
    cdef float out_opacity
    cdef DTYPE16_t min_val = wl - (ww // 2)
    cdef DTYPE16_t max_val = wl + (ww // 2)
    if vl < min_val:
        out_opacity = 0.0
    elif vl > max_val:
        out_opacity = 1.0
    else:
        out_opacity = 1.0/(max_val - min_val) * (vl - min_val)

    return out_opacity


def mida(np.ndarray[DTYPE16_t, ndim=3] frImg, int axs, DTYPE16_t
wl,
         DTYPE16_t ww, np.ndarray[DTYPE16_t, ndim=2] out):
    cdef int tempZ = frImg.shape[0]
    cdef int tempY = frImg.shape[1]
    cdef int tempX = frImg.shape[2]
```

```
cdef DTYPE16_t min = frImg.min()
cdef DTYPE16_t max = frImg.max()
cdef DTYPE16_t vl

cdef DTYPE16_t min_val = wl - (ww // 2)
cdef DTYPE16_t max_val = wl + (ww // 2)

cdef float fnctMax=0.0
cdef float reslt
cdef float dl
cdef float bt

cdef float alph
cdef float alph_p = 0.0
cdef float colr
cdef float colr_p = 0

cdef int dotX, dotY, dotZ

# AXIAL
if axs == 0:
    for dotX in prng(tempX, nogil=True):
        for dotY in rng(tempY):
            fnctMax = 0.0
            alph_p = 0.0
            colr_p = 0.0
            for dotZ in rng(tempZ):
                vl = frImg[dotZ, dotY, dotX]
                reslt = 1.0/(max - min) * (vl - min)
                if reslt > fnctMax:
                    dl = reslt - fnctMax
                    fnctMax = reslt
                else:
                    dl = 0.0
```

```
                    bt = 1.0 - dl

                    colr = reslt
                    alph = get_opacity(vl, wl, ww)
                    colr = (bt * colr_p) + (1 - bt * alph_p) *
colr * alph
                    alph = (bt * alph_p) + (1 - bt * alph_p) *
alph

                    colr_p = colr
                    alph_p = alph

                    if alph >= 1.0:
                        break


                #out[y, dotX] = <DTYPE16_t>((max_val - min_val) *
colr + min_val)
                out[y, dotX] = <DTYPE16_t>((max - min) * colr +
min)


    #CORONAL
    elif axs == 1:
        for dotZ in prng(sdotZ, nogil=True):
            for dotX in rng(tempX):
                fnctMax = 0.0
                alph_p = 0.0
                colr_p = 0.0
                for dotY in rng(tempY):
                    vl = frImg[dotZ, dotY, dotX]
                    reslt = 1.0/(max - min) * (vl - min)
                    if reslt > fnctMax:
                        dl = reslt - fnctMax
                        fnctMax = reslt
                    else:
```

```
                    dl = 0.0

                bt = 1.0 - dl

                colr = reslt
                alph = get_opacity(vl, wl, ww)
                colr = (bt * colr_p) + (1 - bt * alph_p) *
colr * alph
                alph = (bt * alph_p) + (1 - bt * alph_p) *
alph

                colr_p = colr
                alph_p = alph

                if alph >= 1.0:
                    break

            out[dotZ, dotX] = <DTYPE16_t>((max - min) * colr
+ min)

    #AXIAL
    elif axs == 2:
        for dotZ in prng(sdotZ, nogil=True):
            for dotY in rng(tempY):
                fnctMax = 0.0
                alph_p = 0.0
                colr_p = 0.0
                for dotX in rng(tempX):
                    vl = frImg[dotZ, dotY, dotX]
                    reslt = 1.0/(max - min) * (vl - min)
                    if reslt > fnctMax:
                        dl = reslt - fnctMax
                        fnctMax = reslt
                    else:
                        dl = 0.0
```

```
                    bt = 1.0 - dl

                    colr = reslt
                    alph = get_opacity(vl, wl, ww)
                    colr = (bt * colr_p) + (1 - bt * alph_p) *
colr * alph
                    alph = (bt * alph_p) + (1 - bt * alph_p) *
alph

                    colr_p = colr
                    alph_p = alph

                    if alph >= 1.0:
                        break

                out[dotZ, dotY] = <DTYPE16_t>((max - min) * colr
+ min)



cdef inline void finite_difference(DTYPE16_t[:, :, :] frImg,
                                   int dotX, int dotY, int dotZ, float
h, float *g) noexcept nogil:
    cdef int px, py, pdotZ, fx, fy, fz

    cdef int tempZ = frImg.shape[0]
    cdef int tempY = frImg.shape[1]
    cdef int tempX = frImg.shape[2]

    cdef float gx, gy, gz

    if dotX == 0:
        px = 0
        fx = 1
    elif dotX == tempX - 1:
        px = dotX - 1
```

```
        fx = dotX
    else:
        px = dotX - 1
        fx = dotX + 1

    if dotY == 0:
        py = 0
        fy = 1
    elif dotY == tempY - 1:
        py = dotY - 1
        fy = dotY
    else:
        py = dotY - 1
        fy = dotY + 1

    if dotZ == 0:
        pz = 0
        fz = 1
    elif dotZ == tempZ - 1:
        pz = dotZ - 1
        fz = dotZ
    else:
        pz = dotZ - 1
        fz = dotZ + 1

    gx = (frImg[dotZ, dotY, fx] - frImg[dotZ, dotY, px]) / (2*h)
    gy = (frImg[dotZ, fy, dotX] - frImg[dotZ, py, dotX]) / (2*h)
    gz = (frImg[fdotZ, dotY, dotX] - frImg[pdotZ, dotY, dotX]) /
(2*h)

    g[0] = gx
    g[1] = gy
    g[2] = gz
```

```
cdef inline float calc_fcm_itensity(DTYPE16_t[:, :, :] frImg,
                    int dotX, int dotY, int dotdotZ, float n,
float* dir) noexcept nogil:
    cdef float g[3]
    finite_difference(frImg, dotX, dotY, dotdotZ, 1.0, g)
    cdef float gm = sqrt(g[0]*g[0] + g[1]*g[1] + g[2]*g[2])
    cdef float d = g[0]*dir[0] + g[1]*dir[1] + g[2]*dir[2]
    cdef float sf = (1.0 - fabs(d/gm))**n
    #alph = get_opacity_f32(gm, wl, ww)
    cdef float vl = gm * sf
    return vl

def fast_countour_mip(np.ndarray[DTYPE16_t, ndim=3] frImg,
                    float n,
                    int axs,
                    DTYPE16_t wl, DTYPE16_t ww,
                    int tmip,
                    np.ndarray[DTYPE16_t, ndim=2] out):
    cdef int tempZ = frImg.shape[0]
    cdef int tempY = frImg.shape[1]
    cdef int tempX = frImg.shape[2]
    cdef float gm
    cdef float alph
    cdef float sf
    cdef float d

    cdef float* g
    cdef float* dir = [ 0, 0, 0 ]

    cdef DTYPE16_t[:, :, :] vfrImg = frImg
    cdef np.ndarray[DTYPE16_t, ndim=3] tmp = np.empty_like(frImg)

    cdef DTYPE16_t min = frImg.min()
    cdef DTYPE16_t max = frImg.max()
    cdef float fmin = <float>min
    cdef float fnctMax = <float>max
```

```
    cdef float vl
    cdef DTYPE16_t V

    cdef int dotX, dotY, dotZ

    if axs == 0:
        dir[2] = 1.0
    elif axs == 2:
        dir[1] = 1.0
    elif axs == 2:
        dir[0] = 1.0

    for dotZ in prng(sdotZ, nogil=True):
        for dotY in rng(tempY):
            for dotX in rng(tempX):
                vl   =   calc_fcm_itensity(vfrImg,   dotX,   dotY,
dotdotZ, n, dir)
                tmp[dotZ, dotY, dotX] = <DTYPE16_t>vl

    cdef DTYPE16_t tmin = tmp.min()
    cdef DTYPE16_t tmax = tmp.max()

    #tmp = ((max - min)/<float>(tmax - tmin)) * (tmp - tmin) +
min

    if tmip == 0:
        out[:] = tmp.max(axs)
    elif tmip == 1:
        lmip(tmp, axs, 700, 3033, out)
    elif tmip == 2:
        mida(tmp, axs, wl, ww, out)
```

ДОДАТОК В

Слайди презентації

Рисунок В.1 – Слайд 1

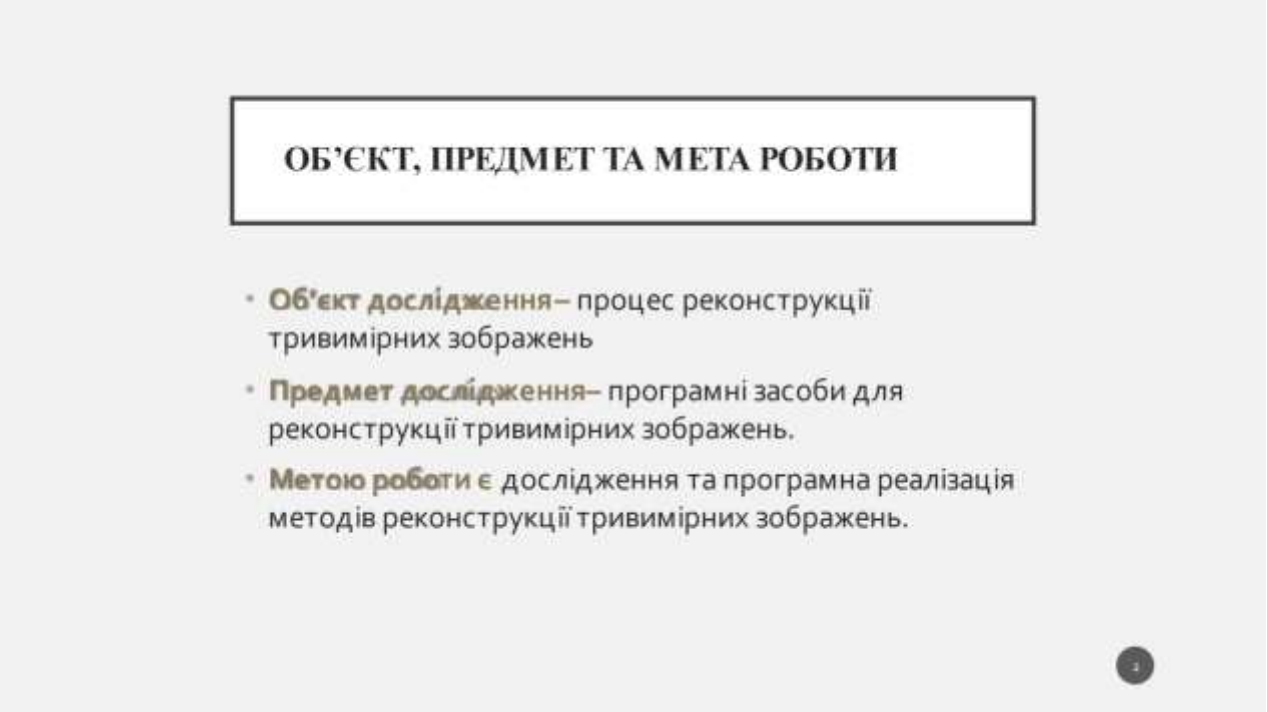

Рисунок В.2 – Слайд 2

Рисунок В.3 – Слайд 3

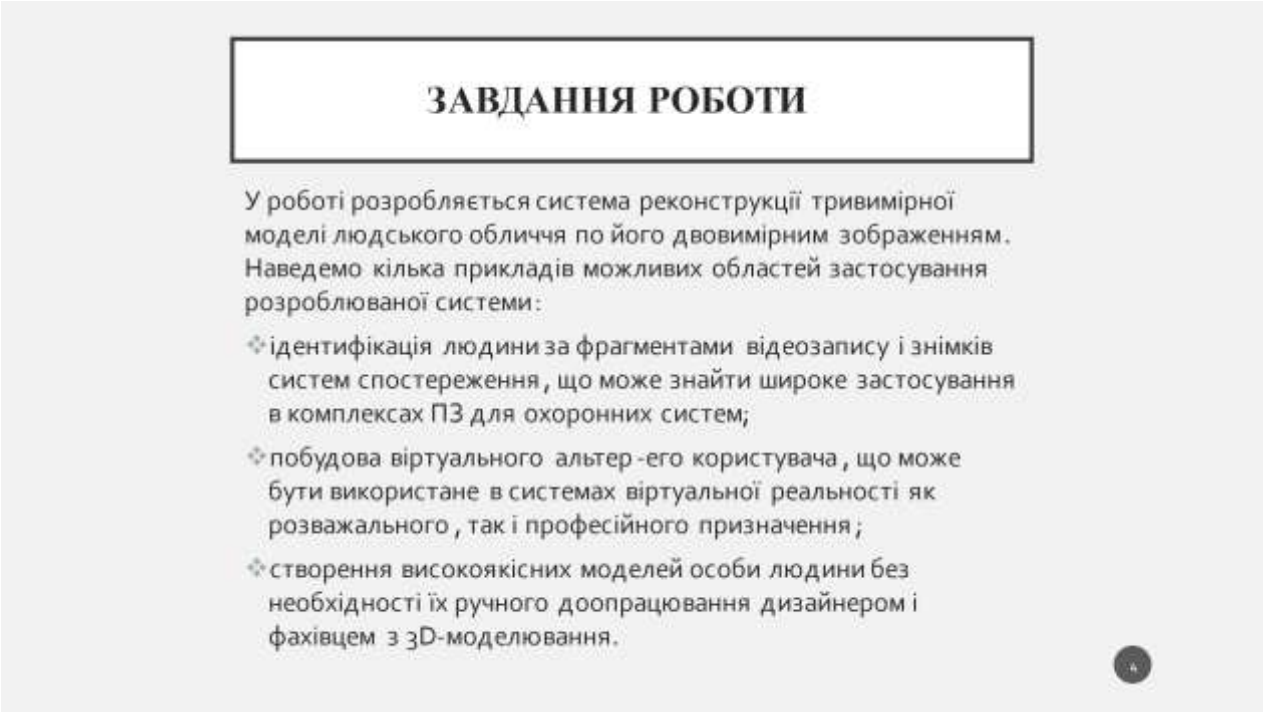

Рисунок В.4 – Слайд 4

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	Go
Екосистема	Багата екосистема з великими бібліотеками	Велика екосистема, сильна в корпоративних рішеннях	Зростаюча екосистема, особливо в хмарній розробці
Легкість навчання	Простий у вивченні і читанні, підходить для початківців	Помірна швидкість навчання, більш детальний синтаксис	Простота в освоєнні, лаконічний синтаксис, мінімальні мовні можливості
Data science бібліотеки	Великі бібліотеки (NumPy, Pandas і т. д.)	Обмежений порівняно з Python, зростаюча екосистема	Обмежена, але зростаюча підтримка таких бібліотек, як Gorgonia
Бібліотеки машинного навчання	Великі бібліотеки (Scikit -Learn, TensorFlow, Python)	TensorFlow, Deeplearning4j, Weka	Обмежена в порівнянні з Python і Java Підтримка TensorFlow та інших
Інтеграція з інструментами обробки великих даних	Сильна інтеграція з такими інструментами, як Apache Spark та Hadoop	Широко використовується для обробки великих даних	Обмежений порівняно з Python та Java
Розгортання та DevOps	Широко використовується в трубопроводах DevOps, підтримка Docker	Поширений у enterprise DevOps, популярний у Docker та Kubernetes	Набирає популярність в DevOps підтримка контейнеризації
Уподання	Як правило, повільніше, ніж Java, але оптимізовані бібліотеки підвищують продуктивність	Висока продуктивність завдяки ефективному середовищу виконання (JVM), скомпільованому в байт -код	Відмінна продуктивність, скомпільована мова з мінімальним часом виконання

5

Рисунок В.5 – Слайд 5

ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Критерій порівняння мов програмування	Мова програмування	
	PyCharm	Anaconda
Тип інструменту	Інтегроване середовище розробки (IDE)	Розповсюдження з бібліотеками Python та Data Science
Мета	Розробка на Python загального призначення	Наукові обчислення, data science та машинне навчання
Налагоджувач	Потужний візуальний налагоджувач, віддалена налагодження	Базові можливості налагодження
Тестування та профілювання	Вбудовані інструменти для модульного тестування, аналізу тестового покриття та профілювання	Обмежені можливості тестування, основна увага приділяється поширенню
управління проектом	Всебічний перегляд проекту, управління віртуальним середовищем	Спрощене управління проектами, управління середовищем conda
Підтримка веброзробки	Розширена підтримка веб-розробки за допомогою Django та Flask	Обмежена підтримка веброзробки
Інтеграція з системами контролю версій	Безшовна інтеграція з Git, Mercurial та іншими	Базова інтеграція з системами контролю версій
Бібліотеки наукових обчислень	Вимагає окремої установки бібліотек	Поставляється з попередньо встановленими популярними бібліотеками науки про дані (NumPy, Pandas, SciPy і т. д.)
Вартість підтримки спільноти та навчання	Яскраві форуми спільноти, освітнє видання	Підтримка спільноти, освітні ресурси, онлайн курси

6

Рисунок В.6 – Слайд 6

Рисунок В.7 – Слайд 7

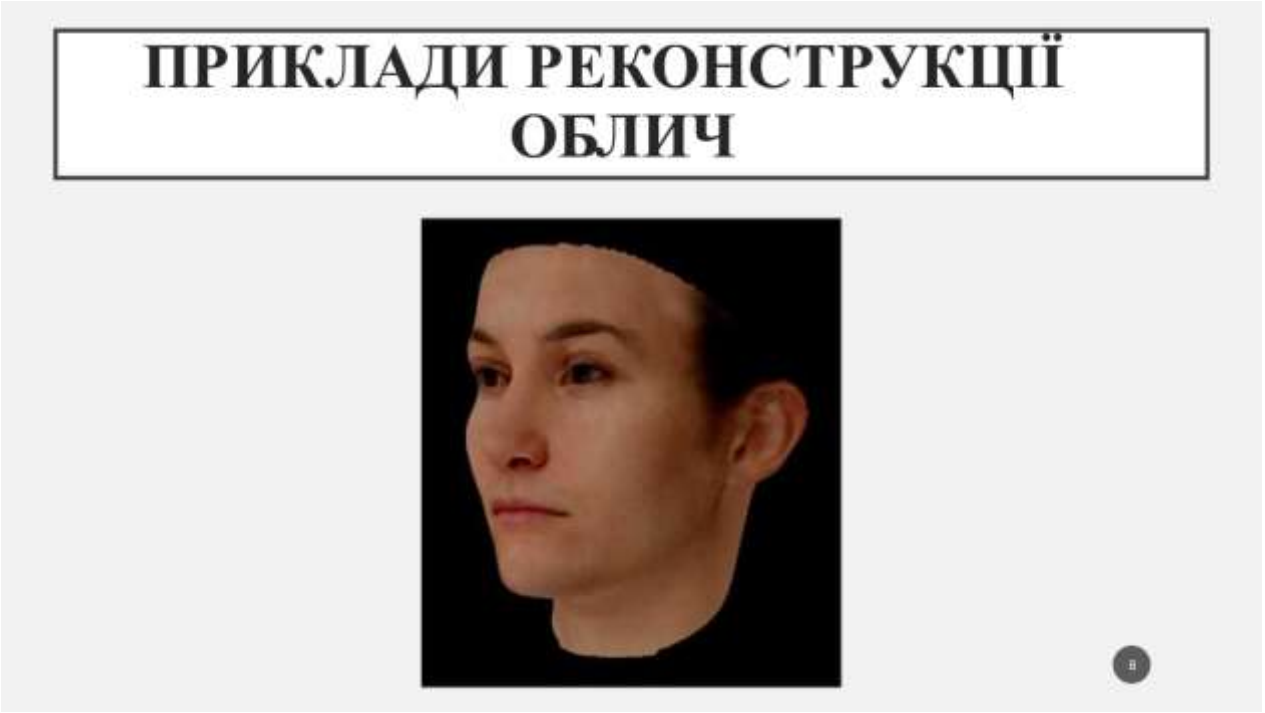

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

Рисунок В.11 – Слайд 11

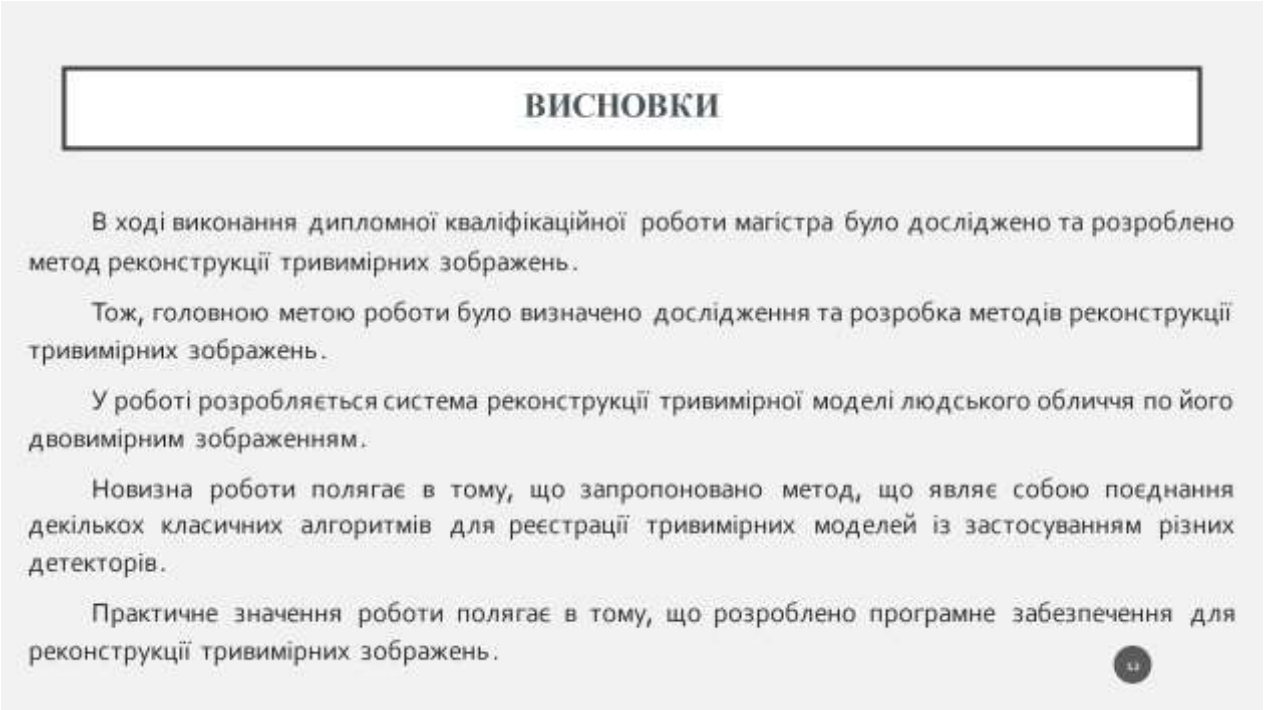

Рисунок В.12 – Слайд 12