НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2016 р.

Дипломна робота

__першого (бакалаврського)__ **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Алгоритми і методи створення опису поверхонь

Виконав (-ла): студент (-ка) __4__ курсу, групи __ДА-22___
(шифр групи)

Семироз Катерина Володимирівна _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к. т. н., доц., Харченко К.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економічна професор, д.е.н. Семенченко Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

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

Нормоконтроль ст. викладач Бритов О.А. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський))

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Семироз Катерині Володимирівні
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Алгоритми і методи створення опису поверхонь
керівник проекту (роботи) Харченко Костянтин Васильович, к. т. н, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 08.06.2016

3. Вихідні дані до проекту (роботи) _____

Набір точок
Перелік математичних методів
Параметри апроксимації

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Розглянути основні етапи побудови поверхні під час 3D сканування та перед 3D друком.
2. Сформулювати методи для порівняння точності моделювання поверхні.
3. Виконати загальний огляд алгоритмів та методів.
4. Розглянути програмну реалізацію алгоритмів та методів.
5. Виконати тестування програмних засобів.

6. Виконати програмну реалізацію методів порівняння точності моделювання поверхні алгоритмами і методами.
7. Виконати функціонально-вартісний аналіз програмного продукту.

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)
1. Мета, завдання та предмет дослідження роботи - плакат;
2. Огляд математичних методів та алгоритмів – плакат;
3. Огляд та порівняння програмних засобів, їх тестування – плакат;
4. Презентація

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Професор, д.е.н Семенченко Н.В.		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Дослідження алгоритмів та методів створення опису поверхні	05.02.2016	
4	Дослідження програмної реалізації методів та алгоритмів	10.03.2016	
5	Визначення методів для порівняння точності моделювання поверхні	15.03.2016	
6	Порівняння роботи програмних продуктів	30.03.2016	
7	Розробка програмної реалізації методів для порівняння точності моделювання поверхні	20.04.2016	
8	Порівняння результатів дослідження	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2016	

Студент _____ К.В.Семироз
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ К.В.Харченко
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

бакалаврської роботи Семироз Катерини Володимирівни

на тему: «Алгоритми і методи створення опису поверхні»

Дипломна робота присвячена дослідженню існуючих алгоритмів та методів для опису об'ємної поверхні. Метою роботи порівняння програмних реалізацій цих алгоритмів та методів, а також порівняння точності реконструювання поверхні.

У роботі розглянуто принцип роботи сімох алгоритмів та методів, також були досліджені бібліотеки та програми з реалізацією даних алгоритмів та методів. Проведено тестування бібліотек та програмних продуктів, після чого зроблено порівняння точності реконструювання поверхні алгоритмами і методами.

Загальний обсяг роботи: 77 сторінки, 37 малюнків, 7 таблиць, 14 посилань.

Перелік ключових слів: 3D сканування, реконструкція поверхні, набір точок, порівння, точність, алгоритми обчислювальної геометрії, алгоритми і методи з використанням неявних функцій.

АННОТАЦИЯ

бакалаврской дипломной работы Семироз Катерины Владимировны

на тему:«Методы и алгоритмы создания описания поверхности»

Дипломная работа посвящена исследованию существующих алгоритмов и методов для описания обьемной поверхности. Целью работы есть сравнения програмных реализаций этих алгоритмов и методов, а так же сравнение точности реконструкции поверхности.

В работе были рассмотрены принципы работы семи алгоритмов и методов, так же были исследованы библиотеки и програмы с реализацией даных алгоритмов и методов. Проведено тестирование библиотек и програмных продуктов, после чего было сделано сравнение точности реконструкции поверхности алгоритмами и методами.

Общий объем работы: 77 страница, 37 рисунков, 7 таблиц, 14 ссылок.

Перечень ключевых слов: 3D сканирование, реконстукция поверхности, облако точек, сравнение, точность, алгоритмы вычислительной геометрии, алгоритмы и методы с использованием неявных функций.

ANNOTATION

for the bachelors work of Semyroz Kateryna

«The algorithms and methods for surface description»

Diploma thesis is devoted to research existing algorithms and methods for 3D surface description. The aim is to compare the software implementation of these algorithms and methods and to compare the accuracy of surface reconstruction.

The paper provides an overview of seven principles of algorithms and methods. Other then that I explored the libraries and software with implementation of current algorithms and methods. Libraries and software had been tested, after that the accuracy of surface reconstruction by algorithms and methods was compared.

Total of 77 pages, 37 pictures, 7 tables, 14 references.

Keyword list: 3D scanning, surface reconstruction, point cloud, comparsion, accuracy, computational geometry algorithms, implicit algorithms and methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ	Програмне забезпечення
ПП	Програмний продукт
MPU	Multilevel Partition of Unity Implicits
SPSR	Screened Poisson Surface Reconstruction
CGAL	Computational Geometry Algorithms Library

ВСТУП

Метою цієї роботи є аналіз існуючих методів для опису поверхонь та їх порівняння між собою, а також створення своєї власної програми з використанням одного з методів та порівняння її з вже існуючими.

Предметною областю є опис поверхонь. Поверхня – це межа, що відокремлює геометричне тіло від іншого тіла чи від зовнішнього простору. Термін «поверхня» має дуже широке застосування в комп’ютерній графіці, фізиці, під час 3D-друку, а також в багатьох інших областях, які взаємодіють з поверхнями фізичних об’єктів.

Актуальність роботи: з появою 3D-принтерів використання опису поверхонь набуло нового розвитку, адже саме з допомогою об'ємного зображення предмету і вдбувається друк в 3D-принтерах. Саме для того, щоб друк відбувався якомога якснішe та точніше, математики та програмісти працюють разом для створення нових, використання вже існуючих різноманітних алгоритмів опису поверхонь та втілення їх в програми, що побудовані спеціально з врахуванням особливості 3D-друку. Крім цього, паралельно з 3D-друком має розвиток і область 3D-сканування, головною задачею якої є зісканувати потрібні предмети з потрібним рівнем деталізації.

В деяких випадках, точність сканування та друку мають вирішальну дію. Наприклад, в медичних цілях вже зараз друкують хрящі, шкіру та деякі органи[1]. В майбутньому науковці та медики планують розширити та вдосконалити можливості 3D-друку, а саме, щоб мати можливість надрукувати більш широкий спектр органів, навіть серце та найменші кровеносні судини, тому потреба в більш точному описі поверхонь буде тільки зростати.

Існуючі рішення: в даний момент існують близько семи основних алгоритмів для створення опису поверхні. Вони поділяються на алгоритми обчислювальної геометрії, а також на алгоритми з використанням неявних функцій.

Програмна реалізація алгоритмів представлена окремими реалізаціями кожної з функцій, а також існує відкрита бібліотека PCL (Point Cloud Library), яку можна використовувати для побудови об’ємних зображень з набору точок.

Найпопулярнішими алгоритмами, які виконують побудову об'ємного об'єкту з набору точок є такі алгоритми, як:

- Триангуляція Делоне,
- Ball Pivoting,
- Marching Cubes,
- Poisson,
- Hoppe,
- MPU.

Усі вони будуть в подальшому розглянуті в даній роботі. Крім цього, будуть описані можливі шляхи покращення існуючих рішень.

1. ОСНОВНІ ЕТАПИ ПОБУДОВИ ПОВЕРХНІ ПІД ЧАС 3D-СКАНУВАННЯ ТА ПЕРЕД 3D-ДРУКОМ

1.1. Основні поняття та етапи роботи

1.1.1. 3D-сканування

3D-сканування – це технологія, під час якої аналізується об’єкт або середовище реального світу та відбувається збір данних щодо того, як цей об’єкт або середовище виглядає, яку має форму, та інколи який колір. Зазвичай, ці дані потім використовуються для створення об’ємної моделі об’єкта або середовища. 3D-сканування може бути корисним в ряді задач, таких як проектування пристроїв, запасних частин при умові відсутності оригінальної комп’ютерної документації на пристрій, а також при необхідності побудови в цифровому вигляді поверхонь складної форми, в тому числі маються на увазі певні художні форми та зліпки.

3D-сканер – це пристрій, що використовується для 3D-сканування. Вони бувають двох типів: контактні та безконтактні. Безконтактні в свою чергу поділяються на активні та пасивні.

Контактні сканери побудовані за принципом обвода об’єкта спеціальним високочутливим щупом, за допомогою якого в комп'ютер передаються тривимірні координати об’єкта, що сканується.

Активні безконтактні сканери випромінюють світло або навіть один з видів радіації для того, щоб відслідкувати відбивання проміння від самого об’єкта. Під час сканування використовуються такі типи випромінювання, як світло, ультразвук та ренгеновське випломінювання.

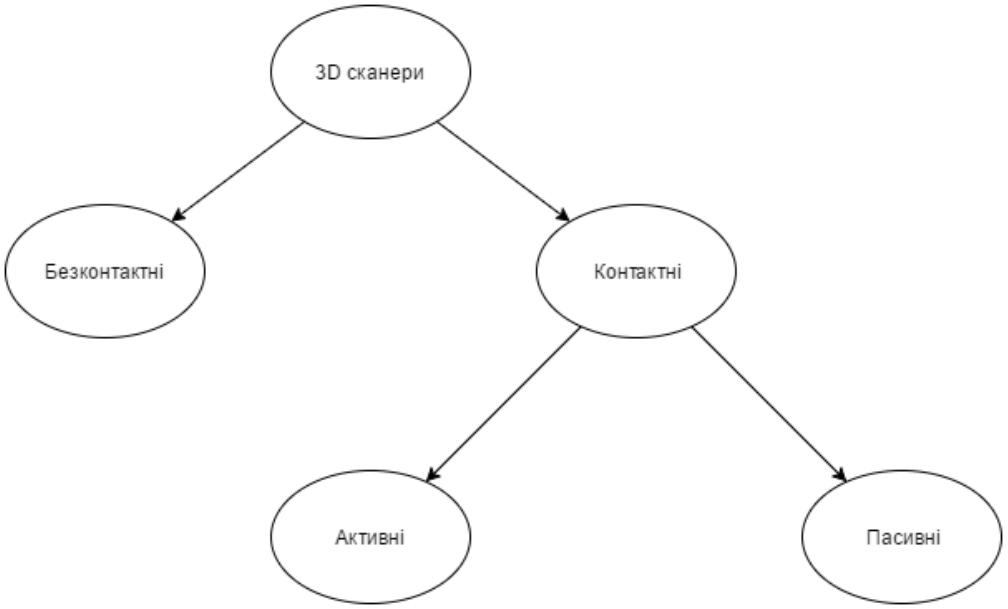

Рисунок 1.1 - Класифікація 3D-сканерів

Пасивні безконтактні 3D-сканери не випромінюють нічого, їх принцип роботи будується на тому, що вони приймають зовнішнє світло, яке відбивається від об’єкта, що сканується. Фактично, пасивні безконтактні 3D-сканери – це фото- або відео- камера, але з деякими відмінностями. Різниця полягає в тому, що на відміну від відеокамери, яка видає кольорові зображення, сканер використовує отримані зображення для розрахунку 3D-даних в межах своєї зони видимості (координати точок поверхні і їх віддаленість від сканера).

Після сканування предмети представлені у вигляді набору точок (набір XYZ-координат виміряних точок об'єкта) або у вигляді полігональної моделі (тріангульований набір точок, тобто уявлення моделі предмета у вигляді набору трикутників).

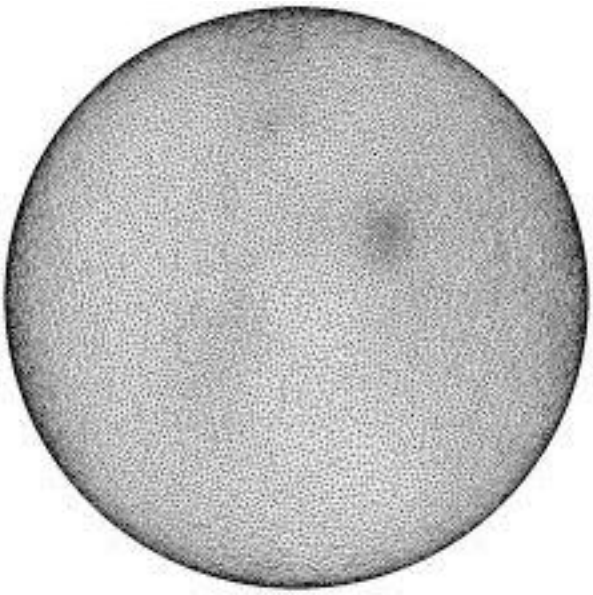

Рисунок 1.2 - Зображення сфери, що представлене набором точок.

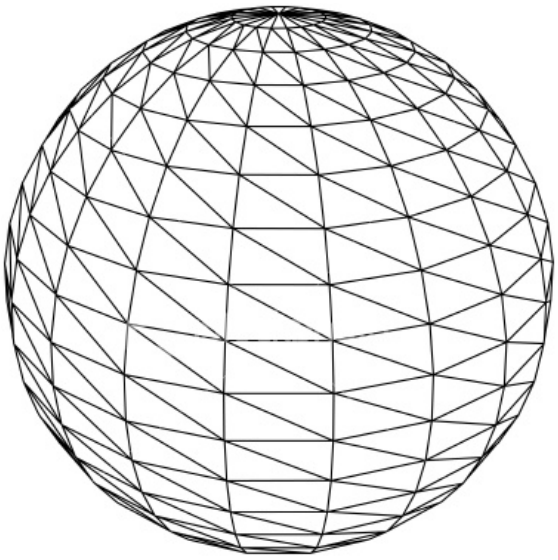

Рисунок 1.3 - Полігональне зображення сфери

В даній дипломній роботі розглядаються саме методи побудови поверхні, які використовують 3D-сканери. Основними характеристиками 3D-сканування можна назвати точніть та роздільну здатність.

Точність - характеристика відповідності розмірів отриманої 3D-моделірозмірами фізичного об'єкта сканування.

Роздільна здатність - розмір мінімального полігону (відстані між точками) отриманої моделі. Тобто дискретизація, з якої оцифровується об'єкт.

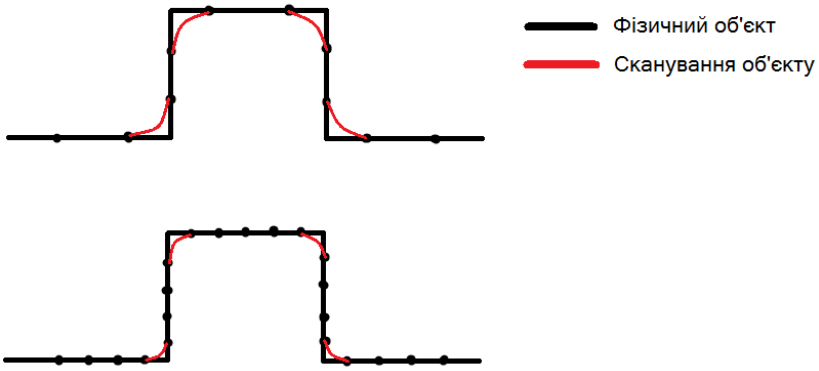

Рисунок 1.4 - Сканування об’єктів з різною точністю

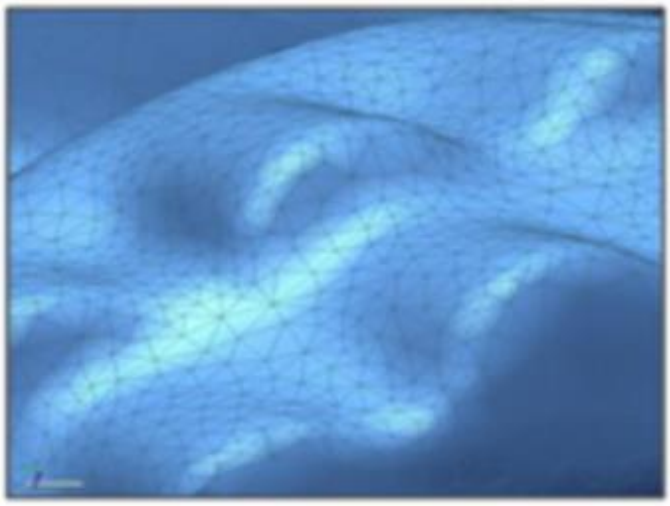

Рисунок 1.5 - Низька роздільна здатність

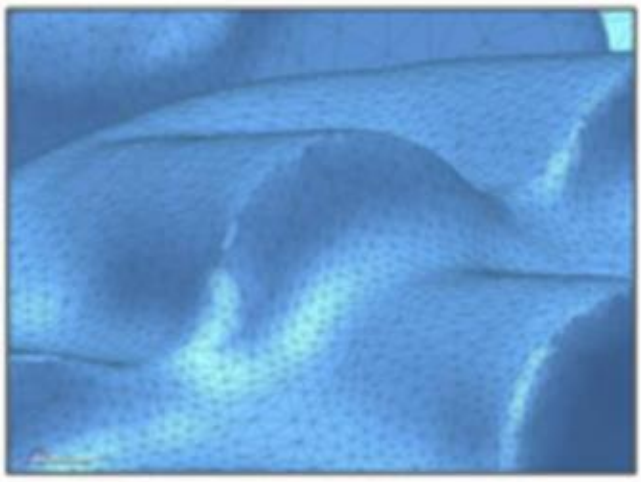

Рисунок 1.6 - Висока роздільна здатність

1.1.2. 3D-друк

3D-друк (або адитивне виробництво) – це сукупність різних процесів, які використовуються для створення трьовимірного об’єкту.

3D-принтер – це пристрій для 3D-друку.

3D-друк відбувається наступним чином:

1. Перш за все, потрібно мати відповідну об’ємну модель предмета на комп’ютері. Способи отримання цієї моделі дуже різні:
    - таку модель можна сконструювати самому, використовуючи відповідні програми,
    - завантажити з інтернет-джерел,
    - отримати шляхом 3D-сканування.
2. Після отримання потрібної об’ємної моделі, її потрібно обробити спеціальною програмою, яка «розрізає» модель на шари, кожен з шарів перетворюється в спеціальний машинний код (G-code).

    G-code – це спеціальна мова програмування, з допомогою якої 3D-принтеру надається інформація щодо того яким чином, звідки і куди буде рухатися 3D-принтер задля того, щоб надрукувати заданий об’єкт. Основні дані, які описуються в G-коді - це:
    - швидкість з якою має рухатися 3D-принтер,
    - координати початку руху кожної дії (координати x, y, z),
    - координати кінця руху кожної дії (координати x, y, z),
    - швидкість подачі матеріалу, з якого відбувається друк.
3. Після переводу об’ємного зображення в G-код, G-код передається в 3D-принтер, який зчитуючи кожен рядок коду друкує об'єкт. Передача може здійснюватися як з допомогою зйомного носія пам'яті (це може бути, наприклад, SD-картка), так і напряму з комп'ютера до 3D-принтера. Другий спосіб складніший, адже під час передачі даних напряму з комп’ютера до принтера, комп’ютер має бути постійно робочим і передача

данних не має обриватися, адже в цьому випадку 3D-друк припиняється і немає можливості запустити його знову з того самого місця. Оскільки предмети друкуються зазвичай від години і до доби часу, то часто буває проблематично тримати комп’ютер або ноутбук ввімкненим весь цей час.

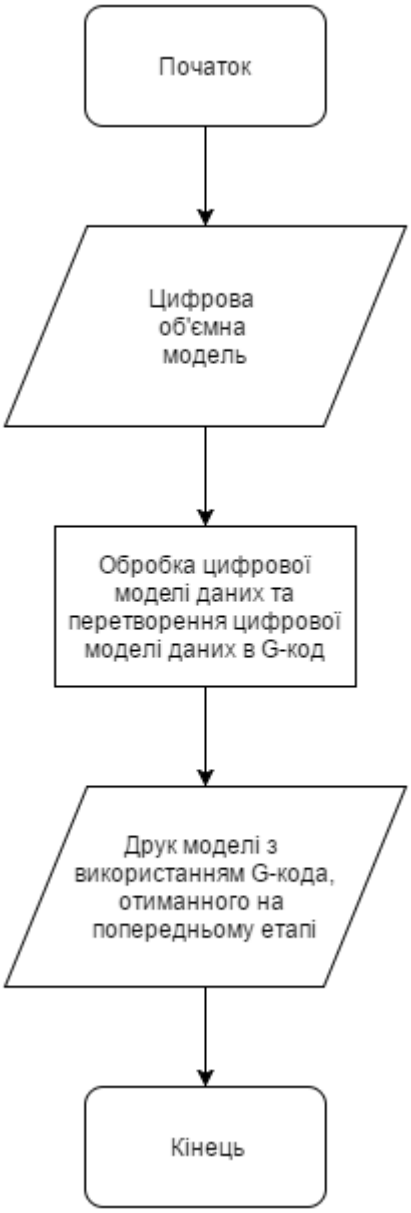

Рисунок 1.7 - Етапи 3D-друку

Якість надрукованної деталі напряму залежить від двох речей: налаштувань принтера та якості цифрової моделі деталі, з використанням якої і відбувається друк. Але якщо при гарній якості цифрової моделі апаратні

настройки легко можна змінити, і надрукувати деталь з потрібною точністю, то при поганій якості самої цифрової моделі, поліпшити друк апаратно буде неможливо. Тому дуже важливу роль грає якість моделювання деталі перед друком.

1.2. Висновки до розділу 1

В даному розділі були розглянуті основні етапи побудови поверхні під час 3D сканування та перед 3D друком. Крім цього, була обгрунтована актуальність даної роботи, яка полягає у тому, що завдяки поліпшенню якості моделювання деталі, можливо збільшити якість друку деталі.

2. МЕТОДИ ДЛЯ ПОРІВНЯННЯ ТОЧНОСТІ МОДЕЛЮВАННЯ ПОВЕРХОНЬ

2.1. Опис методів

Точність моделювання поверхні буде оцінюватися порівнянням різниць між моделюванням ідеальної поверхні (тобто такої, яка описана математичним рівнянням) і поверхні, яка створена з набору точок і змодельована одним з декілької алгоритмів.

Порівняння буде здійснюватися двома методами:

1. Порівняння точок;
2. Порівняння кутів між дотичною та координатною прямою.

А тепер детальніше про кожен з методів.

Порівняння точок буде проводитися таким чином. Спочатку визначаємо геометричну фігуру, яку можна описати математичною формулою. Після цього беремо набір точок для даної фігури і моделюємо об’ємну модель декількома методами або алгоритмами. Далі довільним чином генеруємо набір x-координат для точок фігури (ці x-координати не мають співпадати з координатами тих точок, з яких відбувалася реконструкція поверхні). По наявним x-координатам знаходимо координати y, z в контрольній фігурі, а також в реконструйованій поверхні і знаходимо різницю між цими значеннями. Після цього таким самим чином знаходимо похибку по координатам x та z.

Якщо даний алгоритм пояснювати на двомірній площині, то це буде виглядати так, як описано на рисунку 2.1.

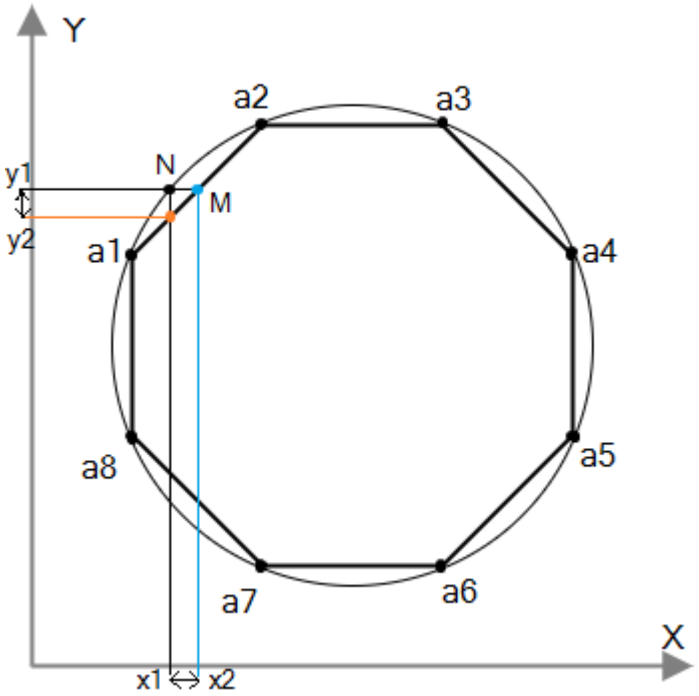

Рисунок 2.1 – Перший метод порівняння

На рисунку 2.1 представлено коло, яке можна описати математичним рівнянням – це наша контрольна фігура. З контрольної фігури вибрані точки a1, a2, a3, a4, a5, a6, a7 та a8, з яких ми і робимо реконструкцію поверхні. Реконструкція поверхні в даному випадку - це многокутник. Порівняння точності алгоритмів відбувається наступник чином: вибираємо координату тестової точки – y1. Далі по координаті y1 знаходимо значення координати x точки на контрольному колі (це точка N), а також знаходимо значення координати y на реконструйованій поверхні (це точка M), які будуть рівні числам x1 та x2 відповідно. Похибка буде становити x2-x1.

Після цього шукаємо похибку по осі y. Контрольне значення x1. По ньому обчислюємо значення y1 та y2 (контрольне значення та значення точки на реконструйованій поверхні). Похибка буде рівна значенню y2-y1.

Для другого методу підготовка проходить таким самим чином: потрібно вибрати фігуру, яку можна пописати математично, після чого треба вибрати набір точок, з яких відбувається реконструкція поверхні. А вже після цього виміри будуть відбуватися наступним чином. До точок, з яких ми виконували

реконструкцію поверхні, треба провести дотичні. Далі вимірюється кут C` між дотичною, яка проведена до контрольної фігури в цій точці і віссю, а також кут C`` між дотичною, яка проведена до реконструйованої фігури і тою самою віссю. Похибка в обчисленнях буде дорівнювати різниці між C`` і C`.

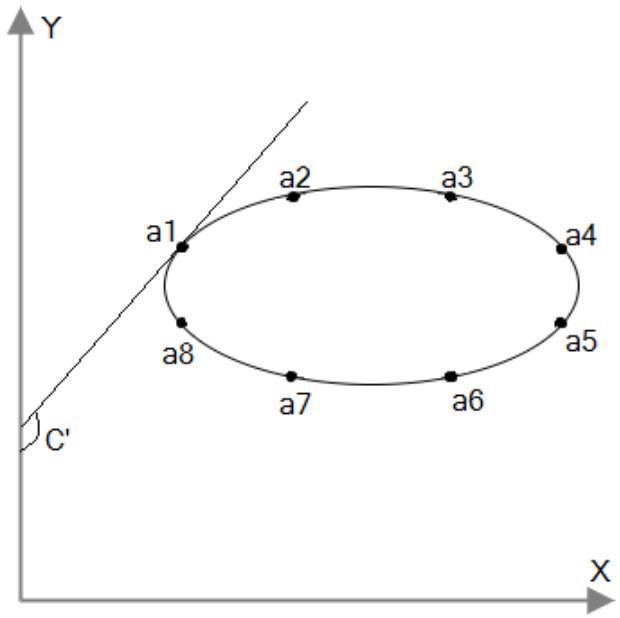

Рисунок 2.2 – Ілюстрація другого методу, контрольна фігура

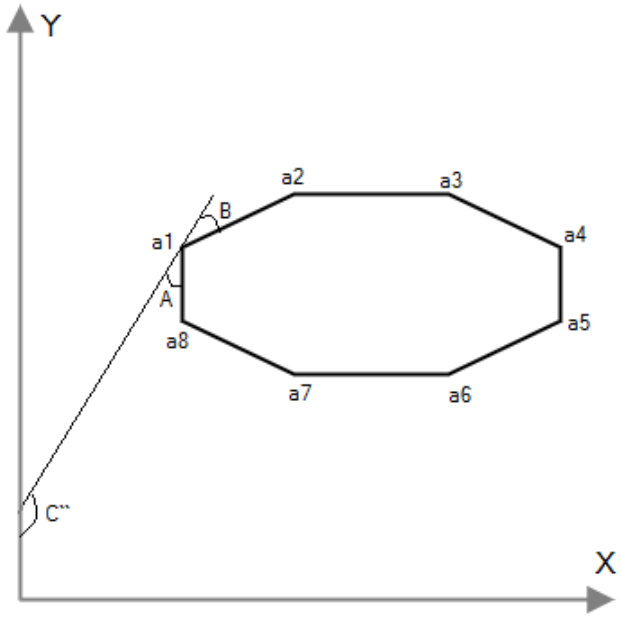

Рисунок 2.3 – Ілюстрація другого методу, реконструйована фігура

На рис. 2.2 зображен еліпс, з якого ми беремо точки a1-a8 і по ним будуємо реконструкцію поверхні. Дотична до точки a1 на контрольній фігурі при перетині з віссю Y утворює кут C`. Дотична до точки a1 на реконструйованій фігурі утворюється тоді, коли кут A дорівнює куту B. Ця дотична утворює з віссю Y кут C``. Похибка буде дорівнювати значенню C` - C``.

2.2. Висновки до розділу 2

В даному розділі були сформульовані два методи для порівняння точності моделювання поверхонь. Перший метод заключається в порівнянні координат точок на контрольній фігурі, а також координат точок на реконструйованій поверхні. Другий метод полягає у порівнянні кута, який утворюється між дотичною і віссю координат у контрольній фігурі та у реконструйованій поверхні.

3. ЗАГАЛЬНИЙ ОГЛЯД АЛГОРИТМІВ ТА МЕТОДІВ

Типовий процес збору даних про 3D модель, а також її побудову проходить через такі етапи:

- Сканування: отримання даних про поверхню об'єкта з допомогою вимірювального пристрою, такого як лазерний сканер.
- Компонування даних: об’єднання та корегування даних, приведення до одної координатної системи, якщо сканувань поверхні було декілька.
- Інтеграція даних: інтерполяція вимірянної вибірки даних або точок, що отримані з вибірки даних з поверхневим представленням зазвичай у вигляді триангульованої сітки.
- Перетворення моделі: проріджження або оптимізація сітки, монтаж з більш високим порядком перетворення, та ін.[4]

В даній дипломній роботі розглядаються саме техніки, які використовуються на етапі інтеграції даних.

Взагалі, всі існуючі інтерполяційні техніки можна розділити на 2 категорії: ті, які виконують побудову поверхні об’єкта, «формуючи» її та ті, які виконують побудову поверхні об’єкта «вирощуючи» поверхню з першопочаткової частинки. В «формуючих» методах маштаб (або об’єм, кількість) тетраедралізації (перетворення на тетраедр) вичисляються з набору точок, переважно з допомогою триангуляції Делоне у трьовимірному просторі. Після чого тетраедри усуваються з випуклої оболонки задля того, щоб максимально досягти оригінальної форми зісканованого об’єкта.

Методи, які «вирощують» поверхню розпочинаються з базового трикутника, після чого розглядається наступна точка з набору, яка приєднується до кордону вже створенної області. І так допоки всі існуючі в першопочатковому наборі точки не будуть розглянуті.

Як вже було зазначено, після сканування деталь представлена у вигляді набору точок. Для того, щоб потім отримати об'ємну модель використовують різні алгоритми. Всі вони розподіляються на 2 типи: алгоритми обчислювальної геометрії та алгоритми з використанням неявниї функцій.

Основними алгоритмами обчислювальної геометрії є:

- триангуляція Делоне,
- Альфа фігур (Alpha-shapes),
- Оборотних куль (Ball Pivoting).

Основними алгоритмами з використанням неявних функцій є :

- Крокуючих кубів (Marching Cubes),
- Метод Пуассона (Poisson),
- Hoppe,
- MPU.

Далі в роботі представлений опис кожного з алгоритмів.

3.1. Алгоритми обчислювальної геометрії

3.1.1. Тріангуляція Делоне

Одним з найпростіших способів опису поверхні є тріангуляція Делоне.

Тріангуляцією називають розбиття геометричної фігури на симплекси. У випадку тривимірного простору це буде розбиття багатогранника на тетраедри.

Темою даного підрозділу є тріангуляція, винайдена в 1934 р. російським математиком Борисом Делоне, яку зручно описувати разом з поняттям діаграми Вороного. Введемо основні терміни, що використовуватимуться нижче.

Визначальними об’єктами діаграми Вороного та тріангуляції Делоне є точки Вороного, які ще називають джерелами, генераторами або вершинами діаграми. Множина точок простору, для яких задана вершина є найближчою

утворюють комірку Вороного. Множина таких комірок в свою чергу утворює діаграму Вороного (Рисунок 3.1.).

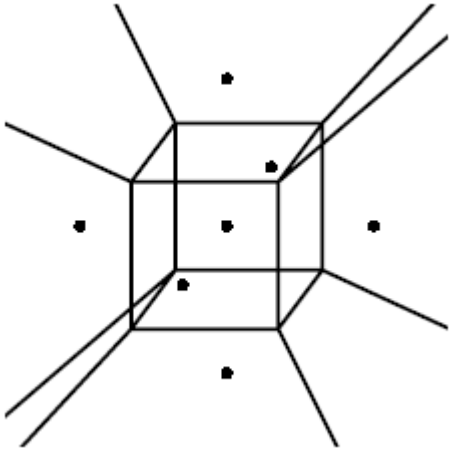

Рисунок 3.1 - Діаграма Вороного в тривимірному просторі.[5]

Тріангуляція Делоне – тріангуляція множини вершин, при якій сфера, описана навколо довільного симплекса, утвореного тріангуляцією, є пустою.

Рисунок 3.2. зображає тріангуляцію Делоне для вершин з рисунка 3.1.

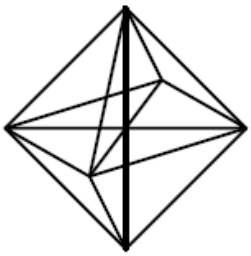

Рисунок 3.2 - Тріангуляція Делоне в тривимірному просторі.[5]

З рис. 3.1. та 3.2. легко бачити, що для множини точок відповідність між тріангуляцією Делоне та діаграмою Вороного – 1 до 1. Це означає, що кожній комірці Вороного можна поставити у відповідність тетраедр, отриманий за

допомогою тріангуляції. З точки зору обчислювальної геометрії діаграма Вороного є дуалом тріангуляції Делоне.

Найпростішим алгоритмом побудови тріангуляції Делоне в тривимірному просторі є інкрементний. Він полягає в додаванні вершин одна за одною, тим самим оновлюючи тріангуляцію на кожному кроці. Оновлення полягає в знаходженні усіх тетраедрів, для яких описані сфери містять нову вершину. Ці тетраедри видаляються і порожня частина розділяється на нові тетраедри так, що додана вершина є вершиною для таких нових тетраедрів. Див. рис. 3.3.

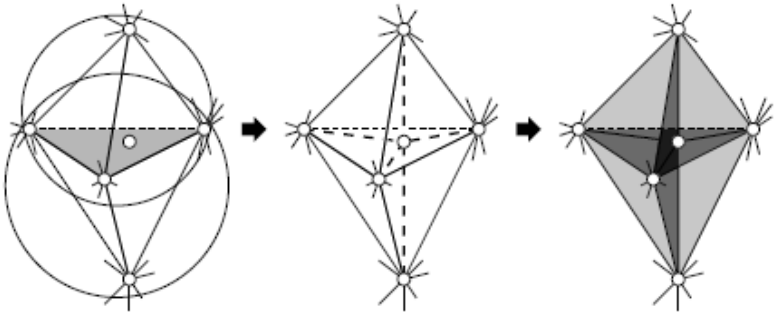

Рисунок 3.3 - Додавання нової вершини.[5]

В загальному випадку обчислювальна складність такого алгоритму – $O(n^2)$, де n – кількість вершин. Цей показник можна покращити до $O(n \log n)$, якщо вхідні дані, а саме, розміщення нових вершин та порядок їх вставки, структурувати. [5]

Наприклад, точки можна вважати структурованими, якщо вони розміщені лексикографічно по осях x та y, завдяки чому нова точка вставляється зовні існуючої межі або так, що вставка нової точки очікується близько до попередньої. [6]

3.1.2. Алгоритм альфа-фігур (Alpha Shapes)

Альфа-фігури (α-фігури) є узагальненням поняття опуклої оболонки для множини точок. Припустимо, що S – скінченна множина точок в R^3 і α – дійсне число, таке що $0 \leq \alpha \leq \infty$. α-фігурою S є багатогранник, що не обов'язково повинен бути опуклим або зв'язним. Для $\alpha = \infty$ α-фігура буде співпадати з опуклою оболонкою S. Однак, зі зменшенням значення α, α-фігура зменшуватиметься поступово утворюючи порожнини. Ці порожнини можуть об'єднуватися та формувати тунелі та отвори. Так, багатогранник зникне, коли α стане достатньо малим, що сфера радіусу α або кілька таких сфер зможуть зайняти місце не включаючи в себе точок з S. Див. рис. 3.4.

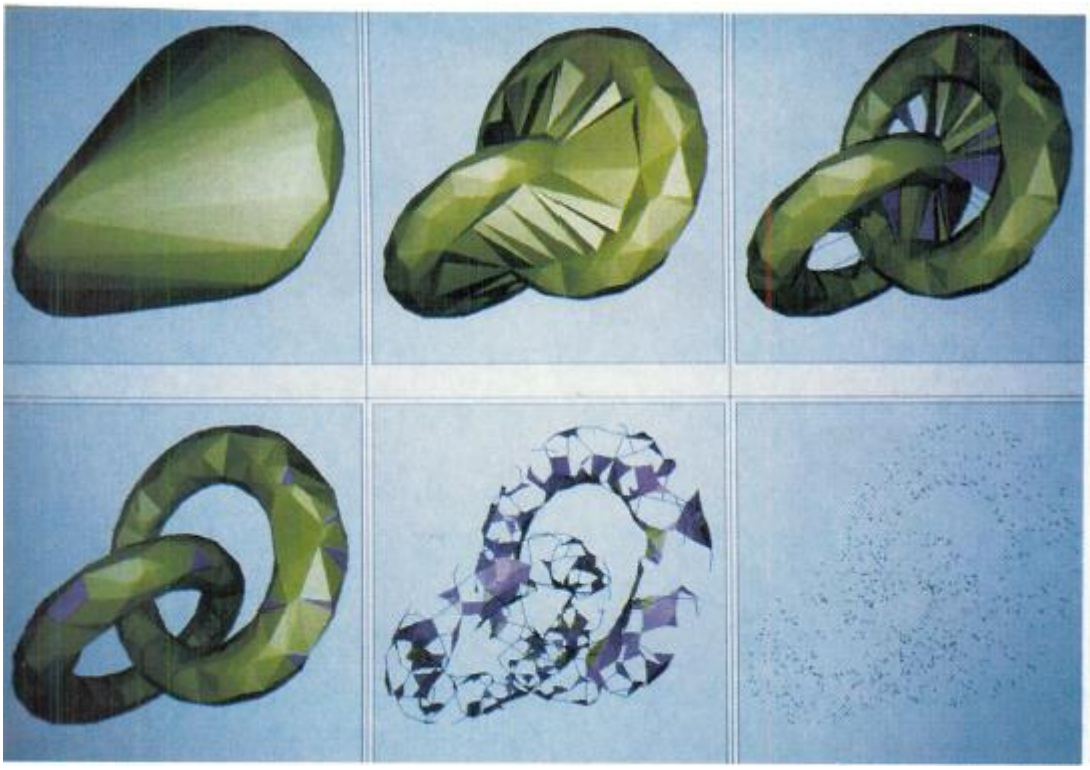

Рисунок 3.4 - α-фігури, починаючи з α = ∞ у лівому верхньому куті і поступово зменшуючи значення α вправо та вниз до α = 0 в правому нижньому куті.[8]

Тут і надалі ми припускаємо, що точки множини S знаходяться у загальній позиції, тобто:

- Жодні 4 точки не лежать на одній площині;
- Жодні 5 точок не лежать на одній сфері;
- Для довільного фіксованого α, найменша сфера через довільні 2, 3 або 4 точки з S матиме радіус відмінний від α.

Для реалізації α-фігур застосуємо алгоритм інкрементної заміни заснований так званих місцевих перетвореннях або замінах:

1. Відсортуємо множину точок S по вказаному напрямку.
2. Перейменуємо точки так що $\{p_1, \dots, p_n\}$ розміщені впорядковано.
3. Ініціалізуємо тріангуляцію Делоне D, так щоб вона містила один тетраедр з вершинами в точках p_1, p_2, p_3, p_4.
4. Для i = 5 ... n:
   - 4.1. Додати точку p_i, з'єднавши її з усіма вершинами D, видимими з p_i.
   - 4.2. Використати заміну для того, аби перетворити D в тріангуляцію Деолне з вершинами $\{p_1, \dots, p_i\}$.

Розглянемо заміну з п.3.2 алгоритму детальніше. Припустимо, що 2 тетраедри σ_U та σ_V мають спільний трикутник σ_T, тобто $T = \{p_i, p_j, p_k\}$, $U = T \cup \{p_u\}$, $V = T \cup \{p_v\}$. Трикутник σ_T називають локальним Делоне, якщо він належить тріангуляції Делоне $V \cup U$. В даному випадку ми розглядаємо випадок, коли точка p_v лежить за межами сфери, описаної навколо σ_U.

Алгоритм інкрементної заміни повинен змінити тріангуляцію так, аби вона знову стала тріангуляцією Делоне. Для цього він визначає трикутники, які не є локальними Делоне і заміняє їх. Нехай σ_T – такий трикутник, що межує з тетраедрами σ_U та σ_V. Знову ж таки, припустимо, що $T = \{p_i, p_j, p_k\}$, $U = T \cup \{p_u\}$, $V = T \cup \{p_v\}$. Якщо фігура $V \cup U$ опукла, то трикутник σ_T може бути замінений за допомогою з'єднання p_u та p_v. Разом з ребром, утвореним таким з'єднанням,

утворюються трикутники, що з'єднують три вершини T. Цю операцію називають заміною трикутника на ребро. Розглянемо протилежний випадок, якщо фігура $V \cup U$ не опукла. Також припустимо, що існує третій тетраедр σ_Z, натягнутий на чотири з п'яти точок $V \cup U$, наприклад $Z = \{p_i, p_j, p_u, p_v\}$. В такому випадку трикутники, не дотичні до ребра, що з'єднує p_i та p_j, можуть бути замінені на єдиний трикутник з вершинами p_u, p_v і p_k. Таку операцію називають заміною ребра на трикутник. Якщо ж тетраедра σ_Z не існує, то відповідно немає заміни, яка могла б вилучити σ_T.

Складність даного алгоритму також становить $O(n^2)$. [8]

3.1.3. Алгоритм оборотних куль (Ball Pivoting)

Метод оборотних куль має за мету знайти трикутну сітку, що інтерполює множину точок. На практиці така множина точок отримується, наприклад за допомогою сканування поверхні. Алгоритм обчислює трикутну сітку для інтерполяції заданої множини точок. Тріангуляція множини точок відбувається за рахунок «обертання» кулі радіусом π по множині точок. Алгоритм починає з базового трикутника і «котить» кулю заданого радіусу ρ навколо цього трикутника. Протягом такого «котіння», м'яч обертається навколо, допоки не доторкнеться до іншої точки з заданої множини, тим самим утворюючи інший трикутник. На наступному кроці обирається інший базовий трикутник і так триває, доки усі базові точки не будуть розглянуті.

Для кращого розуміння розглянемо рис. 3.5.

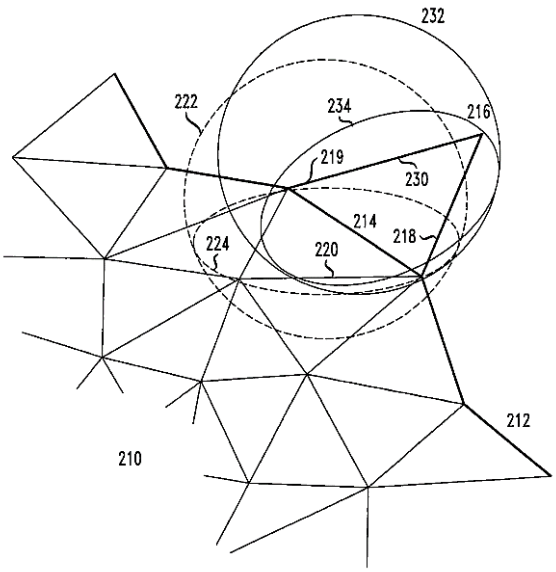

Рисунок 3.5 - Оборотні кулі[9]

На цьому рисунку трикутна сітка 210 вже була частково обчислена. Сукупність «фронтальних» ребер 212 утворює межу сітки. Ребро 214 обране для наступного кроку алгоритму, тобто «котіння» кулі. Поточна куля для ребра 214 позначена штрихованим колом 222. Куля 222 дотикається до трьох вершин трикутника 220. Перетин кулі 222 з площиною, що містить трикутник 220 позначена штрихованим еліпсом 224. Куля 222 обертається навколо ребра 214 в той самий час дотикаючись до двох крайніх точок.

У даному прикладі, куля обертається вільно, доки не торкнеться нової точки 216. Позиція кулі 222, коли вона доторкнулася до точки 216, позначена суцільним колом 232, а перетин цієї кулі з множиною, що містить ребро 214 та точку 216 – суцільним еліпсом 234. Нова точка 216 формує трикутник 230 з ребром 214, навколо якого оберталася куля. Новий трикутник 230 стає частиною трикутної сітки. Лінія «фронтальних» ребер 212 зміщується за рахунок віднімання ребра 214 та додавання нових ребер 218 та 219. Процес продовжують, обираючи нове ребро з 212. Дане зображення описує застосування алгоритму для двовимірного простору, однак його легко спроектувати на тривимірний простір.

Алгоритм оборотних куль також пов’язаний з α-фігурами. Насправді, для кожного трикутника τ, обчисленого за допомогою обходу ρ-кулею, існує порожня відкрита куля bτ, чий радіус менший за ρ. Відповідно алгоритм обчислює підмножину з 2 комірок ρ-фігури з S. Ці комірки є також підмножиною тривимірної тріангуляції Делоне для множини точок. [9]

На практиці реалізація такого алгоритму складатиметься з двох основних частин:

- Пошук базового трикутника – алгоритм пошуку трьох вершин, що лежать на поверхні порожньої сфери заданого радіусу r.
- Розширення тріангуляції – трикутники додаються до тріангуляції за допомогою обертання кулі навколо фронтальних ребер, доки фронтальні ребра не закінчаться.

Опишемо кожну частину детальніше.

Пошук базового трикутника – це перший крок алгоритму. Він розпочинається з однієї точки і перевіряє для кожної пари двох її сусідів, чи можна побудувати такий трикутник, щоб його описана сфера радіусу r була порожньою. На практиці більше одного трикутника може задовольняти такій умові. Коли базовий трикутник знайдено, його ребра додаються до множини фронтальних ребер.

Розширення тріангуляції здійснюється на основі фронтальних ребер. Нехай e – фронтальне ребро або ребро фронту розширення (так інакше називають сукупність всіх фронтальних ребер). Далі ми «обертаємо» кулю радіусу r з її початкової позиції навколо ребра e, доки вона не зустрінеться з точкою v. Після цього перевіряємо, чи сфера є порожньою. Якщо так, то проводимо нове ребро до точки v. Даний приклад проілюстровано на рис. 3.6.

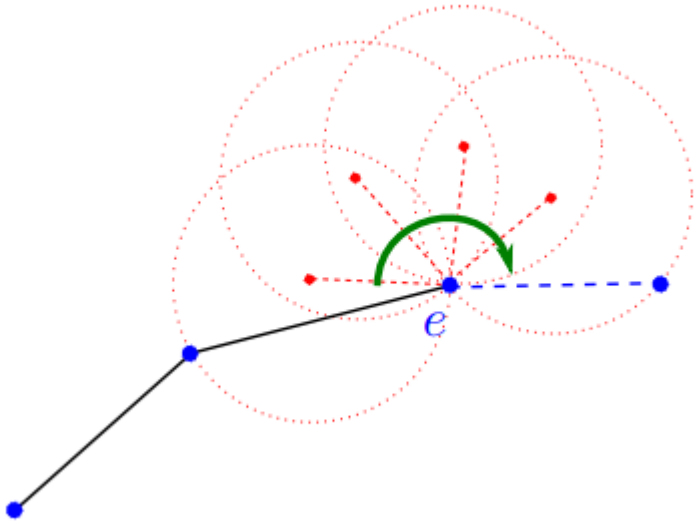

Рисунок 3.6 - Обертання кулі.[9]

На цьому кроці можливі наступні випадки (Рисунок 3.7.):

а. Розширення: v не з’єднана з жодним ребром. Тоді нове ребро додається, e вилучається з фронту розширення, а два нових ребра до нього додаються. Розмір фронту збільшується на 1.

б. Склеювання: v належить до фронтального ребра, але не з’єднане з вершинами e. В такому випадку e вилучається з фронту, а натомість два нових ребра до нього додаються. Т. ч. фронт збільшується на 1.

в. Заповнення тунелю: v вже з’єднана з обома вершинами ребра e. В такому випадку новий трикутник утворюється і e вилучається з фронту розширення, разом з ребрами, що з’єднують e і v. Розмір фронту зменшується на 3.

г. Заповнення порожнини: v з’єднана з однією з вершин ребра e. В такому випадку лише 1 ребро додається до фронту, а 2 вилучаються.

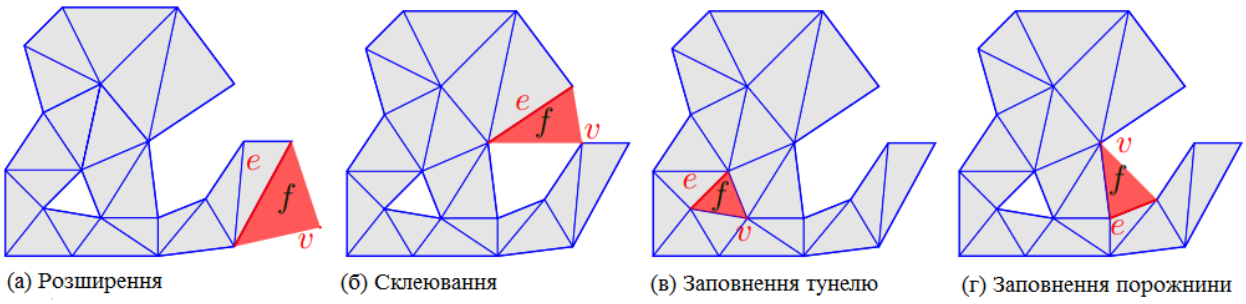

Рисунок 3.7 - Розширення фронту[10]

Процес розширення триває, доки фронт не залишиться порожнім [10].

3.2. Алгоритми та методи з використанням неявних функцій

3.2.1. Алгоритм Крокуючих Кубів (Marching Cubes)

У 1987 році Вільямом Лоренсеном і Харві Клайном було представлено алгоритм, що отримав назву крокуючих кубів, і який використовувася для створення трикутних моделей поверхонь з тривимірних медичних даних постійної густини.

Для вирішення реконструкції поверхні здійснюється два основних кроки. Під час першого визначається поверхня відповідно до заданого значення і будуються трикутники. На другому кроці, для забезпечення якості зображення поверхні, розраховуються нормалі поверхні в кожній вершині кожного трикутника.

Крокуючі куби використовуються принцип «розділяй і володарюй» для визначення поверхні в логічному кубі, утвореному з восьми пікселів, – по 4 на кожен з двох прилеглих зрізів.

Алгоритм визначає, як поверхня перетинається з кубом, а потім рухається (крокує) до наступного куба. Аби знайти перетин поверхні в кубі, присвоїмо одиницю вершині куба, значення в якій перевищує (або рівне) значенню поверхні, яку ми конструюємо. Такі вершини знаходяться на поверхні.

Вершинам куба з значеннями менше поверхні присвоюємо нулі. Вони знаходяться поза поверхнею. Поверхня перетинає ті ребра куба, де одна вершина є зовні поверхні (1), а інша всередині (0). З цим припущенням ми знаходимо топологію поверхні всередині куба, пізніше знаходячи перетин.

Оскільки є 8 вершин в кубі і 2 стани (зовні і всередині), існує $2^8 = 256$ способів, якими поверхня може перетнути куб. Пронумерувавши всі 256 способів, створюємо таблицю для пошуку перетинів між ребрами і поверхнею. Таблиця міститиме ребра, які є перетнутими в кожному випадку.

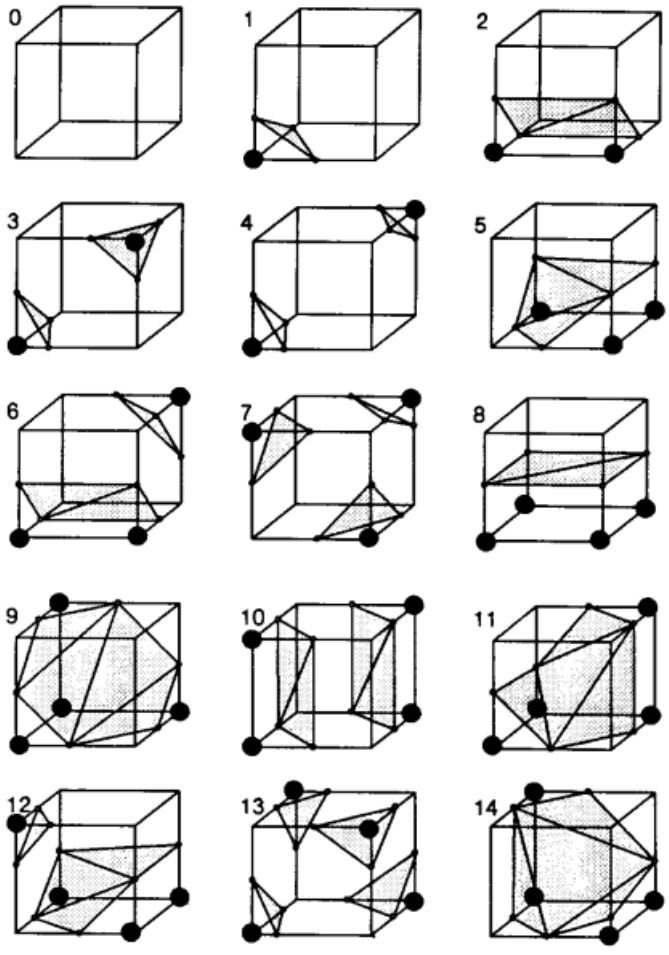

Рисунок 3.8 - Тріангульовані куби [11]

Тріангулювати 256 випадків виснажливо. Дві симетрії куба (незмінна топологія тріангульованих поверхонь та осьова симетрія) зменшують проблему від 256 випадків до 14 шаблонів, які зображені на рис. 3.8.

Найпростіший шаблон, 0, виникає, якщо всі значення вершин є над або під заданою змінною і не породжує трикутників. Наступний, 1, виникає, якщо поверхня відділяє одну вершину від інших сімох, тим самим породжуючи один трикутник, визначений трьома точками перетину.

Створимо індекс для кожного випадку в залежності від стану вершини. Використовуючи нумерацію вершин, зображену на рис.3.9., восьмибітний індекс міститиме один біт для кожної вершини.

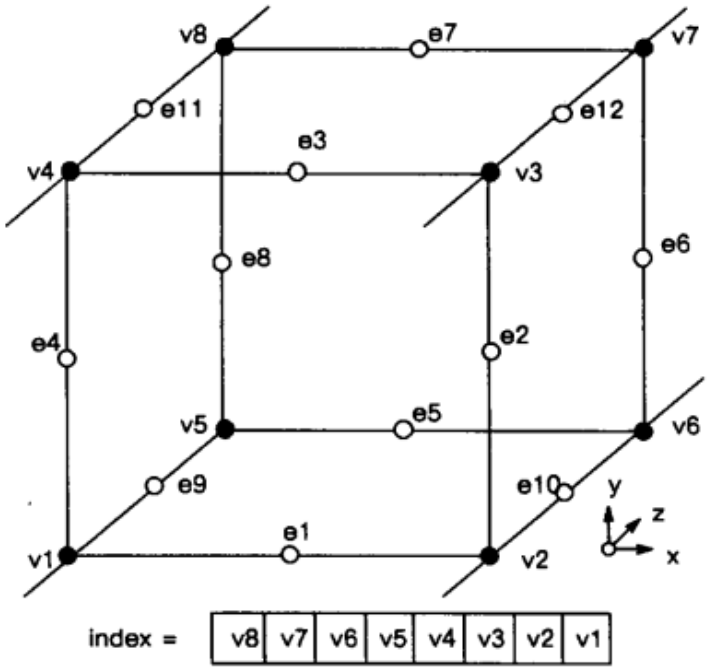

Рисунок 3.9 - Індексація вершин [11]

Індекс служить вказівником на таблицю ребер, яка дає всі можливі перетини для заданої конфігурації куба.

Використовуючи індекс для взначення ребра, яке перетинає поверхня, можна інтерполюваи перетин поверхні вздовж ребра. Використовуватиметься лінійна інтерполяція.

В завершенні в крокуючих кубах розраховується одинична нормаль до кожної вершини трикутника. При використанні поверхні сталої густини нормаль буде рівна градієнтному вектору, що рівний похідній від функції густини:

$$\bar{g}\ (x, y, z) = \nabla\ \bar{f}\ \ (x, y, z)$$

Для розрахунку градієнтного вектору до поверхні, ми спочатку обчислимо градієнтний вектор в вершинах куба і лінійно інтерполюємо градієнт у точці перетину. Градієнт в вершині куба (i, j, k), обчислюючи різницеві рівняння вздовж трьох осей:

$$G_x(i, j, k) = \frac{D\ (i+1, j, k) - D\ (i-1, j, k)}{\Delta x}$$

$$G_y(i, j, k) = \frac{D\ (i, j+1, k) - D\ (i, j-1, k)}{\Delta y}$$

$$G_z(i, j, k) = \frac{D\ (i, j, k+1) - D\ (i, j, k-1)}{\Delta z}$$

В формулах вище D(i, j, k) – це густина в пікселі (i, j) на перерізі k, а $\Delta x, \Delta y, \Delta z$ це довжини ребер куба. Розділивши градієнт на його довжини, отримаємо одиничну нормаль у вершині. [11]

3.2.2. Метод Пуассона (Poisson)

Іншим прикладом використання неявних функцій для вирішення проблеми відтворення поверхні є метод Пуассона, під час його розраховується характеристична функція χ, яка рівна 1 в точках всередині моделі та 0 в точках

поза моделлю, а потім отримується відновлена поверхня з використанням відповідної ізоповерхні.

Важливо розуміти, що існує відповідність між орієнтованими точками, отриманими з поверхні моделі та характеристичною функцією моделі. А саме, градієнт характеристичної функції є нульовим вектором майже скрізь (оскільки характеристична функція є константою майже скрізь), окрім точок поблизу поверхні, де він рівний внутрішньо напрямленій нормалі поверхні. Таким чином, орієнтовані точки можна сприймати як зразки градієнта характеристичної функції моделі (див. рис. 3.10.).

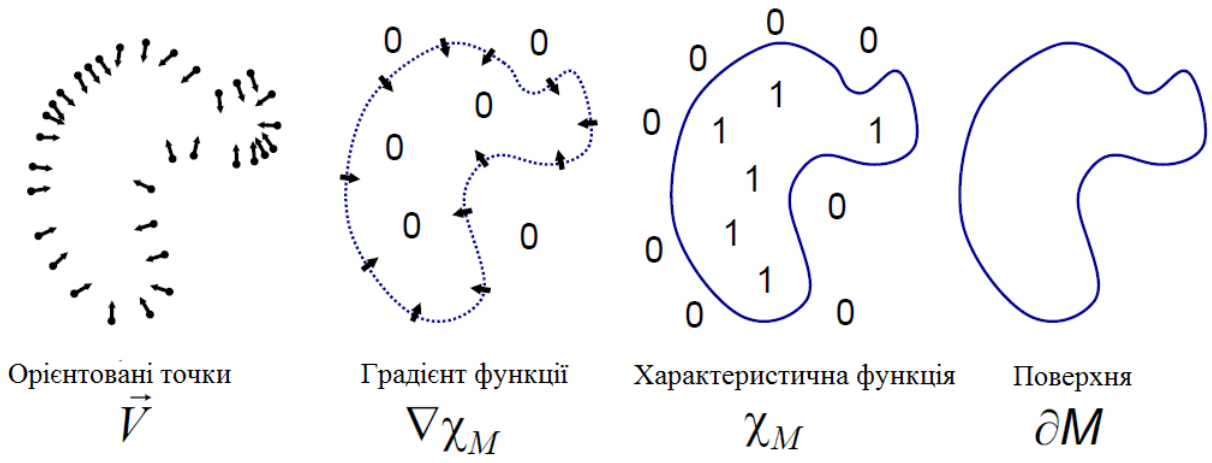

Рисунок 3.10. Відтворення Пуассона в двовимірному просторі. [12]

На вхід подається S – набір зразків s Є S, кожен з яких складається з точки s.p та внутрішньо орієнтованої нормалі s.$\bar{N}$. Припускаємо, що така точка лежить неподалік від поверхні ∂M невідомої моделі M. Нашою метою буде відтворити плавну, тріангульовану апроксимацію поверхні наближуючи характеристичну функцію моделі та отримуючи ізоповерхню, як зображено на рисунку 3.11.

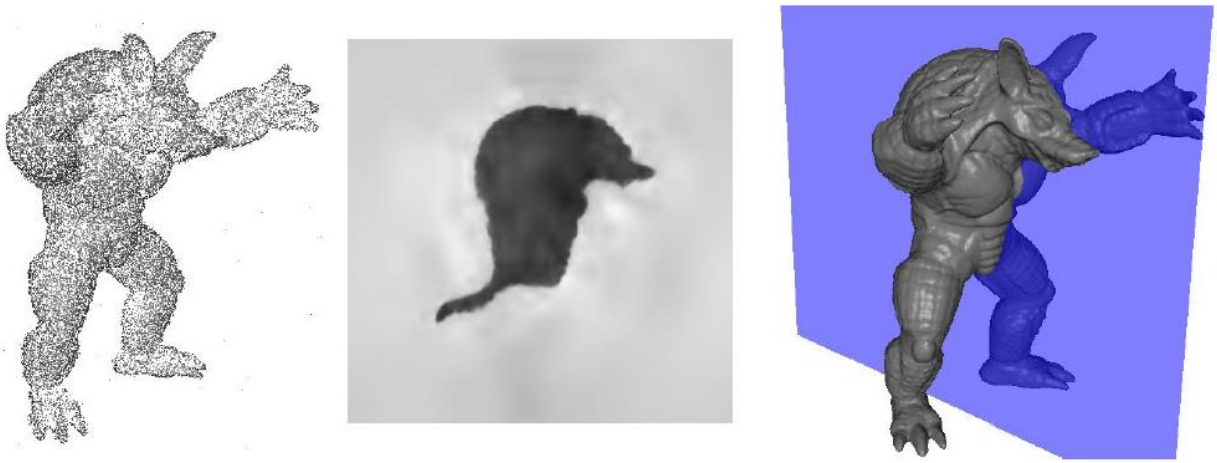

Рисунок 3.11. Точки отримані під час сканування моделі (ліворуч), відтворення Пуассона для поверхні (праворуч) та візуалізація характеристичної функції (посередині). [12]

Оскільки Характеристична функція є частково-константна, явне обчислення її градієнтного поля завершиться отриманням векторного поля з необмеженими значеннями на обмежувальній поверхні. Аби уникнути цього, згорнемо характеристичну функцію за допомогою згладжувального фільтру і врахуємо градієнтне поле згладженої функції. Наступна лема формалізує відношення між градієнтом згладженої характеристичної функції та полем нормалей поверхні.

Лема: Для неперервної M з межею ∂M, нехай χ_M – характеристична функція M, $\bar{N}_{\partial M}(p)$ – внутрішньо направлена нормаль в точці $p \in \partial M$, $\tilde{F}(q)$ – згладжуючий фільтр, і $\tilde{F}(q) = \tilde{F}(p - q)$ – перетворення для точки p. Градієнт згладженої характеристичної функції рівний векторному полю, отриманому в результаті згладжуання нормального поля поверхні:

$$\nabla\left(\chi_M * \tilde{F}\right)(q_0) = \int_{\partial M} \widetilde{F_p}(q_0)\overline{N_{\partial M}}(p)dp$$

Доведення є в [12].

Звісно ж не можна оцінити інтеграл поверхні, оскільки ми все ще не знаємо геометрію поверхні. Однак, вхідні дані є орієнтованими точками, що надає достатньо інформації для апроксимації інтергралу скінченною сумою. Властиво, використовуючи набір точок S для розподілу ∂M на різні частини $P_S \subset \partial M$, ми можемо апроксимувати інтеграл на частинах P_S значеннями в точці s.p:

$$\nabla\left(\chi_M * \tilde{F}\right)(q) = \sum_{s\in S} \int_{P_S} \widetilde{F_p}(q)\overline{N_{\partial M}}(p)dp \approx$$

$$\approx \sum_{s\in S} |P_s| \widetilde{F_{s.p}}(q) s.\bar{N} \equiv \bar{V}(q)$$

Слід відзначити, що при реалізації, особливу увагу слід приділити вибору фільтру. А саме, фільтр повинен задовольняти двом умовам. З одної сторони він повинен бути достатньо вузьким для того, аби уникнути надмірного згладжування даних. З іншого – він повинен бути достатньо широким, аби P_S можна було зручно апроксимувати значеннями s.p.

Сформувавши векторне поле $\bar{V}$, ми хочемо знайти рішення для функції $\tilde{\chi}$ такої, що $D\tilde{\chi} = \nabla \bar{V}$. Однак, $\bar{V}$ в загальному випадку не інтегровне, отже немає точного вирішення в загальному випадку. Аби знайти найкраще апроксимаційне розв'язання, застосуємо оператор дивиргенції для формування рівняння Пуассона [12]:

$$\Delta\tilde{\chi} = \nabla \bar{V}$$

3.2.3. Алгоритм Хоппа для відтворення поверхні

Даний алгоритм запропонував у своїй докторській дисертації математик Вашингтонського університету Х'ю Хопп у 1994 році.

Методи, розглянуті у попередніх розділах, при їх практичній реалізації вимагають накладання додаткових умов, наприклад впорядкування даних, знання роду поверхні або інформації про орієнтацію. Даний алгоритм, на

противагу іншим, вимагає лише знання координат точок в тривимірному просторі.

Алгоритм складається з трьох основних стадій:

1. Початковий розрахунок поверхні.
2. Оптимізація сітки.
3. Покрокова оптимізація гладкої поверхні.

З множини точок $X=\{x_1,\ldots,x_n\}$, що згідно з припущенням знаходяться близько до невідомої поверхні U, алгоритм на першій стадії генерує сітку, що апроксимує U.

Основна ідея полягає в розрахунку напрямленої відстані від X до U. Напрямлену відстань від довільної точки $p \in R^3$ до відомої замкненої поверхні U рохраховують як $d_U(p) = s(p) \bullet d(p, U)$, де $s(p) = \pm 1$ залежно від того, з якої сторони поверхні знаходиться p. Якщо U – обмежена поверхня, то неперервна напрямлена відстань може бути визначена, якщо точка знаходиться в трубчастому околі D поверхні. В нашому випадку важливо відзначити, що знання напрямленої функції відстані d_U – еквівалентне знанню поверхні U. Неявний вигляд U можна подати за допомогою нульової множини $Z(d_U) = \{p : d_U(p) = 0\}$. Незважаючи на те, що ми не знаємо, ні d_U, ні U, нашою метою буде розрахунок d_U з заданих точок та подальше отримання апроксимації їх нульової множини.

Якщо говорити більш детально, то перша стадія алгоритму складається з двох кроків. Перший визначає функцію $\tilde{d}_U$: $D \rightarrow R$, де $D \subset R^3$ – область біля заданих точок, така що $\tilde{d}_U$ розраховує напрямлену функцію відстані d_U. Таким чином $Z(\tilde{d}_U) = \{p : \tilde{d}_U(p) = 0\}$ є нашим наближенням U (Рисунок 3.12).

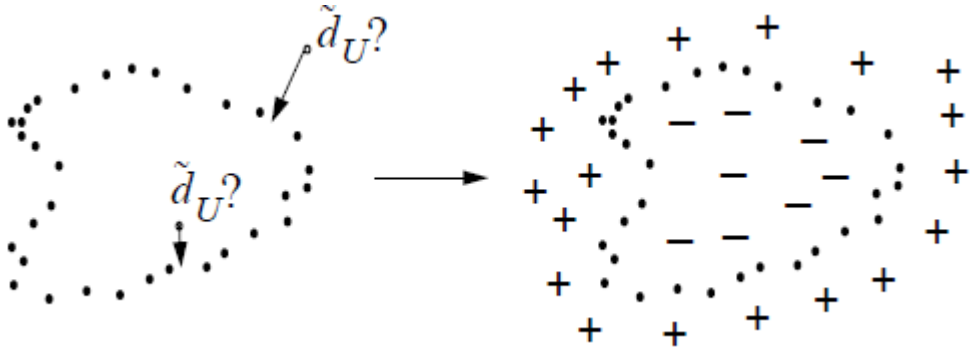

Рисунок 3.12 - Розрахунок d_U по заданих точках. [13]

На другому кроці ми використовуємо контурний алгоритм для отримання наближення для $Z(\tilde{d}_U)$ у вигляді сітки (Рисунок 3.13).

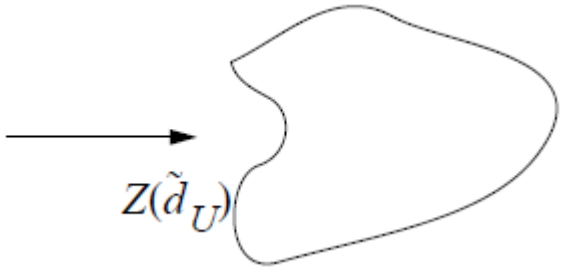

Рисунок 3.13 - Наближення $Z(\tilde{d}_U)$. [13]

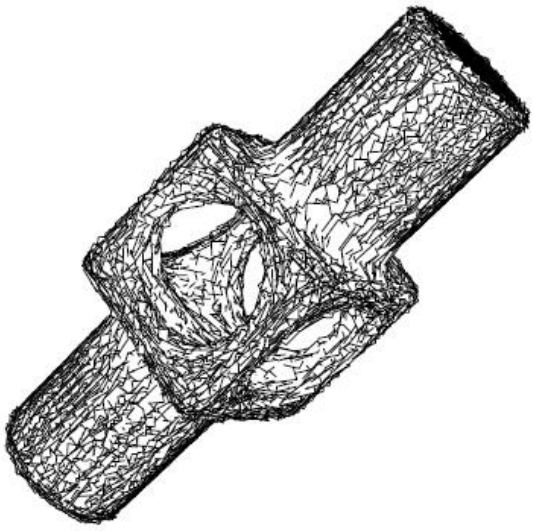

Рисунок 3.14 - Напрямлені дотичні площини. [13]

Ключовим моментом в обчисленні напрямленої функції відстані є асоціювання напрямленої площини з кожною із заданих точок. Ці обчислені дотичні площини служать місцевими лінійними апроксимаціями поверхні. Хоча побудова дотичних площин є відносно легкою, вибір їх орієнтації так, щоб визначити глобально сумісну орієнтацію для поверхні, є однією з основних перешкод, з якими стикається алгоритм. Напрямлені дотичні площини зображені на рисунку 3.14.

Такі площини в подальшому використовуються для визначення напрямленої функції відстані до поверхні. Приклад сітки, отриманої алгоритмом згладжування показаний на Рисунок 3.15 [13].

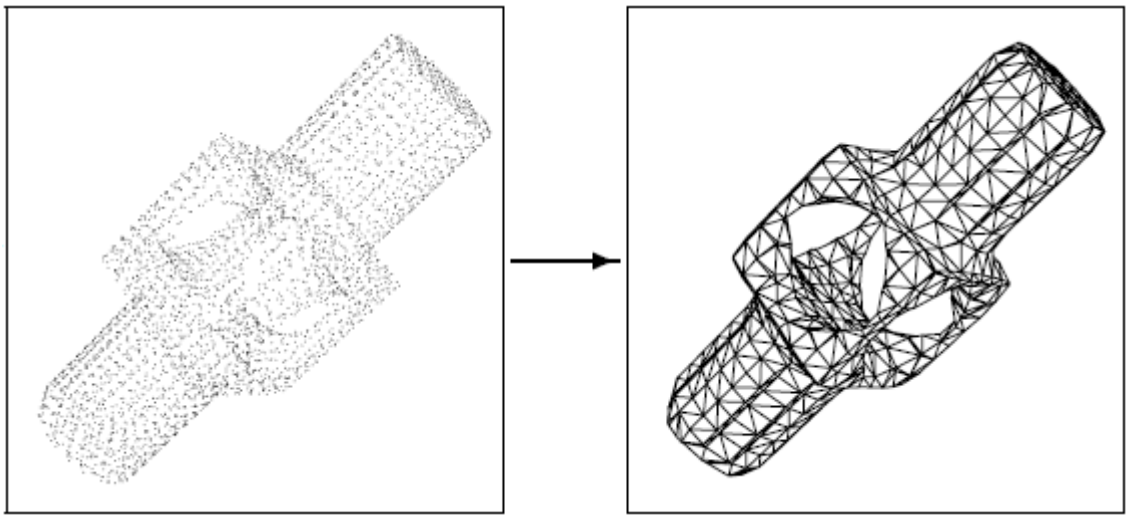

Рисунок 3.15 - Отримання початкової сітки з множини заданих точок [13]

3.2.4. Алгоритм MPU (Multilevel Partition of Unity Implicits – Багаторівневий розподіл об’єднаних неявних функцій)

Концепція алгоритму MPU складається з трьох ключових чинників:

1. Частинні квадратичні функції, що захоплюють локальні частини поверхні.
2. Функції зважування (партиції/розподіли поверхні), що об’єднують разом ці локальні функції частин поверхні.

3. Метод розподілу за допомогою вісімкового дерева.

В даному методі вводиться новий клас неявних моделей, розроблених спеціально для задоволення вимог швидкої та точної побудови поверхонь з великих множин точок. Цей клас має однойменну алгоритмом назву MPU.

Для заданої множини точок $P = \{p_1,\ldots p_N\}$, отриманої з поверхні в R^3, неявна функція MPU $f: R^3 \rightarrow R$ подає апроксимацію напрямлених функцій відстані відносно поверхні. Ця апроксимація є точною біля поверхні і приблизною на далеких відстанях від поверхні. Поверхня сама по собі потім апроксимується нульовою множиною функції відстані. Припустимо, що точки з P мають одиничні нормалі $N = \{n_1,\ldots,n_N\}$, що показують напрямленість поверхні.

На практиці такі нормалі можуть бути розраховані або з початкового сканування поверхні або підстановкою найменших квадратів до P.

Для створення неявного вигляду, розпочинаємо з паралелепіпеда, що обмежує множину точок і створюємо його розподіл на основі вісімкового дерева.

Для кожної комірки вісімкового дерева визначається частинна квадратична функція (функція місцевої фігури) така, щоб вона відповідала точкам комірки. Ці функції служать в якості напрямлених функцій відстані і мають значення 0 біля заданих точок так є додатними (всередині) або від'ємними (зовні) на відстані від заданих точок. Наближені нормалі точок використовуються для розрізнення зовнішньої та внутрішньої орієнтації. Якщо наближення функції недостатньо точне (тобто не співпадає з точками), комірка розподіляється і процедура повторюється, доки не буде отримано необхідного рівня точності. В місцях біля спільної межі двох комірок, функції об'єднуються відповідно до ваги з розподілу об'єднаних функцій. Так задається глобальна неявна функція поверхні.[14]

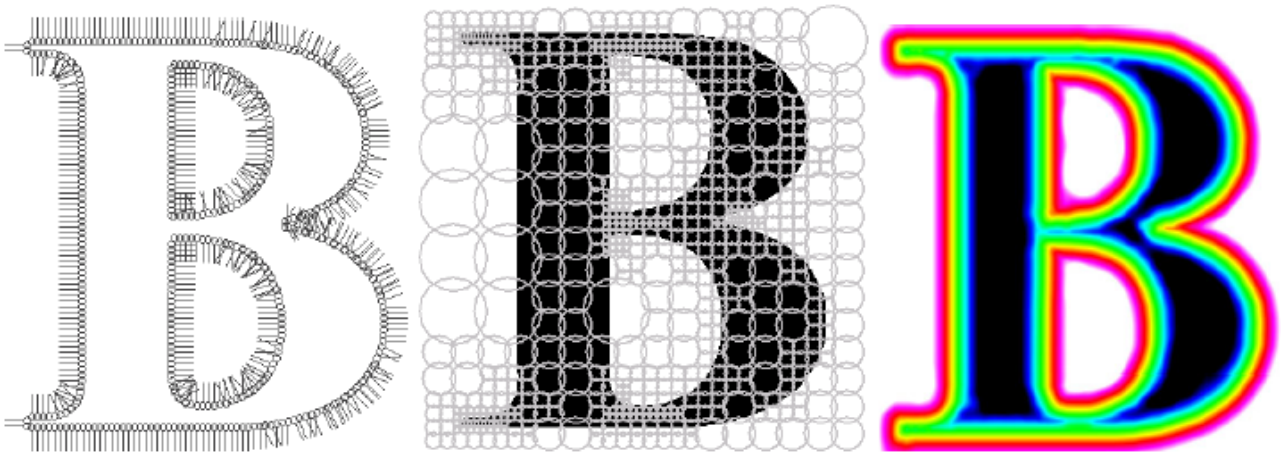

Рисунок 3.16 - MPU. Зліва – набір точок з нормалями. Посередині – круги, що відповідають адаптивному розподілу. Справа – функція відстані, в якій нульовий рівень розміщений між жовтою та зеленою лініями. [14]

3.3. Висновки до розділу 3

В даному розділі були розглянуті сім алгоритмів і методів створення опису поверхонь. Розглянуті алгоритми і методи є двох видів:

- алгоритми обчислювальної геометрії. Це триангуляція Делоне, алгоритм альфа-фігур та алгоритм оборотних куль;
- алгоритми і методи з використанням неявних функцій. Це алгоритм крокуючих кубів, метод Пуассона, алгоритм Хоппа та алгоритм MPU.

Розділ вводить в основну концепцію даних алгоритмів, показуючи їх взаємозв'язок, а також подаючи короткий опис понять обчислювальної геометрії, на які дані методи спираються.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТА МЕТОДІВ

4.1. Алгоритми обчислювальної геометрії

4.1.1. Реалізація триангуляції Делоне та алгоритму Альфа-форм

Реалізація триангуляції Делоне, а також алгоритму Альфа-форм була зроблена з допомогою бібліотеки CGAL (Computational Geometry Algorithms Library). Дана бібліотека є доволі складною у налаштуванні, адже для цього треба було завантажити Bootstrap, CMake та Qt5 для відображення графіки.

4.1.2. Реалізація алгоритму оборотних куль (Ball pivoting)

Дана реалізація була досліджена з допомогою онлайн-ресурсу IPOL Journal – Image Processing On Line. IPOL позиціонує себе як відкритий науковий журнал для відтворюваних досліджень. Код, з допомогою якого відбувалася реконструкція поверхні, був написаний Джулією Дігн та Мігелєм Коломом та опублікований на сайті 1 липня 2014 року. [1]

Використання онлайн ресурсу дуже просте. Потрібно лише завантажити набір точок на сайт і поверхня моделі буде автоматично реконструйована. Для завантаженого файлу з точками є обмеження – не більше 150000 точок. Або можна використати демо-приклади, що наявні на сайті. Мною був використаний набір точок, який реконструюється в так званого «стенфордського зайця» (див. рис. 4.1).

Дані на сайт потрібно завантажувати у форматі .ply. Результати після обробки можна скачати також у вигляді файлу у форматі .ply.

Той самий заєць в програмі MeshLab буде виглядати так, як показано на рис. 4.2.

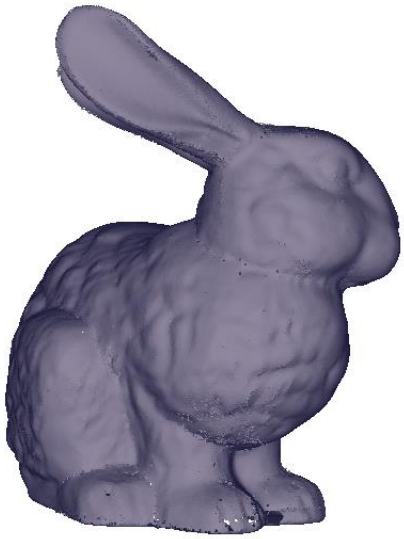

Рисунок 4.1 – Реконструйована модель стенфордського зайця, що відображається на сайті

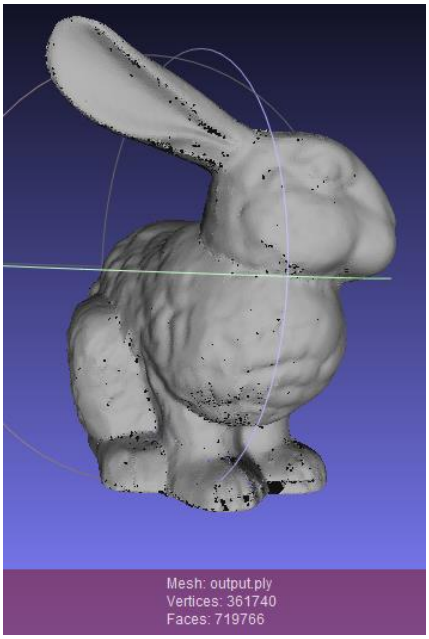

Рисунок 4.2 – Реконструйована модель стендфордського зайця, відображення з допомогою MeshLab

Як можемо бачити, алгоритм реконструює поверхню не так якісно, як попередні алгоритми, адже на поверхні зайця можемо бачити отвори. Наявність цих отворів може регулюватися.

4.2. Алгоритми з використанням неявних функцій

4.2.1. Реалізація алгоритму Крокуючих Кубів

Для реалізації алгоритму Крокуючих Кубів була використана програма, автор якої є Ерік Смістад. Ця програма завантажена на гітхаб аккаунті програміста. Всю програму можна завантажити один архівом, але також потрібні додавткові настройки. А власне потрібна встановлена і налаштована програма CMake.

Після завантаження, користуючись вказівками Еріка Містарда, її досить легко запустити. Для цього потрібно лише:

1. Скомпілювати програму командою в консолі cmake CMakeLists.txt.
2. Запустити програму, також використовуючи консоль і команду у вигляді filename.raw sizeX sizeY sizeZ [stepSizeX stepSizeY stepSizeZ] [spacingX spacingY spacingZ].

4.2.2. Реалізація алгоритму Poisson

Реалізація даного алгоритму була виконана у 2006му році Михайлом Кажданом та Метью Боліто. Програма написана на мові C++. З часом програма вдосконалювалася її розробниками та зараз налічує вже більше ніж 8 версій. Вихідний код програми викладений на сайті Jonh Hopkins University у розділі, що присвячений Михайлу Каждану за адресою http://www.cs.jhu.edu/~misha/. В дипломній роботі розглядається версія 8.0, яка є найновішою на даний момент.

Програма Screened Poisson Surface Reconstruction (в подальшому буде використовуватися абревіатура SPSR) має зручне управління через консоль. Це є великою перевагою, адже користувачеві не потрібно нічного додатково встановлювати, достатньо лише зайти в консоль та прописати необхідні команди. SPSR має змогу обробляти файли формату .ply, .bnpts або ascii файл з координатами точок. Кожному типу файла наданий короткий опис вимог до того, яким чином мають бути впорядковані координати точок.

Всього користувач може задати 21 параметр. Основними параметрами є адреси вхідного та вихідного файлів, які потрібно прописувати в консолі після спеціальних ідентифікаторів --in та --out.

Приклад запуску програми:

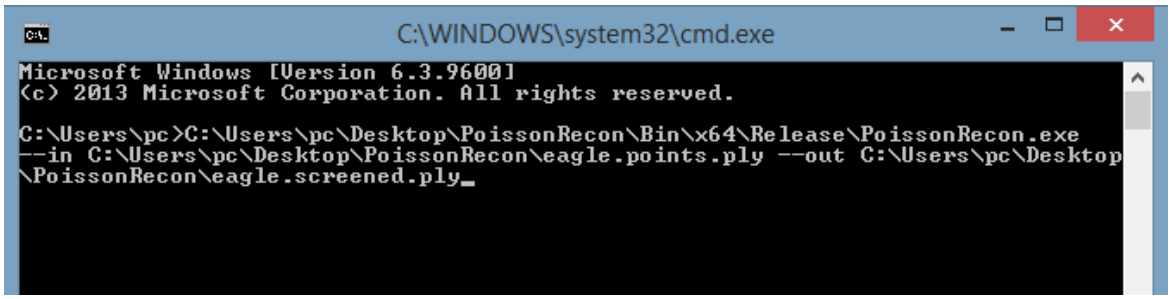

Рисунок 4.1 – Приклад команди для запуску програми

Першою строкою є повний шлях до виконуючого файлу - PoissonRecon.exe. Після цього написаний ідентифікатор --in , який вказує на те, що після нього прописана адреса до файлу з набором точок, з цього файлу буде відбуватися зчитування даних. Далі ідентифікатор --out вказує на те, що після нього вказана адреса файлу, в який буде записуватися результат. Вихідні дані записуються лише у формат .ply.

Список інших ідентифікаторів, з допомогою яких можна вказати додаткові параметри роботи програми:

- --linearFit

  Включення цієї команди дозволяє програмі використовувати лінійну інтерполяцію для оцінки положень ізо-вершин.
- --degree <B-spline degree>

  На вхід приймає число, що дорівнює значенню B-сплайну, яке буде використовуватися для визначення системи кінцевих елементів. Чим більше значення, тим вищий буде порядок апроксимації. Але така

перевага буде досягатися за рахунок використання додаткового часу та пам’яті (через використання більш щільних системних матриць).

За замовченням значення параметру дорівнює 2.

x --color <pull factor>

Якщо значення вказане, то реконструкція коду передбачає, що вхідні дані подаються разом зі значеннями кольорів і програма екстраполює кольори до вершин реконструйованої сітки.

x --voxel <output voxel grid>

Цей ідентифікатор означає, що після нього йде строка, яка визначає ім'я файлу, до якого буде записана вибірка неявної функції. Файл буде записаний бінарним кодом, перші чотири байти якого відповідають за роздільну здатність вібірки, решта 4 x 2^d x 2^d x 2^d байтів відповідають за значення неявної функції.

x --depth <reconstruction depth>

Після цього ідентифікатору йде число, яке означає максимальну глибину дерева, яке буде використано для реконструкції поверхні. Вказане значення глибини d означає, що роздільна здатність реконструкції на воксельній сітці не буде більшим ніж 2^d x 2^d x 2^d.

Треба зазначити, що поки реконструкція не адаптувалася до дерева октантів щільності вибірки, задана глибина реканструкції є лише верхньою межею.

За замовченням, значення параметру дорівнює 8.

x --fullDepth <adaptive octree depth>

Після цього ідентифікатору йде число, яке визначає межу, над якою глибиною дерево октантів буде адаптовано. На більшій глибині октодерево буде завершеним та буде містити 2^d x 2^d x 2^d вузлів.

За замовченням, значення параметру дорівнює 5.

x --voxelDepth <voxel sampling depth>

Після цього ідентифікатору йде число, яке визначає глибину звичайної сітки, над якою неявна функція проводить вибірку. Вказане значення глибини d означає, що роздільна здатність реконструкції на воксельній сітці не буде більшим ніж 2^d x 2^d x 2^d.

За замовченням, значення параметру дорівнює значенню параметра після ідентифікатора --depth.

x --cgDepth <conjugate gradients solver depth>

Після цього ідентифікатору йде число, яке визначає глибину до якої рішення сполучених градієнтів будуть використовуватися для рішення лінійної системи. Поза цієї глибини буде використовуватися релаксація Гауса-Зейделя.

За замовченням, значення параметру дорівнює 0.

x --scale <scale factor>

Після цього ідентифікатору йде число, яке визначає співвідношення між діаметром кубу, що використовується для реконтрукції та діаметром кубу, який обмежує зразки.

За замовченням, значення параметру дорівнює 1.1.

Список інших параметрів, які користувач може вказати:

x --primalVoxel

x --samplesPerNode <minimum number of samples>

x --pointWeight <interpolation weight>

x --iters <GS iters>

x --threads <number of processing threads>

x --confidence

x --nWeights

x --polygonMesh

x --density

x --verbose

Як приклад роботи програми, була завантажена з інтернету вибірка точок, що складають собою фігуру орла. Після чого, ця вибірка точок була дана програмі для подальшої її обробки.

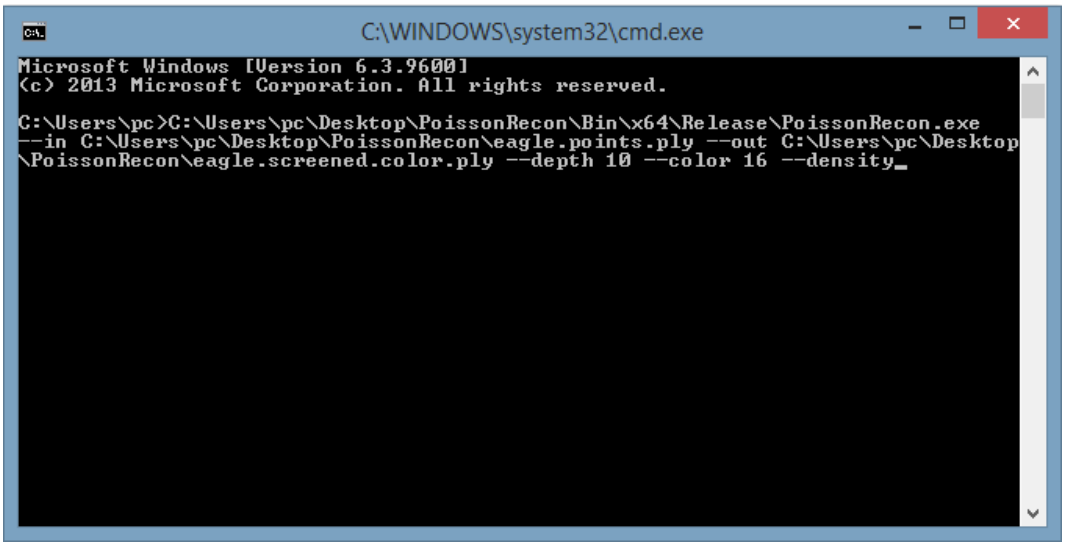

Рисунок 4.2 – Приклад команди для виконання програми

Команда для обробки файлу складалася з таких компонентів:

- C:\Users\pc\Desktop\PoissonRecon\Bin\x64\Release\PoissonRecon.exe – адреса виконуючого файлу програми;
- --in C:\Users\pc\Desktop\PoissonRecon\eagle.points.ply – адреса файлу, з якого будуть зчитуватися дані;
- --out C:\Users\pc\Desktop\PoissonRecon\eagle.screened.color.ply – адреса файлу, куди буде відбуватися запис оброблених даних;
- --depth 10 – глибина вибірки рівна 10
- --color 16 – показник того, що буде враховуватися колір фігури
- --density – показник щільності моделі

В результаті перетворення можна спостерігати об’ємну модель орла. Перегляд об’ємної моделі для зручності був зроблений за допомогою програми MeshLab.

Рисунок 4.3 – Об’ємна модель орла

Для удостовірності правильної роботи програми, виконаємо ще раз обробку даних, але трохи змінивши команди.

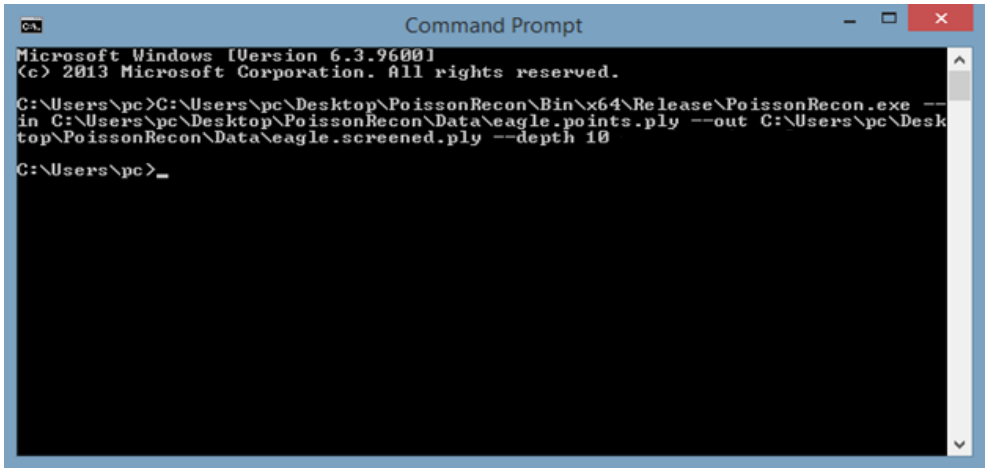

Рисунок 4.4 - Приклад команди для виконання програми

Команда для обробки файлу складалася з таких компонентів:

- C:\Users\pc\Desktop\PoissonRecon\Bin\x64\Release\PoissonRecon.exe – адреса виконуючого файлу програми;

- --in C:\Users\pc\Desktop\PoissonRecon\eagle.points.ply – адреса файлу, з якого будуть зчитуватися дані;
- --out C:\Users\pc\Desktop\PoissonRecon\eagle.screened.ply – адреса файлу, куди буде відбуватися запис оброблених даних;
- --depth 10 – глибина вибірки рівна 10

Очевидно, що цьому випадку модель орла буде виглядати інакше, адже інформації про колір не було додано.

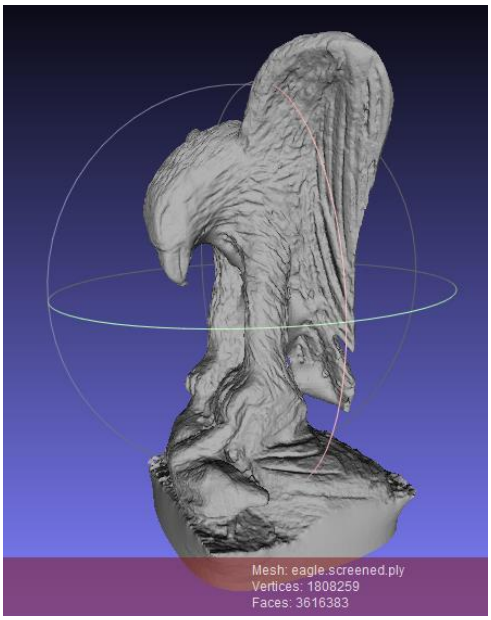

Рисунок 4.5 – Об’ємна модель орла

4.3. Висновки до розділу 4

В даному розділі розглядалися різні програмні продукти та бібліотеки, які реалізовують розглянуті в попередньому розділі методи та алгоритми. Крім цього, в розділі були наведені деякі результати тестування, а саме модель стенфордського зайця, яка була реконструйована з допомогою алгоритму обертових куль, а також модель орла, яка була реконструйована з допомогою алгоритму Пуассона. Якість моделі зайця виявилася набагато меншою, ніж якість моделі орла в зв’язку з особливостями алгоритмів.

5. ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПОРІВНЯННЯ ТОЧНОСТІ МОДЕЛЮВАННЯ ПОВЕРХНІ

5.1. Особливості програмної реалізації

Програмна реалізація першого методу для порівняння точності моделювання поверхні виконується на мові програмування C++ та с допомогою середи розробки Visual Studio 2012. В даній роботі приводиться тільки опис методу для того, щоб проілюстувати його покроковий принцип роботи.

Метод приймає на вхід такі дані:

1. Математичне рівняння досліджуваної фігури
2. Файл з розширенням .ply, який утворюється в ході реконструюваня поверхні алгоритмами та методами.

Метод подає на вихід:

1. Побудований графік з кривою похибки методу

Порядок дій методу:

1. Розглядається файл з вихідними даними алгоритму.
2. Генерується набір з n точок фігури.
3. Для кожної точки обчислюються координати по кожній з 3х осей (x, y, z) і заносяться в 3 списка. Це контрольні значення.
4. Поки не закінчиться набір даних зі списка з координатами точок по осі x, беремо кожну координату і з допомогою файлу з вихідними даними обчислюємо координати точки по осях y та z.
5. Вираховуємо різницю між контрольним значенням точки по осі y та z і між значеннями, вирахуваним у п.4. Заносимо у таблицю похибок.

6. Для тих самих точок з набору контрольних значень повторюємо кроки 4-5, але вже виходячи з того, що нам відомі координати значень по осях y та z.
7. Будуємо графік, який відображає значення похибок.

5.2. Висновки до розділу 5

В ході даного розділу були описані базові вимоги до програмної реалізації методів порівняння точності моделювання поверхні – це мова програмування C++ та середа розробки Visual Studio 2012. Такі вимоги обумовлені тим, що візуалізаційні властивості мови програмування C++ доволі розвинуті та пристосовані до обробки графічних даних. Крім цього, в ході даного розділу було покроково описано реалізацію одного з методів порівняння точності моделювання поверхонь. Для опису був обраний саме метод з порівнянням координат точок на контрольній моделі та на реконструйованій поверхні.

6. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

6.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

передбачати мінімальні витрати на впровадження програмного продукту.

6.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування, F_2 – вибір оптимального середовища розробки, F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Java;

б) мова програмування C++.

Функція F_2:

а) IntelliJ Idea;

б) Visual Studio.

Функція F_3:

а) графічний інтерфейс додатку;

б) консольний інтерфейс додатку.

6.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (Рисунок 6.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 6.1).

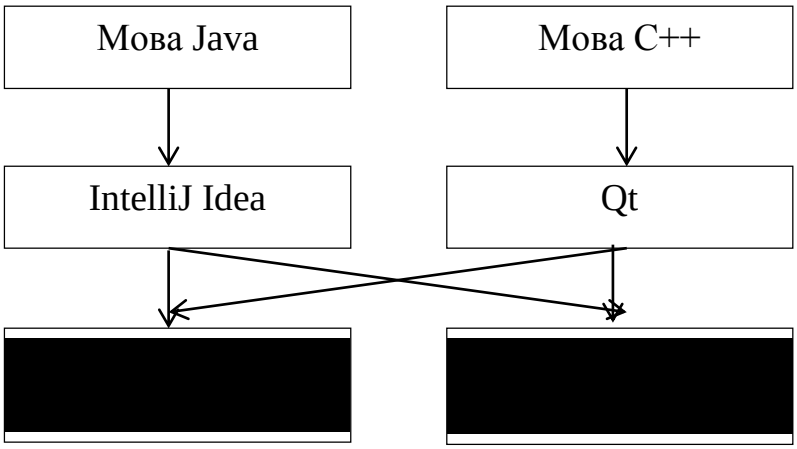

Рисунок 6.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 6.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Простота в реалізації, але менший набір вбудованих функцій	Менший набір специфічних вбудованих функцій
	Б	Швидкодія	Складний для програмування
F2	А	Стабільність та кросплатформеність	Довготривала реалізація графічного інтерфейсу
	Б	Простіша реалізація графічного інтерфейсу	Велика кількість помилок, нестабільність роботи
F3	*А*	Надзвичайна простота створення	Не 'user-friendly'
	Б	Зручність для пересічного користувача	Складніша реалізація, повільніше працює

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки в рамках даної задачі швидкодія C++ буде не настільки значною, щоб можна було заради цього знехтувати складністю її реалізації обираємо варіант А.

Функція *F2:*

Оскільки програма має бути стабільною та працювати на різних платформах, то для цього обираємо варіант А.

Функція *F3:*

Оскільки, щодо інтерфейсу програмного продукту не має вишуканих вимог (ПП є виключно робочою частиною, в подальшому планується абсолютна автоматизація), то обидва варіанти А і Б влаштовують.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1A – F2A – F3A
2. F1A – F2A – F3Б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

6.2 Обґрунтування системи параметрів ПП

6.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X1 – швидкодія мови програмування;

X2 – час обробки даних;

X3 – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає час, який витрачається на дії.

X3: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

6.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 6.2.

Таблиця 6.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Час обробки запитів користувача	X2	мс	1000	420	60
Потенційний об’єм програмного коду	X3	кількість строк коду	2000	1500	1000

За даними таблиці 6.2 будуються графічні характеристики параметрів – рис. 6.2 – рис. 6.4.

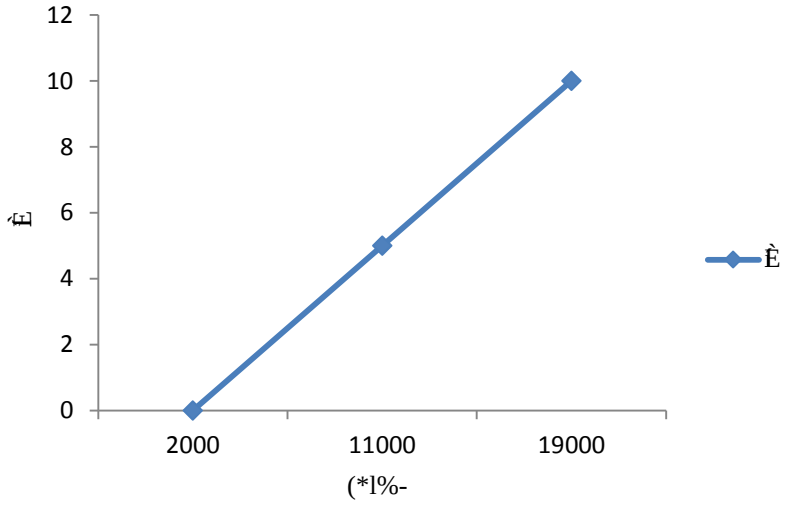

Рисунок 6.2 – X1, швидкодія мови програмування

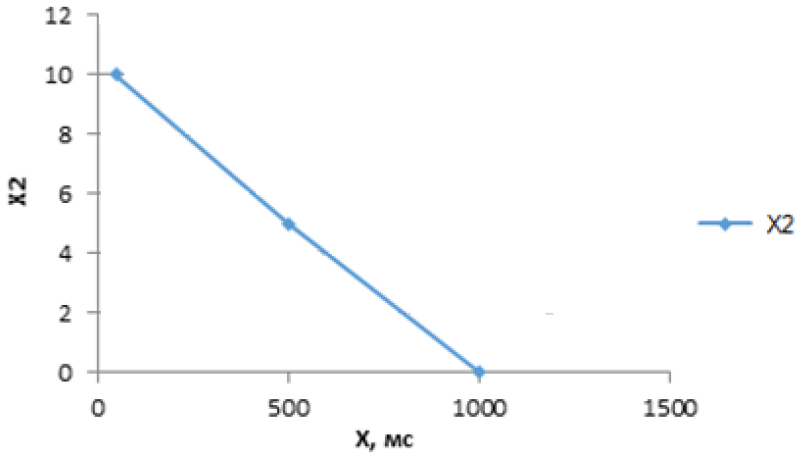

Рисунок 6.3 – X2, час виконання запитів користувача

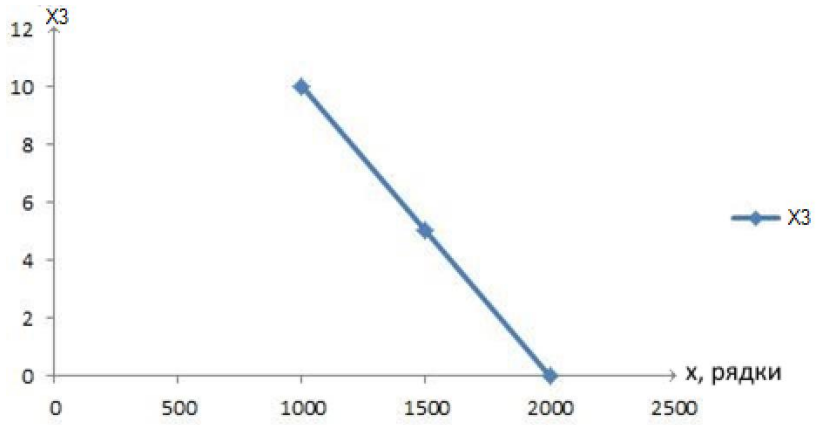

Рисунок 6.4 – X3, потенційний об’єм програмного коду

6.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень. Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

визначення рівня значимості параметра шляхом присвоєння різних рангів;

перевірку придатності експертних оцінок для подальшого використання;

визначення оцінки попарного пріоритету параметрів;

обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 6.3.

Таблиця 6.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,33	0,109
X2	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,67	215,2
X3	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,33	214,92
	Разом		11	11	11	11	11	11	11	80	0	430,229

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 80,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}67$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 430{,}229.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 430{,}229}{7^2(3^3 - 3)} = 4{,}39 > W_{\text{норм}} = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 6.4.

Таблиця 6.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{ де } b_i = \sum_{i=1}^{N} a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j.$$

Як видно з таблиці 6.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 6.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j			Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1,0	1,5	0,5	3.0	0.333	8.0	0.32	21.25	0.312
X2	0,5	1,0	0,5	2.0	0.222	5.5	0.22	15.25	0.223
X3	1,5	1,5	1,0	4.0	0. 445	11.5	0.46	31.75	0.465
Всього:				9	1	25	1	68.25	1

6.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 6.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів; $K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 6.6 – Розрахунок коефіцієнтів технічного рівня варіантів реалізації основних функцій ПП

Параметри x_i	Параметри x_j			Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,5	0,5	3.0	0.333	8.0	0.32	21.25	0.312
X2	0,5	1,0	0,5	2.0	0.222	5.5	0.22	15.25	0.223
X3	1,5	1,5	1,0	4.0	0. 445	11.5	0.46	31.75	0.465
Всього:				9	1	25	1	68.25	1

Таблиця 6.7 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри x_i	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	а)	Xl	11000	7	0,312	2,1
		X3	1200	5	0,465	2,3
F2	а)	X2	100	3	0,223	0,6
		X3	1200	7,5	0,465	3,4
F3	б)	X2	100	3,5	0,223	0,7
		X3	1700	1	0,465	0,4
	а)	X3	1200	1	0,465	0,4

За даними з таблиці 6.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$$К_{К1} = 2,184+2,325+0,669+3,488+0,78+0,465 = 9,911$$

$$К_{К2} = 2,184+2,325+0,669+3,488+0,465 = 9,131$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

6.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Але варіант II реалізації програмного забезпечення включає ще одне завдання:

3. Написання алгоритму збереження інформації у вигляді компонента, зручного для візуалізації.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б, завдання 3 до группи Г. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Завдання 3 відноситься за складністю до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$Т_О = Т_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \qquad (6.1)$$

де $Т_Р$ – трудомісткість розробки ПП; $К_П$ – поправочний коефіцієнт; $К_{СК}$ – коефіцієнт на складність вхідної інформації; $К_М$ – коефіцієнт рівня мови програмування; $К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8.

Тоді, за формулою 6.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$Т_2 = 27 * 0.9 * 0.8 = 19.44$ людино-днів.

Для третього завдання (використовується алгоритм третьої групи складності, ступінь новизни Г):

$Т_р = 15$ людино-днів; $К_п = 0,6$; $К_{ст} = 1$; $Т_З = 15 * 0,6 * 1 = 9$.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I = (122.4 + 19.44) * 15 = 2127,6$ людино-годин;

$T_П$ = (122.4 + 19.44 + 9) * 15 = 2262,6 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 14000 грн., один спеціаліст по цифровій обробці сигналів з окладом 22000грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$С_Ч = \frac{14000 + 14000 + 22000}{3 \cdot 21 \cdot 15} = 52{,}91\ грн.$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = C_ч \cdot T_i \cdot К_Д,$$

де $С_Ч$ – величина погодинної оплати праці програміста; T_i – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 52,91· 2127,6 · 1.2 = 135085,58 грн.

II. $С_{ЗП}$ = 52,91· 2262,6 · 1.2 = 143657,00 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 135085,58 · 0.22 = 29718,8276 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 143657,00 · 0.22 = 31604,54 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 14000 грн., з коефіцієнтом зайнятості 0,33 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 12000 \cdot 0,33 = 47520$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 47520 \cdot (1 + 0.33) = 63201.6$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 63201,6 \cdot 0,22 = 13904,352$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,136 \cdot 0,33 \cdot 0,1506 = 11,533$ грн.,

де N_C – середньо-споживча потужність приладу; $K_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії (150.06 коп/1КВт).

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 63201.6 + 13904,352 + 2300 + 460 + 11,533 + 5360 = 85237,485$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 85237,485/1706,4 = 49.95$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 49.95 \cdot 2127,6 = 106273,62$ грн.;

II. $С_М = 49.95 \cdot 2262,6 = 113016,87$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 135085,58 \cdot 0,67 = 90507,34$ грн.;

II. $С_Н = 143657,00 \cdot 0,67 = 96250,19$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 135085,58 + 29718,8276 + 106273,62 + 90507,34 = 361585,3676$ грн.;

II. $С_{ПП}$ = 143657,00 + 31604,54 + 113016,87 + 96250,19 = 384528.6 грн.;

6.5 Вибір кращого варіанта ПП за техніко-економічним рівнем

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1}$ = 9,911/ 361585,3676 = $2,74 \cdot 10^{-5}$;

$К_{ТЕР2}$ = 9,131/ 384528.6 = $2,37 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 2,74 \cdot 10^{-5}$.

6.6 Висновки до розділу 6

В даній частині дипломної роботи було проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ $2{,}74 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Java
- використання середовища розробки Intellij Idea;
- консольний інтерфейс.

Даний варіант виконання програмного комплексу дає користувачу відмінний функціонал і швидкодію і робить простішим виконання завдання.

ВИСНОВКИ

Дипломна робота присвячена дослідженню методів та алгоритмів створення опису поверхні. В загальному випадку ці алгоритми і методи використовуються в області 3D сканування.

У результаті виконання дипломної роботи були:

1. Досліджені алгоритми та методи створення опису поверхні, такі як алгоритми обчислювальної геометрії:

   - триангуляція Делоне,
   - Альфа фігур (Alpha-shapes),
   - Оборотних куль (Ball Pivoting);

   Також були розглянуті такі алгоритмами з використанням неявних функцій:

   - Крокуючих кубів (Marching Cubes),
   - Метод Пуассона (Poisson),
   - Алгоритм Хоппа,
   - Алгоритм MPU.

2. Було запропоновано перевіряти точність моделювання поверхні двома методами. Перший метод заключається в тому, щоб звіряти координати точок на контрольній фігурі, а також координати точок на реконструованій поверхні. Другий метод полягає у порівнянні кута, який утворюється між дотичною і віссю координат у контрольній фігурі та у реконструйованій поверхні.

3. Було досліджено різноманітні бібліотеки та програмні реалізації алгоритмів та методів, такі як:

   - The Computational Geometry Algorithms Library(CGL Library),

   - Онлайн-ресурс IPOL Journal (Image Processing On Line),

- Програма Михайла Каждана
- Програма Еріка Містарда

В ході дослідження був зроблений висновок, що онлайн-ресурс IPOL Journal працює найповільніше. Це може бути зумовлено тим, що обробка даних відбувається на сервері, а не на комп’ютері користувача. Крім цього, алгоритм оборотних куль виявився єдиним, який може конструювати поверхню з дірками, що також негативно впливає на якість створення опису поверхні.

4. Спроектовано програмний засіб для порівняння точності моделювання поверхні.
5. Виконано економічний аналіз розробки.

Для подальшого розвитку необхідно удосконалити програмний засіб для порівняння точності моделювання поверхні.

Основна потенційна галузь застосування – 3D сканування. Використовуючи розробку можливо вибрати більш підходящий метод або алгоритм, а також програмний продукт під час моделювання об’ємної поверхні, в залежності від різних потреб.

СПИСОК ЛІТЕРАТУРИ

1. Миронов В. Биопечать вместо донорских органов / Миронов В. // Журнал «Наука и Жизнь», - 2013. - №11.
2. Онлайн-ресурс IPOL Journal (Image Processing On Line). – Режим доступа: http://www.ipol.im/pub/art/2014/81/. – Дата доступа: 10.05.2016.
3. Самарский А.А. Численные методы: Учебное пособие для вузов. / Самарский А.А., Гулин А.В. // Главная редакция физико-математической литературы, – М.: Наука, 1989 – 127 с.
4. Bernardini F. The Ball-Pivoting Algorithm for Surface Reconstruction / Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.
5. Goodman E. G. Handbook of Discrete and Computational Geometry. Second Edition. / Goodman E. G., O'Rourke J. // Chapman & Hall. CRC – 2004. ISBN 1-58488-301-4
6. Hjelle Ø. Triangulations and Applications / Hjelle Ø., Dæhlen M. // Springer – 2006. ISBN 978-3-540-33260-2
7. Cheng S.-W. Delaunay Mesh Generation / Cheng S.-W., Dey T. K., Shewchuk J.R. // CRC Press – 2013. ISBN 978-1-58488-731-7
8. Edelsbrunner H. Three-Dimensional Alpha Shapes / Edelsbrunner H., Mucke E. P. // ACM Transactions on Graphics, Vol. 13, - 1994. - №13.
9. Patent US 6968299 B1, Method and apparatus for reconstructing a surface using a ball-pivoting algorithm – Режим доступа: http://www.google.com/patents/US6968299. – Дата доступа: 14.05.2016.
10. Digne J. An Analysis and Implementation of a Parallel Ball Pivoting Algorithm – Режим доступа: http://www.ipol.im/pub/art/2014/81/article_lr.pdf. – Дата доступа: 17.05.2016.
11. Lorensen W. E. Marching Cubes: a high resolution 3D Sufrace reconstruction algorithm / Lorensen W. E., Harvey E.C. - 1987.

12. Kazhdan M. Poisson Surface Reconstruction / Kazhdan M., Bolitho M., Hoppe H. // Eurographics Symposium on Geometry Processing – 2006.
13. Hoppe H. Surface Reconstruction from Unorganized Points / Hoppe H. // University of Washington: a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy. – 1994.
14. Ohtake Y. Multi-level Partition of Unity Implicits / Ohtake Y., Belyaev A., Alexa M., Turk G., Seidel H.-P. // ACM SIGGRAPH – 2003.