ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Метод удосконалення параметрів FDM-друку на основі машинного зору»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Віталій ВОЛОШИН
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-63
Віталій ВОЛОШИН

Керівник: Віталій ЗАЛИВА
Старший викладач к-ф ІПЗ

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Волошину Віталію Віталійовичу

1. Тема кваліфікаційної роботи: «Метод удосконалення параметрів FDM-друку на основі машинного зору»

керівник кваліфікаційної роботи Віталій Залива, Старший викладач к-ф ІПЗ,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, FDM-друк, метод удосконалення параметрів, вимоги до якості друку.

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

1. Огляд літератури.
2. Аналіз технології FDM-друку.
3. Параметри FDM-друку які впливають на якість та сучасні методи їх оптимізації.
4. Розробка методу удосконалення параметрів FDM-друку.

5. Метод удосконалення параметрів FDM-друку.
6. Експериментальні результати.

5. Перелік ілюстративного матеріалу: *презентація*
   1. Аналіз технології FDM-друку.
   2. Проблематика FDM-друку та параметри, що впливають на якість.
   3. Метод удосконалення параметрів FDM-друку на основі машинного зору.
   4. Калібрування камери та позиціонування.
   5. Аналіз процесу 3D-друку на основі G-коду.
   6. Процес обробки зображень.
   7. Виявлення друкарських дефектів та удосконалення параметрів друку, для їх виправлення.
   8. Експериментальні результати.
   9. Порівняльний аналіз.

6. Дата видачі завдання«16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.10-23.10.2024	
2	Вивчення матеріалів для аналізу технології FDM-друку	23.10-30.10.2024	
3	Дослідження параметрів FDM-друку які впливають на якість та сучасні методи їх оптимізації	30.10-05.11.2024	
4	Розробка методу удосконалення параметрів FDM-друку	05.11-15.11.2024	
5	Постановка експеременту	15.11-25.11.2024	
6	Оформлення роботи: вступ, висновки, реферат	25.11-26.11.2024	
7	Розробка демонстраційних матеріалів	26.11-27.11.2024	
8	Попередній захист роботи	28.11.2024	

Здобувач вищої освіти _____ Віталій ВОЛОШИН
(підпис)

Керівник
кваліфікаційної роботи _____ Віталій ЗАЛИВА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 71с., 45 рис., 1 табл., 1 дод., 21 джерело.

Об`єкт дослідження – процес друку методом наплавлення розплавленого матеріалу (FDM) з використанням адитивних технологій.

Предмет дослідження – метод удосконалення параметрів FDM-друку на основі машинного зору.

Мета роботи – розробка методу удосконалення параметрів FDM-друку на основі машинного зору, що дозволить покращити якість друку та підвищити стабільність процесу.

Методи дослідження – включають аналіз та узагальнення літературних джерел з тематики FDM-друку та методів контролю якості, моделювання процесів оптимізації друку, розробку алгоритмів машинного зору для автоматичного визначення та корекції дефектів, а також експериментальну перевірку методу на практичних зразках.

У роботі розглянуто можливість інтеграції камери в систему FDM-друку, що пропонує рішення для раннього виявлення дефектів у процесі друку, що дозволяє проводити автоматизоване коригування параметрів друку в режимі реального часу. Камери дозволяють проводити моніторинг процесу на кожному етапі друку, аналізуючи шари матеріалу, їх розміщення та якість поверхні. Алгоритми обробки зображень дають можливість точно виявляти дефекти, прогнозувати можливі відхилення від норми та автоматично змінювати параметри друку, щоб уникнути погіршення якості виробу.

КЛЮЧОВІ СЛОВА: FDM-ДРУК, АДИТИВНІ ТЕХНОЛОГІЇ, ІНТЕГРАЦІЯ КАМЕРИ, ОБРОБКА ЗОБРАЖЕНЬ, КОНТРОЛЬ ЯКОСТІ, АВТОМАТИЗАЦІЯ, ДЕФЕКТИ ДРУКУ, 3D-ДРУК, УДОСКОНАЛЕННЯ ПАРАМЕТРІВ ДРУКУ.

ABSTRACT

Text part of the master's qualification work: 71 pages, 45 pictures, 1 table, 1 appendix, 21 sources.

Object of research – he process of printing by fused deposition modeling (FDM) using additive technologies.

Subject of research – the method of improving FDM printing parameters based on computer vision.

Purpose of the work – to develop a method for improving FDM printing parameters using computer vision to enhance print quality and increase process stability.

Research methods: include analyzing and summarizing literature on FDM printing and quality control methods, modeling the optimization processes of printing, developing computer vision algorithms for automatic detection and correction of defects, and experimentally validating the method on practical samples.

The work explores the integration of a camera into the FDM printing system, offering solutions for early defect detection during the printing process, enabling automated real-time adjustment of printing parameters. Cameras facilitate process monitoring at each stage of printing by analyzing material layers, their placement, and surface quality. Image processing algorithms enable accurate defect detection, prediction of potential deviations from the norm, and automatic parameter adjustments to prevent product quality deterioration.

KEYWORDS: FDM PRINTING, ADDITIVE TECHNOLOGIES, CAMERA INTEGRATION, IMAGE PROCESSING, QUALITY CONTROL, AUTOMATION, PRINTING DEFECTS, 3D PRINTING, PRINT PARAMETER OPTIMIZATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MTM – Multi-Template Matching

ICP – Iterative Closest Point

ГЗМ – Гауссова Змішана Модель

STL – Stereolithography

AHC – Agglomerative Hierarchical Clustering

ВСТУП

Технології адитивного виробництва, зокрема метод формування виробів шляхом моделювання методом наплавлення розплавленого матеріалу (Fused Deposition Modeling, FDM), за останнє десятиліття набули широкого застосування у різних галузях — від прототипування до серійного виробництва. Вони забезпечують високу гнучкість у створенні складних геометричних форм, зменшують виробничі витрати та скорочують час виготовлення прототипів. Проте, попри численні переваги, якість виробів, створених методом FDM-друку, досі залишається проблемним аспектом, оскільки на неї впливають численні параметри процесу, такі як швидкість друку, температура екструдера, товщина шару та інші. Неправильні налаштування цих параметрів можуть призвести до дефектів у виробах, таких як шаруватість, деформації, пористість та інші.

Одним із перспективних підходів для контролю якості друку є застосування систем машинного зору, які дозволяють здійснювати автоматизований аналіз виробів у реальному часі. Поєднання машинного зору з методами оптимізації параметрів друку відкриває можливості для створення систем, що здатні самостійно коригувати налаштування друку з метою досягнення максимальної якості кінцевого виробу. Це дозволить не лише зменшити кількість дефектів, але й оптимізувати витрати матеріалів та часу.

Метою даного дослідження є розробка методу удосконалення параметрів FDM-друку на основі машинного зору, що дозволить покращити якість друку та підвищити стабільність процесу.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- здійснити аналіз сучасних технологій FDM-друку та методів контролю якості за допомогою машинного зору

- визначити основні параметри, що впливають на якість друку, та дослідити можливі методи їх удосконалення;
- розробити алгоритм для аналізу якості друкованих виробів на основі зображень;
- провести експериментальну перевірку запропонованого методу та оцінити його ефективність.

Об’єкт дослідження — процес друку методом наплавлення розплавленого матеріалу (FDM) з використанням адитивних технологій.

Предмет дослідження — метод удосконалення параметрів FDM-друку на основі технологій машинного зору для покращення якості друкованих виробів.

Методологія дослідження включає аналіз та узагальнення літературних джерел з тематики FDM-друку та методів контролю якості, моделювання процесів оптимізації друку, розробку алгоритмів машинного зору для автоматичного визначення та корекції дефектів, а також експериментальну перевірку методу на практичних зразках. У процесі дослідження використовуються методи математичного моделювання, обробки зображень та машинного навчання для аналізу отриманих даних і оптимізації параметрів друку.

Практична значущість роботи полягає у створенні методу, який може бути інтегрований у FDM-принтери для автоматичного налаштування параметрів друку, що сприятиме зниженню витрат філаменту, підвищенню якості друку та зменшенню часу на налагодження процесу.

1 ОГЛЯД ЛІТЕРАТУРИ

S. Graves, Johann C. Rocholl (Rostock): Style Delta Robot Kinematics.[1]

У статті розглянуто кінематику дельта-роботів, які широко застосовуються у 3D-друці завдяки своїй високій швидкості та точності. Автори описують математичні моделі для аналізу рухів кінцівок роботів, що дозволяє більш ефективно програмувати їх для складних завдань, таких як нанесення шарів у FDM-друці. Цей підхід може бути корисним для оптимізації друкарських процесів через моделювання точних траєкторій.

Camera calibration with OpenCV: asymmetrical circle calibration pattern.[2]

У статті надається практичний посібник із калібрування камер за допомогою OpenCV із використанням асиметричних шаблонів кругів. Такий підхід забезпечує високу точність вимірювань, необхідних для аналізу геометрії об’єктів. Калібрування камери є критично важливим етапом для систем машинного бачення у 3D-друці, оскільки від нього залежить точність аналізу якості надрукованих моделей.

J. Heikkila, 2000: Geometric camera calibration using circular control points.[3]

Автор описує метод геометричної калібровки камер із використанням кругових контрольних точок. Це фундаментальний підхід для синхронізації зображень, отриманих з різних ракурсів, що є важливим для створення точних 3D-моделей. У контексті 3D-друку цей метод може допомогти у виявленні дефектів шляхом точного аналізу шарів.

OpenCV: Template Matching.[4]

Шаблонне співставлення (template matching) описує техніки знаходження об’єктів у зображенні за допомогою порівняння фрагментів. Алгоритми, зокрема Normalized Cross-Correlation, підходять для задач

локалізації об'єктів у машинному баченні. У 3D-друці ця методика може бути застосована для автоматичного розпізнавання дефектів чи перевірки відповідності друкованих шарів еталонним зображенням.

A. Crivellaro, V. Lepetit, 2014: Robust 3D Tracking with Descriptor Fields.[5]

У роботі представлено методи надійного трекінгу тривимірних об'єктів із використанням дескрипторів. Алгоритми ефективно визначають положення об'єкта в просторі навіть за умов часткової видимості або завад. Для 3D-друку це може бути корисним у відстеженні процесу друку та контролі якості у реальному часі.

J. Canny, 1986: A Computational Approach to Edge Detection.[6]

Класичний алгоритм Кенні для виявлення країв, який використовується для аналізу контурів об’єктів. У 3D-друці цей підхід дозволяє ідентифікувати межі друкованих шарів, виявляючи потенційні недоліки в геометрії.

Y. Chen, G. Medioni, 1992: Object modelling by registration of multiple range images.[7]

Стаття пропонує метод реєстрації декількох зображень для створення 3D-моделей об'єктів. Це дозволяє об’єднувати різні перспективи сканування для створення цілісної моделі, що може бути корисним у відтворенні деталей друку та перевірці точності процесу.

P.J. Besl, N.D. McKay, 1992: A Method for Registration of 3-D Shapes.[8]

Автори описують алгоритм Iterative Closest Point (ICP), який застосовується для вирівнювання тривимірних форм. Цей підхід широко використовується в 3D-друці для перевірки точності надрукованих моделей, зіставляючи їх із цифровими проєктами.

M. Varma, A. Zisserman, 2003: Texture classification: are filter banks necessary?[9]

Автори досліджують необхідність використання фільтрувальних банків для аналізу текстур. Робота підкреслює, як спрощені підходи можуть

забезпечити ефективну класифікацію текстур, що корисно для виявлення дефектів у 3D-друці, зокрема текстурних артефактів.

S.-C. Zhu et al., 2005: What are Textons?[10]

Робота аналізує текстони — базові елементи текстурного аналізу. Використання цього підходу може бути корисним для визначення поверхневих характеристик друкованих об’єктів.

L. Liu et al., 2019: From BoW to CNN: Two Decades of Texture Representation for Texture Classification.[11]

Стаття надає огляд еволюції методів аналізу текстур від традиційних підходів до використання глибоких нейронних мереж. Це дозволяє більш точно класифікувати текстури поверхонь 3D-друків та визначати потенційні дефекти.

2 АНАЛІЗ ТЕХНОЛОГІЇ FDM-ДРУКУ

2.1 Принципи роботи FDM-друку

Метод FDM-друку, або моделювання методом наплавлення розплавленого матеріалу, рис. 2.1, є одним з найпоширеніших способів адитивного виробництва.

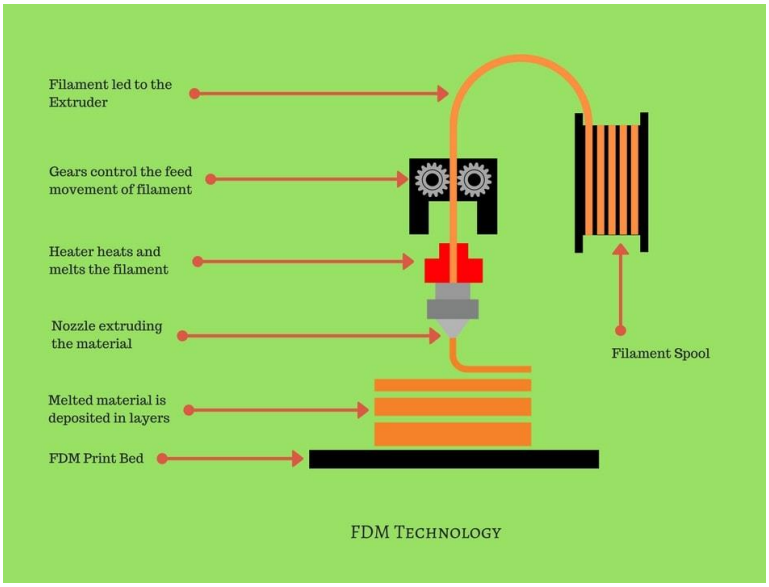

Рис. 2.1 – Схематичне зображення, що демонструє роботу технології 3D-друку

FDM-принтери, рис. 2.2 – це найчастіше декартові принтери. Декартова система координат - це система координат, яка використовується принтером для переміщення друкуючої головки та робочої пластини. У цих принтерах кожній осі (X, Y і Z) відповідають три рейки. Друкуюча головка (весь екструдер і блок сопел) рухається в напрямку X і Y, тоді як робоча платформа рухається в напрямку Z[1].

Спочатку в екструдер подається сировина, філамент. Нитки для FDM-принтерів доступні у двох різних діаметрах: 1,75 мм, що є найпоширенішим типом, та 3 мм. Конкретний принтер використовує тільки один тип філамента.

Шестеренчастий механізм в екструдері тягне нитку і штовхає її вниз до нагрівача, де вона розплавляється. Температура плавлення залежить від типу використовуваної нитки і зазвичай коливається від 190°Сдля PLA до 300°Сдля полікарбонату. Потім розплавлена нитка подається до сопла.

Сопла зазвичай доступні в двох діаметрах (0,2 і 0,4 мм), але є й інші розміри. Діаметр сопла може впливати на товщину шару і якість друку. Сопло наносить матеріал на робочу платформу у відповідній геометрії відповідно до моделі, яку потрібно надрукувати. Сопло рухається в напрямках X і Y і наносить перший шар відбитка.

Коли перший шар повністю нанесено, платформа опускається на невелику величину, що дорівнює товщині шару, і знову починається друк другого шару. Цього разу перший і другий шари все ще гарячі, і в результаті вони зливаються разом, утворюючи міцне з'єднання. Цей процес триває доти, доки не буде надруковано весь об'єкт.

Після друку об'єкт можна легко видалити вручну або за допомогою простого скребка. Деталі, надруковані методом FDM, мають видимі лінії шарів і, як правило, дуже грубі. Вони можуть бути додатково оброблені шліфуванням, поліруванням, ґрунтуванням і фарбуванням, згладжуванням парою ацетону, гальванічним покриттям тощо.

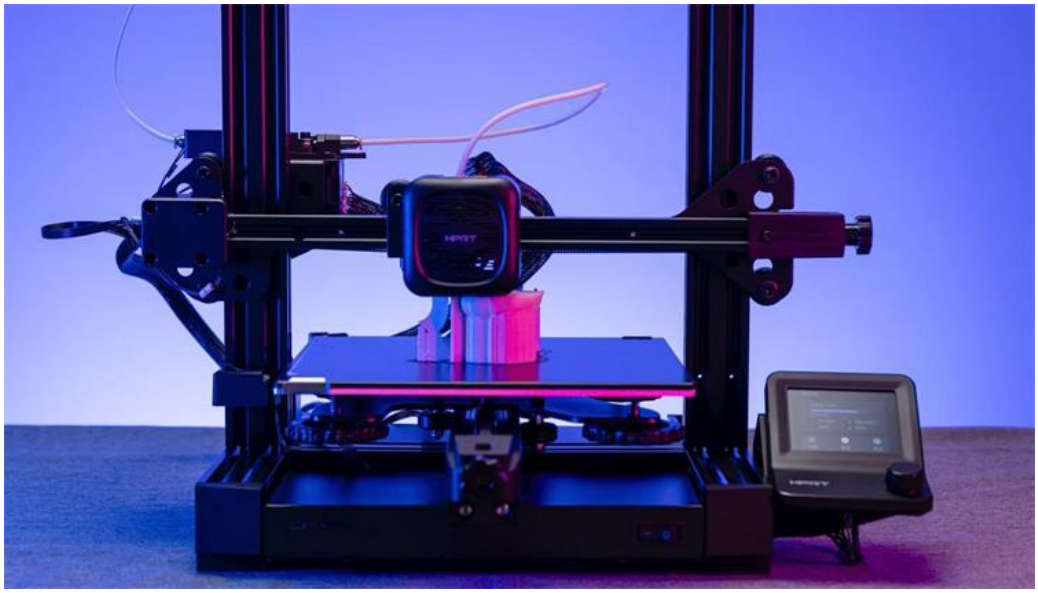

Рис. 2.2 – Приклад FDM-принтера

2.2 Компоненти FDM 3D-принтера

1. Рама (каркас), рис. 2.3 – Забезпечує стійкість і жорсткість конструкції принтера. Вона слугує основою для всіх інших компонентів, що сприяє зниженню вібрацій і покращенню точності друку.

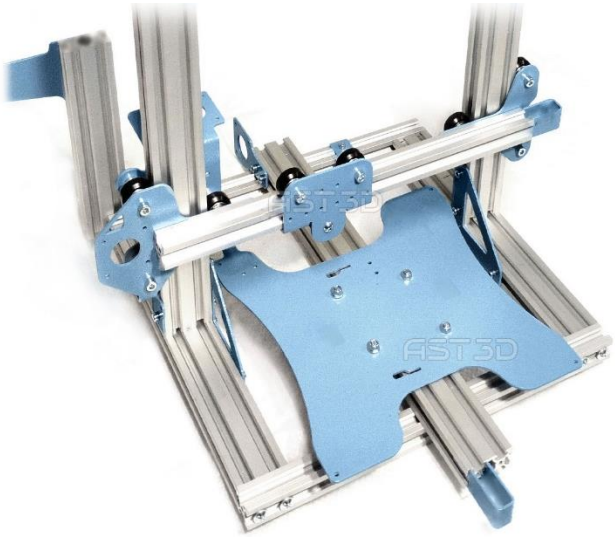

Рис. 2.3 – Рама FDM-принтера

2. Екструдер, рис. 2.4 – Складається з подаючого механізму і гарячого кінця (хотенду). Екструдер подає філамент у зону нагрівання, де він плавиться і через сопло наноситься на платформу.

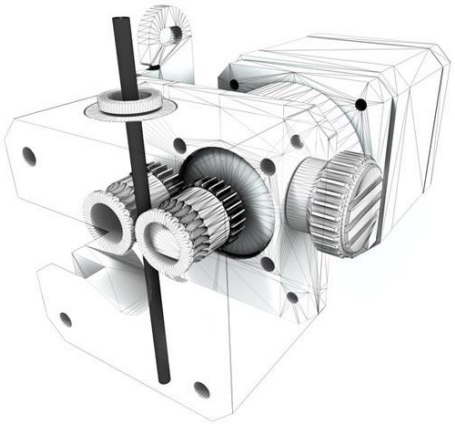

Рис. 2.4 – Екструдер FDM-принтера

3. Хотенд (гарячий кінець), рис. 2.5 – Це частина екструдера, де відбувається нагрівання філамента до необхідної температури. Важливі елементи хотенду включають: нагрівальний елемент, що відповідає за розігрів філамента до потрібної температури та термопара або терморезистор, який вимірює температуру для точного контролю процесу плавлення.

Рис. 2.5 – Хотенд FDM-принтера

4. Сопло, рис. 2.6 – Маленький отвір на кінці хотенду, через який розплавлений пластик видавлюється на платформу. Розмір сопла впливає на деталізацію та швидкість друку.

Рис. 2.6 – Сопло FDM-принтера

5. Платформа для друку (робочий стіл), рис. 2.7 – Поверхня, на яку наноситься розплавлений пластик. Багато моделей принтерів мають підігрівну

платформу, що сприяє кращому зчепленню першого шару і зменшує ризик деформації виробу під час друку.

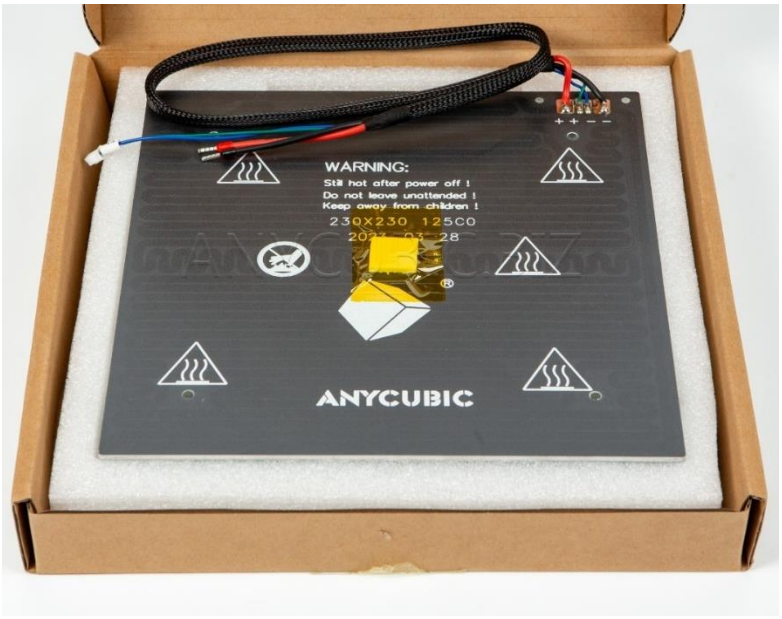

Рис. 2.7 – Платформа для друку FDM-принтера

6. Двигуни (крокові двигуни), рис.2.8 – Забезпечують переміщення екструдера і платформи по осях X, Y і Z. Крокові двигуни дозволяють точно контролювати положення екструдера та платформи, що забезпечує високу точність позиціонування під час друку.

Рис. 2.8 – Крокковий двигун FDM-принтера

7. Ремені, рис.2.9 – Вони забезпечують рух екструдера і платформи по відповідних осях. Ремені використовуються для плавного руху в осях X і Y, а направляючі і стержні дозволяють переміщення по осі Z.

Рис. 2.9 – Ремень FDM-принтера

8. Контролер (плата керування), рис.2.10 - Основний "мозок" принтера, що керує усіма процесами. Контролер обробляє дані від 3D-моделі та координує рух двигунів, подачу філамента, нагрівання хотенду та платформи, а також інші функції[2].

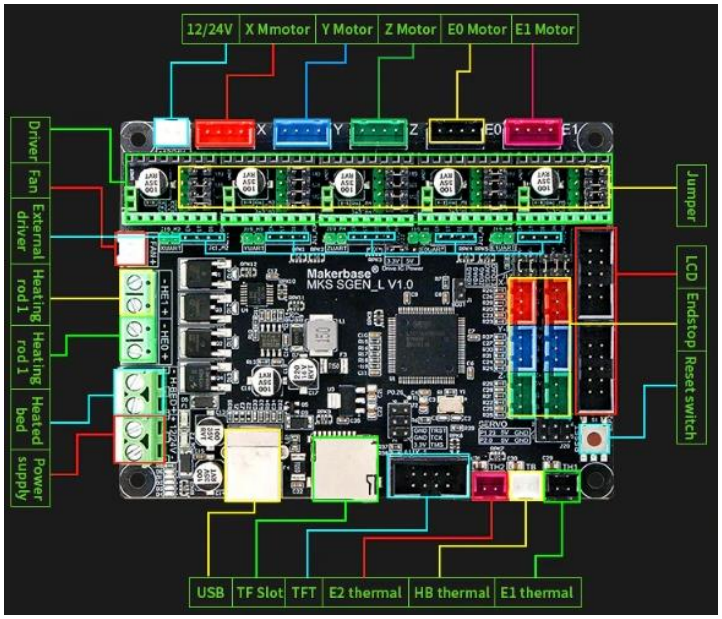

Рис. 2.10 – Контролер FDM-принтера

9. Дисплей і/або панель управління, рис. 2.11 - Дозволяє взаємодіяти з принтером, змінювати налаштування, запускати, призупиняти та завершувати процес друку.

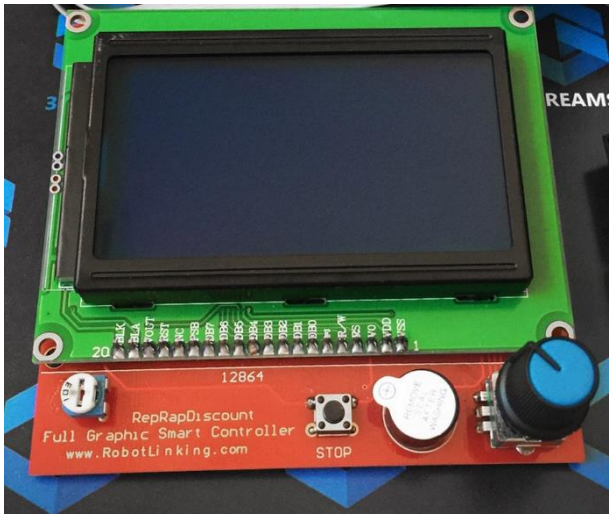

Рис. 2.11 – Дисплей і/або панель управління FDM-принтера

10. Система охолодження, рис. 2.12 – Може включати вентилятори для охолодження екструдера, сопла та/або самого виробу. Охолодження покращує якість друку, особливо для дрібних деталей і тонких шарів, запобігаючи перегріванню філамента після його нанесення.

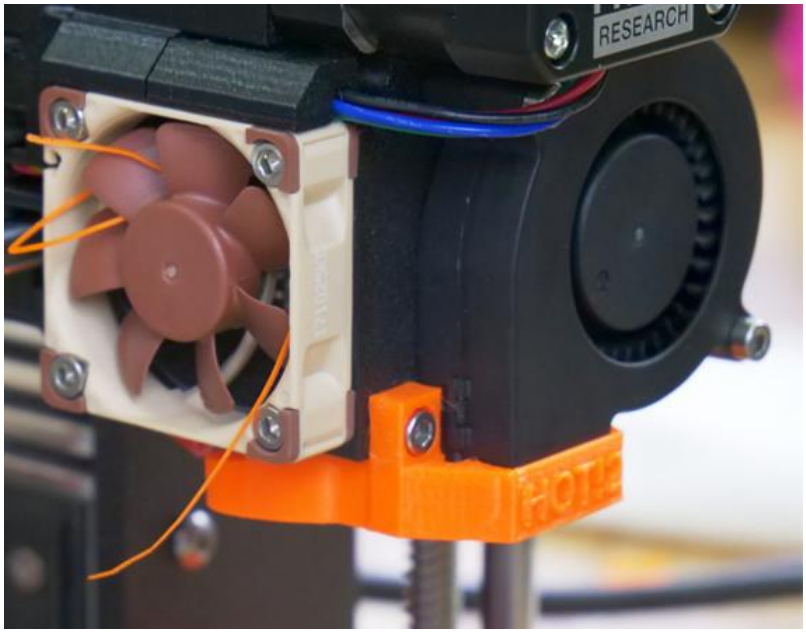

Рис. 2.12 Система охолодження FDM-принтера

2.3 Типи FDM 3D-принтерів

1. Cartesian-принтери, рис. 2.13, це найпоширеніший тип 3D-принтерів, що використовуються для виготовлення об'єктів методом пошарового нанесення матеріалу. Назва "Cartesian" походить від координатної системи Декарта, яку ці принтери використовують для переміщення робочого інструмента в трьох вимірах: X, Y і Z.

Принцип роботи Cartesian-принтера базується на основі механічної системи, яка забезпечує точне переміщення друкуючої головки або платформи вздовж осей:

- **Ось X:** горизонтальне переміщення.
- **Ось Y:** рух вперед-назад.
- **Ось Z:** вертикальне переміщення для створення шарів.

Друкуюча головка (екструдер) або лазер рухається по цій координатній системі, наносить матеріал (пластик, метал, смолу тощо) шар за шаром, формуючи модель.

Основні елементи Cartesian-принтерів:

1. **Рама:** міцна основа, на якій встановлюються всі компоненти. Зазвичай виготовляється з алюмінію або сталі.

2. **Друкуюча платформа:** може бути нерухомою або рухатися вздовж осі Z або Y.

3. **Екструдер:** відповідає за нагрівання і подачу матеріалу (наприклад, PLA, ABS).

4. **Лінійні направляючі:** забезпечують точний рух по осях.

5. **Контролер:** електронний блок для керування рухом, температурою та іншими параметрами друку.

6. **Двигуни:** крокові двигуни, що забезпечують точне позиціонування всіх рухомих частин.

Переваги Cartesian-принтерів

1. **Простота конструкції:** дозволяє легко зрозуміти принцип роботи і налаштувати обладнання.

2. **Точність друку:** висока точність завдяки прямолінійному руху по осях.

3. **Широка доступність:** більшість доступних 3D-принтерів, зокрема популярні моделі (наприклад, Creality Ender-3, Prusa i3), побудовані за Cartesian-принципом.

4. **Можливість модифікації:** користувачі можуть додавати або змінювати компоненти.

Недоліки Cartesian-принтерів

1. **Обмеження швидкості:** через масивність конструкції занадто швидкий рух може призвести до втрати точності.

2. **Чутливість до калібрування:** для якісного друку потрібне ретельне налаштування осей і платформи.

3. **Великі розміри:** конструкція займає більше місця порівняно з деякими альтернативами (наприклад, Delta-принтерами).

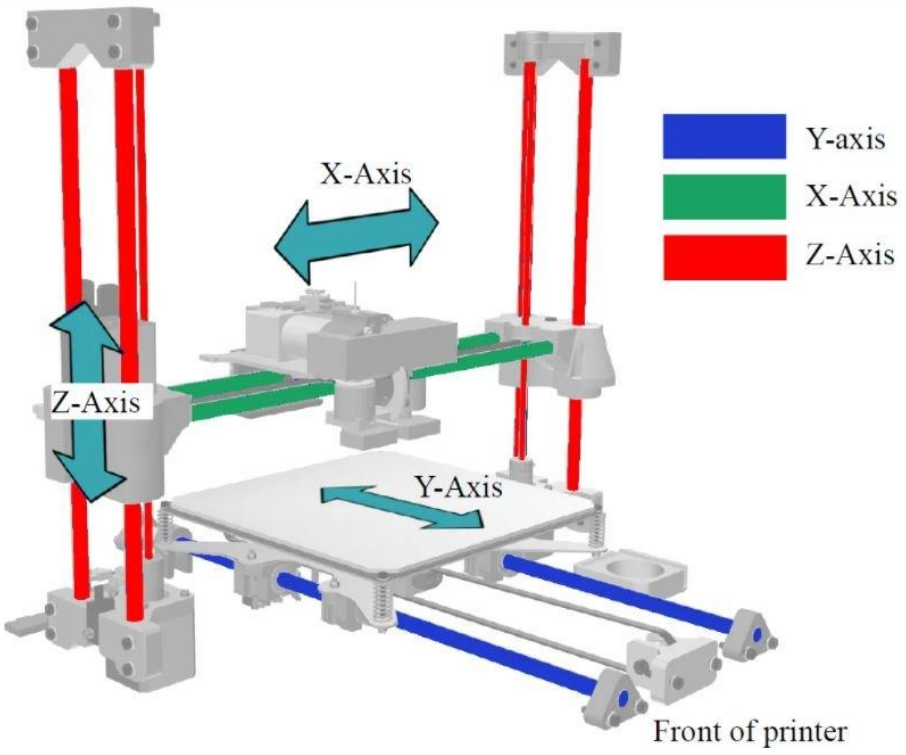

Рис. 2.13 – Схематичне зображення Cartesian-принтеру

2. Delta-принтери, рис. 2.14, це тип 3D-принтерів, які використовують тривісну кінематику для переміщення друкуючої головки. Їх конструкція базується на трьох вертикальних колонах (або напрямних), по яких рухаються каретки, під'єднані до платформи через паралельні шарнірні важелі. Цей тип принтерів ідеально підходить для задач, де потрібна висока швидкість друку та плавні переходи.

Принцип роботи Delta-принтера базується на тривимірній декартовій системі координат, але з унікальною кінематикою:

- Друкуюча головка підвішена на трьох незалежних важелях, які прикріплені до кареток.
- Каретки рухаються вгору-вниз вздовж вертикальних колон, що дозволяє змінювати положення головки.
- Завдяки синхронізації руху кареток принтер забезпечує плавний і швидкий рух друкуючої головки.

Цей підхід дозволяє друкувати складні геометричні форми з високою точністю.

Основні компоненти Delta-принтера:

1. **Вертикальні колони:** три стійки, які утримують рухомі каретки.
2. **Каретки:** рухаються вгору-вниз вздовж колон, забезпечуючи зміну положення друкуючої головки.
3. **Шарнірні важелі:** з'єднують каретки з платформою, забезпечуючи плавний рух.
4. **Друкуюча платформа:** зазвичай круглої форми, нерухома.
5. **Екструдер:** встановлений на кінці друкуючої головки.
6. **Контролер:** забезпечує синхронізацію рухів і контроль друку.
7. **Крокові двигуни:** керують каретками, забезпечуючи високу точність переміщення.

Переваги Delta-принтерів:

1. **Висока швидкість друку:** завдяки легкості конструкції друкуючої головки.

2. **Плавність руху:** забезпечує рівномірний розподіл матеріалу, що важливо для складних моделей.

3. **Простота друку високих об’єктів:** конструкція Delta-принтера дозволяє створювати об'єкти значної висоти.

4. **Менша маса рухомих частин:** зменшує інерцію і дозволяє зберігати високу якість друку на високих швидкостях.

Недоліки Delta-принтерів

1. **Складність калібрування:** для початківців налаштування може бути складним завданням.

2. **Менша стабільність:** при друку великих об'єктів можуть виникати проблеми з точністю через специфіку кінематики.

3. **Обмеження форми платформи:** через круглу друкарську платформу є обмеження на максимальні розміри моделі.

4. **Вища вартість:** складніша механіка і кінематика підвищують ціну порівняно з Cartesian-принтерами.

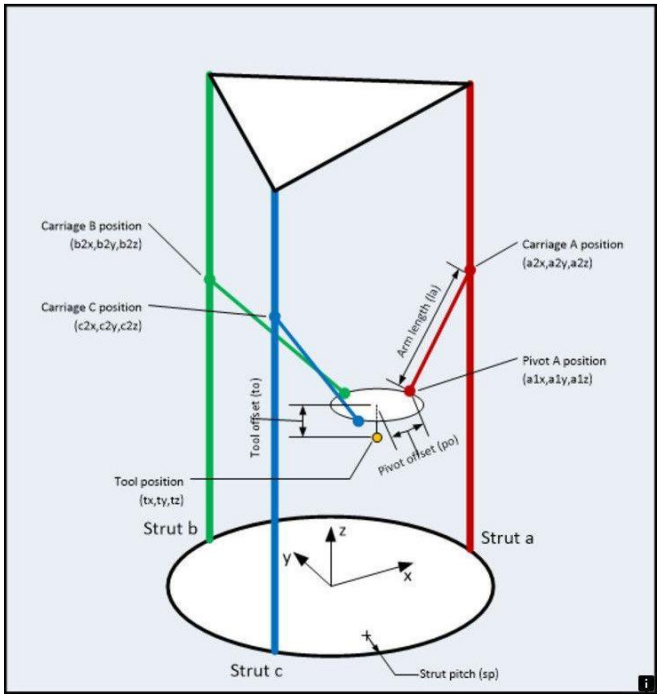

Рис. 2.14 – Схематичне зображення Delta-принтеру

3. CoreXY-принтери, рис. 2.15, це сучасний і високоефективний тип 3D-принтерів, який використовує унікальну кінематику для переміщення друкуючої головки. Завдяки своїй конструкції, ці принтери забезпечують високу швидкість, точність і стабільність друку, що робить їх популярними серед ентузіастів 3D-друку та професіоналів.

Принцип роботи CoreXY-принтера базується на спеціальній кінематичній схемі, яка передбачає використання двох ременів і двох крокових двигунів для переміщення друкуючої головки по осях X і Y:

- Два двигуни синхронно керують рухом друкуючої головки по горизонтальній площині.
- Для переміщення по осі Z зазвичай використовується окремий механізм (гвинтова пара або інша система).

Особливістю CoreXY є те, що двигуни залишаються нерухомими, зменшуючи вагу рухомих частин, що дозволяє досягти високої швидкості і точності.

Основні компоненти CoreXY-принтера:

1. **Жорстка рама:** зазвичай виготовляється з алюмінієвих профілів для забезпечення стабільності.
2. **Ременна система:** два ремені, що перетинаються, забезпечують рух друкуючої головки.
3. **Друкуюча платформа:** зазвичай рухається лише по осі Z, що зменшує вібрацію.
4. **Друкуюча головка:** легка і швидка, прикріплена до ременів.
5. **Крокові двигуни:** два двигуни для осей X і Y, окремий двигун для осі Z.
6. **Направляючі:** забезпечують плавність і точність руху друкуючої головки.
7. **Контролер:** управляє рухом, нагрівом і іншими параметрами друку.

Кінематика CoreXY має кілька унікальних властивостей:

- Обидва двигуни працюють одночасно для створення руху по осі X або Y.
- Повороти ременів дозволяють передавати рух без втрат, забезпечуючи точність.
- Нерухомі двигуни зменшують інерцію і покращують стабільність конструкції.

Переваги CoreXY-принтерів

1. **Висока швидкість друку:** завдяки легкості рухомих частин.
2. **Точність:** кінематика мінімізує помилки і забезпечує плавність руху.
3. **Компактність:** раціональна конструкція дозволяє створювати відносно невеликі принтери з великим робочим об’ємом.
4. **Мінімальна вібрація:** двигуни залишаються нерухомими, що зменшує вплив інерції на якість друку.
5. **Універсальність:** підходять для друку як великих, так і складних деталей.

Недоліки CoreXY-принтерів

1. **Складність налаштування:** правильне встановлення і калібрування ременів потребує досвіду.
2. **Вартість:** через складність механіки CoreXY-принтери можуть бути дорожчими за Cartesian-принтери.

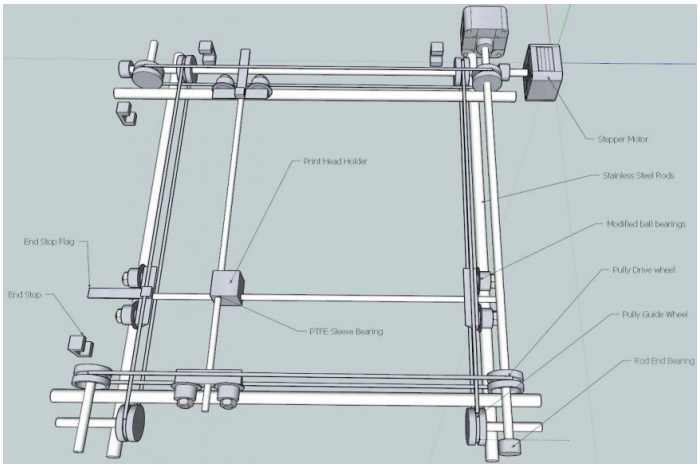

Рис. 2.15 – Схематичне зображення CoreXY -принтеру

2.4 Філамент і його види

Філамент — це спеціальний матеріал, який використовується у 3D-принтерах для створення об'єктів методом адитивного виробництва (3D-друку). Зазвичай це пластикова нитка або стрічка, яка подається в друкуючу головку принтера, де розплавляється та наноситься шар за шаром для створення моделі.

Основні характеристики філаменту

1. **Склад:** матеріал виготовляється з різних полімерів, до яких можуть додаватися домішки (металеві частинки, дерев'яні волокна тощо) для надання специфічних властивостей.
2. **Діаметр:** найпоширеніші розміри філаментів — **1.75 мм** і **2.85 мм** (іноді 3 мм).
3. **Температура плавлення:** кожен вид філаменту має свою температуру, при якій він розплавляється і стає придатним для друку.
4. **Механічні властивості:** міцність, гнучкість, жорсткість тощо залежать від типу матеріалу.

Види філаменту:

1. PLA, рис. 2.16 – це скорочення від полімолочної кислоти, термопластичного полімеру, отриманого з відновлюваних ресурсів, зокрема кукурудзяного крохмалю або цукрової тростини. Його можна переробляти на промислових підприємствах, але він не є біорозкладним.

Він має низьку температуру друку і не вимагає підігріву. Ще однією перевагою використання PLA є відсутність неприємного запаху під час друку (на відмінну ABS). Це потрібний матеріал для одноразового контакту з харчовими продуктами. Однак PLA менш довговічна, ніж ABS або PETG, і сприйнятливий до нагрівання. Не використовуйте його при виготовленні предметів, які можуть бути зігнуті, перекручені або неодноразово падати, наприклад, чохлів для телефонів, іграшок, що швидко зношуються, або ручок інструментів.

Рис. 2.16 – Приклад PLA філаменту

2. ABS, рис. 2.17, є найпоширенішим матеріалом для 3D-друку. Фактично він оточує нас у вигляді: чохлів для телефонів, велосипедних шоломів і навіть конструкторів LEGO. Завдяки своїй міцності та термостійкості цей пластик служитиме вам довго і не швидко зношуватиметься.

Хоча у такого типу нитки є багато переваг, варто пам'ятати та про особливості її використання. Наприклад, слід враховувати, що під час друку виділяє неприємний і шкідливий запах. Але це питання можна вирішити, якщо ви провітрюватимете кімнату, в якій працюватимете.

Рис. 2.17 – Приклад ABS філаменту

3. PETG, рис. 2.18 – це прозора нитка, яка може створити міцний та гладкий об'єкт. Її використовують, починаючи від садової техніки та закінчуючи простою пляшкою води. У цей пластик можна класти харчові продукти.

Варто пам'ятати, що такий матеріал гігроскопічний, тому щоб він не увібрав у себе вологу, його потрібно зберігати в сухому приміщенні.

Рис. 2.18 – Приклад PETG філаменту

4. TPE (термопластичні еластомери), рис 2.19, є класом матеріалів, які є сумішшю пластику і гуми. Вони включають TPU (термопластичний поліуретан), TPC (термопластичний сополіефір) та інші.

Ці пластмаси дуже м'які та гнучкі. Вони стають все більш поширеними в адитивному виробництві деталей, які можна згинати або розтягувати без будь-якої деформації. TPU також, як правило, більш довговічні та можуть забезпечити високу стійкість до стирання, масел, хімікатів, а також високих і низьких температур, ніж нитка TPE. TPC має стійкість до високих температур і відмінну стійкість до ультрафіолетового випромінювання. Він особливо цінується в медичних додатках, а також у портативних та пристроях. TPE також доступні у вигляді порошку та смоли.

Рис. 2.19 – Приклад TPE філаменту

5. Поліамід (PA), рис. 2.20, широко відомий як нейлон, є міцним і альтернативним матеріалом, що використовується для широких застосувань. Матеріал відрізняється своєю міцністю та стійкістю до високих температур та ударів. Це забезпечує хорошу міцність виробів та механічну міцність.

PA зазвичай армується вуглецевими, скляними та кевларовими волокнами. Він широко поширений у високотехнологічних інженерних застосуваннях, таких як шестерні, фітинги та інструменти. Також доступний у вигляді порошку.

Хоча він не буде друкувати так само легко, як PLA або PETG, вам може знадобитися високотемпературне сопло, оскільки для обробки деяких сумішей знадобиться температура до 300 ° C. Правильне зберігання нейлону також має вирішальне значення, оскільки він може вбирати воду якщо його залишити на відкритому повітрі. Ця волога погіршує якість матеріалу та призводить до погіршення якості та міцності друку.

Рис. 2.20 – Приклад PA філаменту

6. Матеріал під назвою акрилонітрилстиролакрилат (ASA), рис. 2.21, відомий високою ударною в'язкістю та хімічною стійкістю. Він також стійкий до ультрафіолетового випромінювання, зберігаючи свої властивості та зовнішній вигляд навіть за впливу сонячного світла, що робить його ідеальним для використання на відкритому повітрі.

Це простіший у пресі родич ABS. Але він також вимагає високих температур екструдера та столу, а також корпусу для протидії жолобленню, розтріскуванню та усадці. Це, звичайно, не для кожного принтера. З ним можуть впоратися потужніші настільні машини та, природно, промислові FDM.

Рис. 2.21 – Приклад ASA філаменту

7. PVB, рис. 2.22, скорочення від полівінілбутиралю, є спеціальною ниткою, яка найбільш відома своєю здатністю вирівнювати шари ізопропіловим спиртом. Це робить його безпечнішою альтернативою ABS, якому для вирівнювання шарів потрібен ацетон. Компанія Polymaker, наприклад, пропонує нитки PVB різних кольорів для обробки спиртової машини Polysher для досягнення рівномірних результатів. Крім того, це прозорий матеріал, тому можливі напівпрозорі та чіткі відбитки.

PVB має властивості матеріалу, аналогічні PLA, за винятком того, що у нього погана адгезія шару і, отже, найгірші механічні властивості, що,

безумовно, є недоліком. Також слідкуйте за тим, щоб нитка залишалася сухою, тому що він також має відносно вищі гігроскопічні властивості. Якщо ви хочете надрукувати гладкі моделі, варто звернути увагу на PVB.

Рис. 2.22 – Приклад PVB філаменту

8. HIPS, рис. 2.23 – поєднує в собі гнучкість та міцність. Він може тримати високі ударні навантаження та працює з багатьма видами клеїв. Також зручність у використанні полягає в тому, що вам не потрібен абразив, щоб усунути зайве. Цей матеріал легко забирається за допомогою розчину лимонену.

Рис. 2.23 – Приклад HIPS філаменту

9. PVA, рис. 2.24, цей пластик розчинний у воді. Його призначення полягає в тому, щоб користуватися ним як матеріалом, що підтримує, разом з іншою ниткою. Також слід бути обережним при зберіганні, оскільки вологе повітря може завдати йому шкоди.

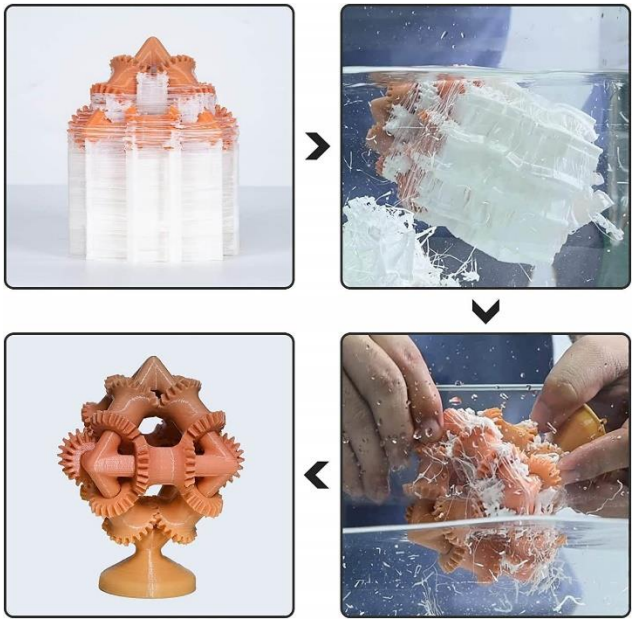

Рис. 2.24 – Приклад PVA філаменту

3 ПАРАМЕТРИ FDM-ДРУКУ ЯКІ ВПЛИВАЮТЬ НА ЯКІСТЬ ТА СУЧАСНІ МЕТОДИ ЇХ ОПТИМІЗАЦІЇ

3.1 Огляд параметрів друку, що впливають на якість

На якість друкованих виробів впливає низка параметрів FDM-друку, які можна розділити на дві групи: основні та допоміжні. До основних параметрів належать:

- **Температура екструдера** — визначає швидкість розплавлення матеріалу та його адгезію до попередніх шарів. Занадто низька температура може призвести до недостатнього[3] розплавлення пластику та поганому зчепленню шарів. Занадто висока температура може спричинити перегрівання пластику та деформацію деталі, рис.3.1.

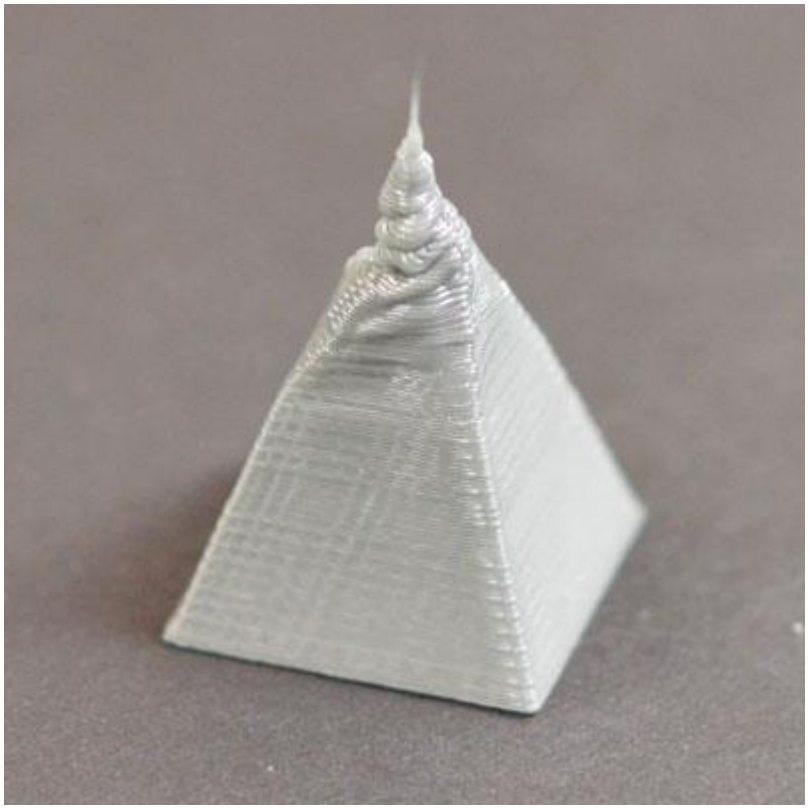

Рис. 3.1 – Дефект друку при виборі неправильної температури екструдера

- **Швидкість друку** — впливає на точність та рівномірність шарів. Висока швидкість може призвести до нерівномірного розподілу матеріалу та утворення ниток, рис.3.2. Занадто низька швидкість збільшує час друку, але може покращити якість поверхні.

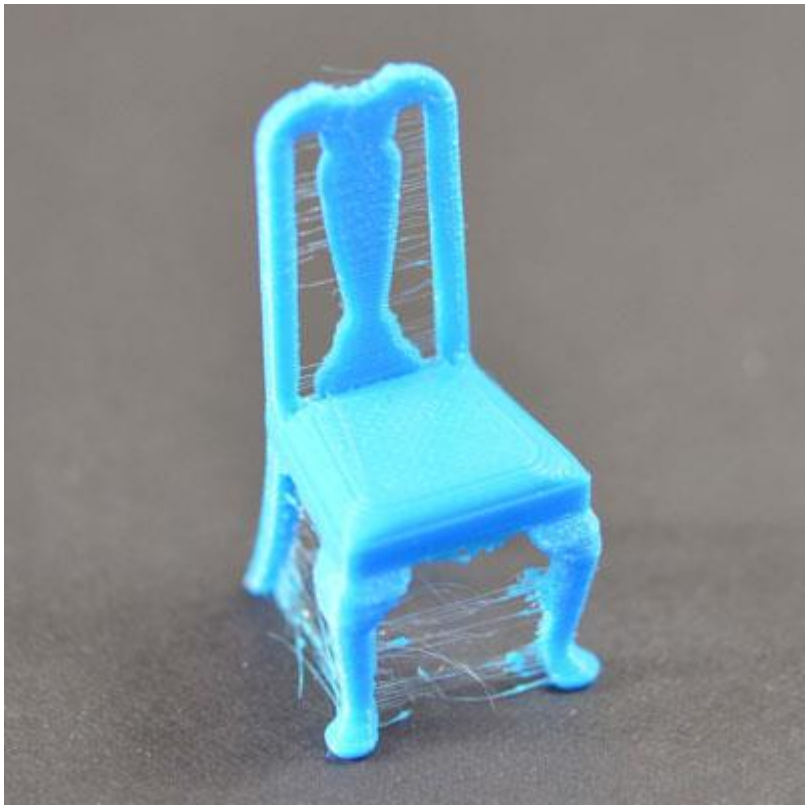

Рис. 3.2 – Дефект друку при виборі неправильної швидкості друку

- **Заповнення моделі** — щільність внутрішніх структур моделі, що впливає на міцність та вагу, рис.3.3.

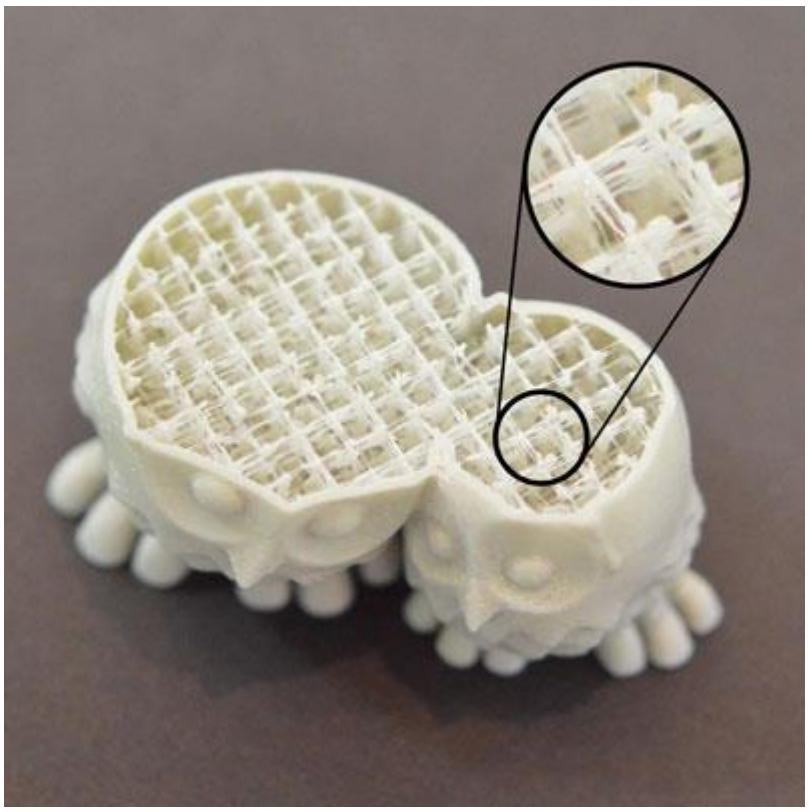

Рис. 3.3 – Дефект друку при виборі неправильної заповненості моделі

Допоміжні параметри, такі як температура платформи, швидкість вентилятора, вибір матеріалу тощо, також суттєво впливають на кінцевий результат друку. Налаштування цих параметрів має значення для запобігання дефектам, наприклад, розшаруванню, деформаціям чи підвищеній пористості.

3.2 Сучасні методи удоскалення параметрів друку

Розглянемо сучасні методи оптимізації параметрів друку в технології FDM, що включають підходи для підвищення якості та продуктивності друкованих виробів за рахунок оптимізації налаштувань друку, контролю та автоматизації процесу.

Моделювання та симуляція параметрів друку. Для досягнення оптимальної якості виробів часто використовуються інструменти симуляції, які дозволяють віртуально налаштувати і перевірити параметри друку до початку процесу. Поширені програмні рішення, такі як ANSYS, Simufact та

Autodesk Netfabb, дозволяють моделювати поведінку матеріалу при друці, відстежувати можливі деформації та визначати, як різні налаштування вплинуть на кінцеву якість виробу. Це дає змогу уникнути проб і помилок на реальних матеріалах і економити час та витратні матеріали.

Методи машинного навчання для прогнозування дефектів. Методи машинного навчання активно використовуються для аналізу та прогнозування якості виробів під час друку. Алгоритми, зокрема нейронні мережі та машинне навчання на основі дерев рішень, навчаються на великій кількості зразків, що дозволяє їм передбачати появу дефектів у режимі реального часу. Наприклад, система може ідентифікувати ознаки таких дефектів, як розшарування чи деформація, і автоматично скоригувати параметри друку (температуру, швидкість подачі філамента) для запобігання їхньому виникненню[4].

Оптимізація параметрів за допомогою генетичних алгоритмів. Генетичні алгоритми (ГА) використовуються для визначення оптимальних налаштувань параметрів друку шляхом автоматизованого тестування різних комбінацій і відбору кращих варіантів. Генетичні алгоритми функціонують за принципом природного відбору: вони генерують «популяцію» варіантів параметрів, «схрещують» найкращі рішення і «мутують» їх, щоб знайти оптимальні параметри, які забезпечують найвищу якість і міцність виробу.

Ітеративне налаштування за допомогою програмної оптимізації. Програмні інструменти оптимізації, такі як G-code оптимізатори, можуть автоматично генерувати більш ефективні траєкторії друку та коригувати параметри подачі матеріалу. Вони оптимізують рухи друкуючої головки для зменшення часу друку, одночасно зменшуючи напругу на екструдері та підвищуючи якість поверхні. Деякі програмні рішення також використовують зворотний зв’язок для аналізу попередніх результатів друку та налаштування параметрів для наступних завдань, що забезпечує поступове покращення якості.

Оптимізація на основі топологічного аналізу. Топологічний аналіз дозволяє оптимізувати структуру моделі для друку, що сприяє зменшенню

маси, покращенню міцності та ефективному використанню матеріалу. Використання цього методу дозволяє визначати оптимальні параметри для друку моделей із складною геометрією, мінімізуючи внутрішнє напруження матеріалу і знижуючи ризик дефектів під час друку.

Адаптивне налаштування висоти шару. Адаптивне налаштування висоти шару дозволяє змінювати товщину шару залежно від геометрії моделі, що дає змогу підвищити точність у деталях та зменшити час друку. Наприклад, при друку гладких поверхонь можна використовувати товстіші шари для швидшого результату, а для складних чи дрібних деталей – зменшувати висоту шару, що покращує деталізацію і точність.

Оптимізація з використанням фініт-елементного аналізу (FEA). Фініт-елементний аналіз дозволяє аналізувати і передбачати поведінку матеріалу під час друку, що допомагає уникати можливих дефектів, таких як деформації чи розшарування. Цей метод часто застосовується для оптимізації параметрів друку металевих деталей, де потрібно враховувати термічні напруження та інші механічні властивості матеріалу[5].

3.3 Використання машинного зору для оптимізаціїї параметрів друку

Впровадження машинного зору для оптимізації параметрів FDM-друку значно розширює можливості моніторингу та автоматизованого контролю якості виробів. Машинне бачення дозволяє детально відстежувати процес формування кожного шару, контролюючи такі важливі параметри, як однорідність, товщина шару, точність нанесення матеріалу та виявлення дефектів на поверхні виробу. Система аналізує зображення кожного шару, порівнюючи їх із цільовою моделлю, і в разі виявлення відхилень автоматично коригує параметри, зокрема швидкість друку та температуру, для забезпечення стабільності процесу.

Однією з важливих функцій машинного зору є виявлення дефектів і відхилень від стандартів якості. Технологія дозволяє ефективно виявляти такі дефекти, як розшарування, недостатнє або надмірне нанесення матеріалу, деформації та інші аномалії, що можуть виникати під час друку. Завдяки автоматичному визначенню відхилень від стандартів система здатна швидко вносити корективи, що мінімізує ризик дефектів та забезпечує високу якість кожного шару.

Інтеграція методів обробки зображень у процес FDM-друку дозволяє машинному зору автоматично коригувати параметри друку для покращення якості виробу. Наприклад, якщо система виявляє зміну адгезії матеріалу на певних ділянках, вона може автоматично регулювати температуру екструдера або швидкість подачі філамента, досягаючи стабільного результату. Системи машинного зору також часто інтегруються з методами машинного навчання, такими як нейронні мережі, які навчаються розпізнавати складні дефекти, включно з внутрішніми деформаціями, на основі великого набору зображень. Це забезпечує точніше розпізнавання дефектів і дозволяє вчасно вживати заходів для збереження якості друку.

Крім того, система машинного зору дозволяє вимірювати розміри шарів у процесі друку та порівнювати їх із цифровою моделлю. У разі відхилень система може автоматично змінити налаштування швидкості або інші параметри друку для забезпечення точності друку та відтворення складних геометрій. Машинне бачення також дає змогу передбачати потенційні проблеми до їхньої появи: аналізуючи текстуру шару або відхилення температури матеріалу, система може прогнозувати ймовірність появи деформацій або розшарувань і заздалегідь коригувати параметри друку, стабілізуючи процес[6].

Додатково, машинний зір дозволяє автоматично регулювати висоту шару та швидкість друку відповідно до виявлених особливостей у поточному шарі. Наприклад, якщо система виявляє ділянки, які потребують більшої точності, вона знижує висоту шару та швидкість друку, що особливо корисно при друці

складних об’єктів, де рівномірність і точність мають критичне значення. Загалом, інтеграція технологій машинного зору в FDM-друк забезпечує значне підвищення якості, стабільності та ефективності процесу.

4 РОЗРОБКА МЕТОДУ УДОСКОНАЛЕННЯ ПАРАМЕТРІВ FDM-ДРУКУ

4.1 Постановка завдання оптимізації параметрів FDM-друку

Оптимізація параметрів FDM-друку є ключовим завданням для підвищення якості та продуктивності адитивного виробництва. Від ефективності налаштування параметрів залежить точність виготовлених виробів, їхня міцність, зовнішній вигляд, а також час і ресурсні витрати на друк. Завдання оптимізації полягає у визначенні найбільш ефективних значень таких параметрів, як швидкість подачі філамента, температура екструдера та платформи, висота шару, а також параметри швидкості друку, які можуть змінюватися залежно від геометрії та матеріалу[7].

Процес оптимізації FDM-друку охоплює низку аспектів. Насамперед, слід визначити критерії оптимізації, які можуть включати мінімізацію дефектів (наприклад, розшарувань, деформацій, пористості), забезпечення високої адгезії шарів, а також досягнення заданої геометричної точності та однорідності поверхні. При цьому важливо враховувати компроміс між швидкістю друку і якістю виробу, оскільки збільшення швидкості може призвести до зниження точності та появи дефектів.

Оптимізація параметрів також має враховувати властивості матеріалу: різні філаменти (PLA, ABS, PETG тощо) мають свої специфічні температурні та механічні характеристики, що потребують індивідуального підходу до налаштування температури, швидкості подачі, а також обробки поверхні. Наприклад, для матеріалів із високою температурою плавлення критичною є стабільність температури екструдера, що безпосередньо впливає на адгезію шарів та точність друку.

Для ефективного налаштування процесу необхідно розробити модель, що враховує залежність між параметрами друку і якісними показниками

кінцевого виробу. Тут зокрема актуальним є використання методів машинного навчання для створення моделі, яка здатна прогнозувати результат друку на основі налаштувань параметрів. Застосування таких моделей дозволяє здійснювати автоматизоване коригування параметрів у реальному часі з урахуванням змінних умов друку.

Розробка методів виявлення дефектів за допомогою машинного зору є ще одним важливим аспектом оптимізації, оскільки вона дозволяє оцінювати стан поверхні друкованого виробу та якість шарів у процесі друку. Така система дозволяє оперативно реагувати на можливі проблеми, що виникають під час друку, автоматично коригуючи параметри, щоб мінімізувати ризик дефектів та втрати матеріалу.

Таким чином, завдання оптимізації параметрів FDM-друку можна представити як багатокритерійну задачу, що охоплює вибір оптимальних значень параметрів для забезпечення стабільної якості, зменшення витрат на друк і підвищення продуктивності. Розробка ефективних алгоритмів для цього завдання дозволить значно розширити можливості FDM-технології, забезпечивши високу якість виробів навіть за складних умов виробництва.

Основні цілі оптимізації включають:

- **Мінімізація кількості дефектів** – зменшення ризику розшарування, деформацій, зміщення шарів та інших дефектів.
- **Підвищення точності** – забезпечення максимальної відповідності виробу початковій 3D-моделі.
- **Збільшення міцності виробів** – оптимізація параметрів, які впливають на внутрішню структуру та зчеплення шарів.

Підходом до реалізації цього завдання є розробка автоматизованої системи, що здійснює аналіз параметрів у режимі реального часу, використовуючи дані з системи машинного зору.

4.2 Метод контролю якості на основі машинного зору

Метод контролю якості на основі машинного зору в системах FDM-друку забезпечує автоматизоване виявлення та аналіз дефектів, моніторинг параметрів друку, а також оперативне коригування процесу для забезпечення високої якості кінцевих виробів. Основою такого методу є інфраструктура з камерами, які безперервно відстежують процес друку, та алгоритми обробки зображень, які аналізують параметри кожного шару[8].

Компоненти методу:

1. Система збору даних. Камери високої роздільної здатності розміщуються так, щоб охопити весь робочий простір 3D-принтера, фіксуючи кожен новий шар у процесі друку. Окрім камер, додаткові датчики можуть вимірювати температуру екструдера, стан філамента, швидкість подачі, тощо. Зібрані зображення та дані про параметри друку є основним матеріалом для подальшого аналізу.

2. Попередня обробка зображень. Перед надходженням на аналіз, отримані зображення проходять етапи корекції, усунення шумів та вирівнювання. Це необхідно для нормалізації зображень та покращення їх якості, зокрема виділення чітких контурів шарів та зменшення можливих артефактів, що можуть завадити правильному аналізу.

3. Аналіз дефектів у режимі реального часу. Зображення обробляються за допомогою алгоритмів машинного навчання, які визначають різноманітні дефекти, як-от розшарування, неякісну адгезію, пропуски матеріалу чи деформації. Використання нейронних мереж дозволяє розпізнавати складні форми дефектів, які можуть бути невидимими для простих алгоритмів обробки зображень.

4. Зворотний зв'язок і корекція параметрів. При виявленні дефекту система контролю якості автоматично надсилає сигнал на контролер 3D-принтера. Контролер може змінити швидкість друку, температуру, висоту шару та інші параметри, що зменшить вірогідність повторення дефекту в подальших

шарах. Це дає можливість оперативно реагувати на зміни та досягати стабільного рівня якості виробу.

5. Система моніторингу та навчання. Зібрані дані та виявлені дефекти архівуються для подальшого аналізу та використання у навчанні моделей машинного зору. Це дозволяє системі з часом удосконалювати свої можливості виявлення дефектів, оскільки база навчальних зображень постійно оновлюється й розширюється.

5 МЕТОД УДОСКОНАЛЕННЯ ПАРАМЕТРІВ FDM-ДРУКУ

Цей метод рис. 5.1, забезпечує ефективне виявлення та корекцію друкарських дефектів майже в реальному часі, що дозволяє підвищити якість друку та знизити витрати на матеріали.

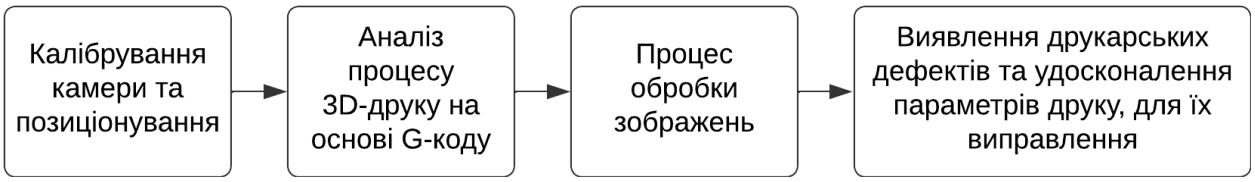

Рис. 5.1 – Метод удосконалення параметрів FDM-Друку

1. **Калібрування камери та позиціонування**: Виконується налаштування камери за допомогою шаблону для точного визначення параметрів камери, початкової позиції та однорідності координат. Це забезпечує проєкцію точок 3D об’єкта на площину зображення[9].

2. **Аналіз процесу 3D-друку на основі G-коду**: Аналізується G-код, розділяючи його на шари і сегменти, що дозволяє створити віртуальний вигляд зверху та псевдовид збоку, такий підхід дає можливість оцінити розбіжність вертикального розміру деталі з еталонною висотою.

3. **Процес обробки зображень**: Розділяється на три гілки, кожна з них відповідає за аналіз різних аспектів друку: перевірку висоти в псевдокутовому вигляді, відповідність глобальної траєкторії та локальну перевірку текстури. Це дозволяє враховувати як глобальні, так і локальні параметри друку.

4. **Виявлення друкарських дефектів та удосконалення параметрів друку, для їх виправлення**: За допомогою попередніх шагів можна виявити значну кількість дефектів друку, які усуваються за допомогою відповідних команд G-коду без налаштування механічних параметрів принтера та режимів нарізки.

5.1 Калібрування камери та позиціонування

Друкована область знаходиться під монокулярним наглядом, рис. 5.2, де одна камера забезпечує коригований верхній вигляд і псевдокутовий вид надрукованої частини.

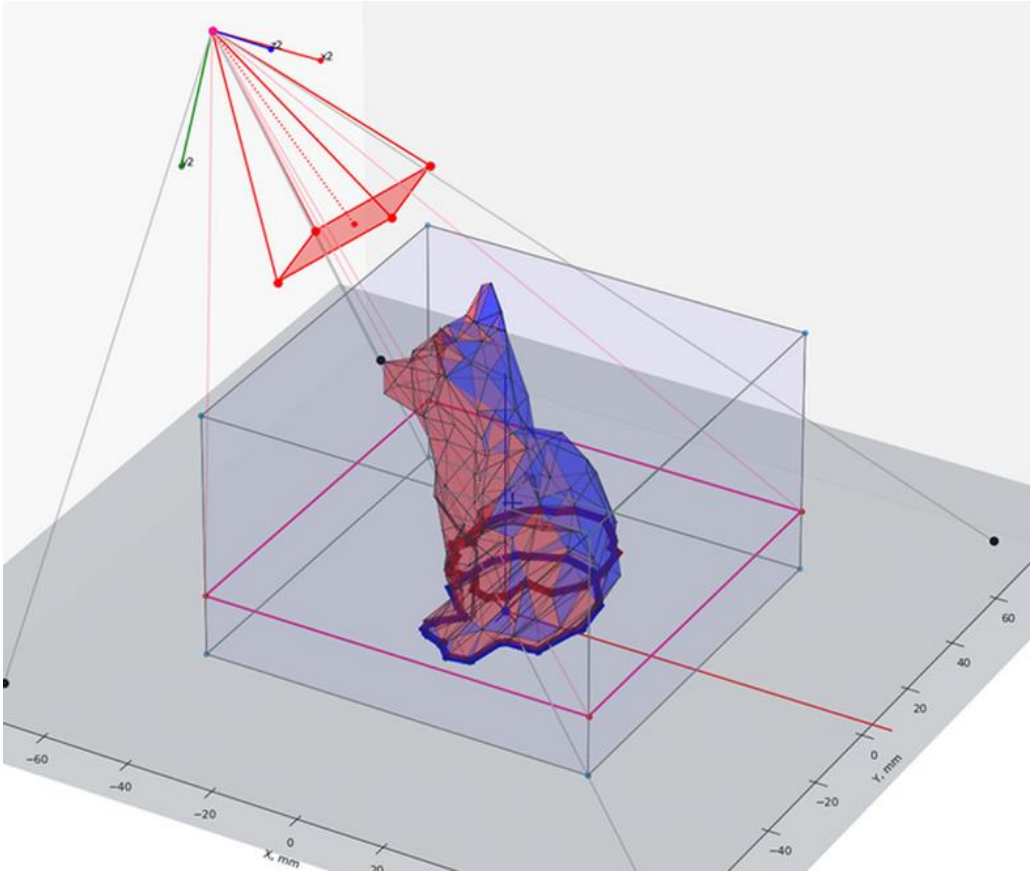

Рис. 5.2 – Положення камери відносно моделі

Маркерна пластина з контрастними квадратами (розміром 15x15 мм та 10x10 мм), розміщена на платформі для друку, слугує системою орієнтирів для камери, яка дозволяє визначати просторову позицію робочої області відносно об'єктива[10]. Розрахунок позиції камери в однорідних координатах дозволяє обчислити взаємозв’язок один-до-одного між евклідовими точками $\mathbb{R}^n$, записаними в G-коді або в моделі STL, і точками планарного зображення, знятими камерою, шляхом застосування проективних перетворень, рис. 5.3.

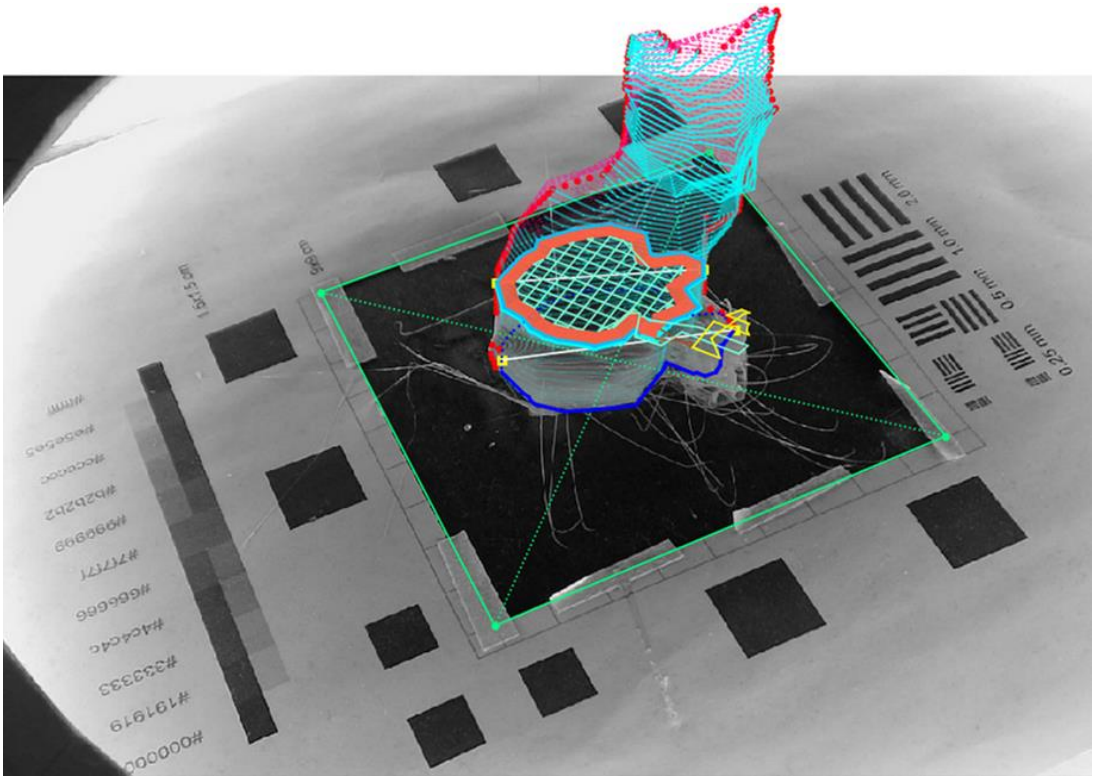

Рис. 5.3 – Траєкторії G-коду, спроектовані на вихідний кадр зображення

У задачах комп'ютерного зору проективні перетворення використовуються як зручний спосіб представлення реального тривимірного світу шляхом розширення його до тривимірного проективного простору $\mathbb{P}^n$, де його точки є однорідними векторами.

Позиції пікселів зображення відповідають їх тривимірному просторовому положенню відповідно до наступного рівняння, де індекс p означає «площину зображення», а індекс w означає «світовий простір»:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}, \quad (5.1)$$

де $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ – це власні параметри камери, отримані під час калібрування, f_x і f_y – фокусні відстані в координатах зображення, c_x і c_y –

координати оптичного центру в координатах зображення (головна точка), $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ – матриця обертання, а $t = [t_x \quad t_y \quad t_z]^T$ – вектор перекладу.

Проективні перетворення застосовуються для коригованих кадрів друкованої області, що дозволяє отримувати віртуальний вигляд зверху, рис. 5.4, наче камера встановлена паралельно до нормалі поверхні платформи для друку, а також псевдокутовий вигляд, наче камера встановлена перпендикулярно до неї, рис. 5.5. Спостерігаючи за надрукованим шаром через об'єктив камери, можна розглядати зріз матеріалу як набір пікселів, що характеризують локальні області шару. Аналіз двовимірного зображення надає розуміння про текстуру об'єкта.

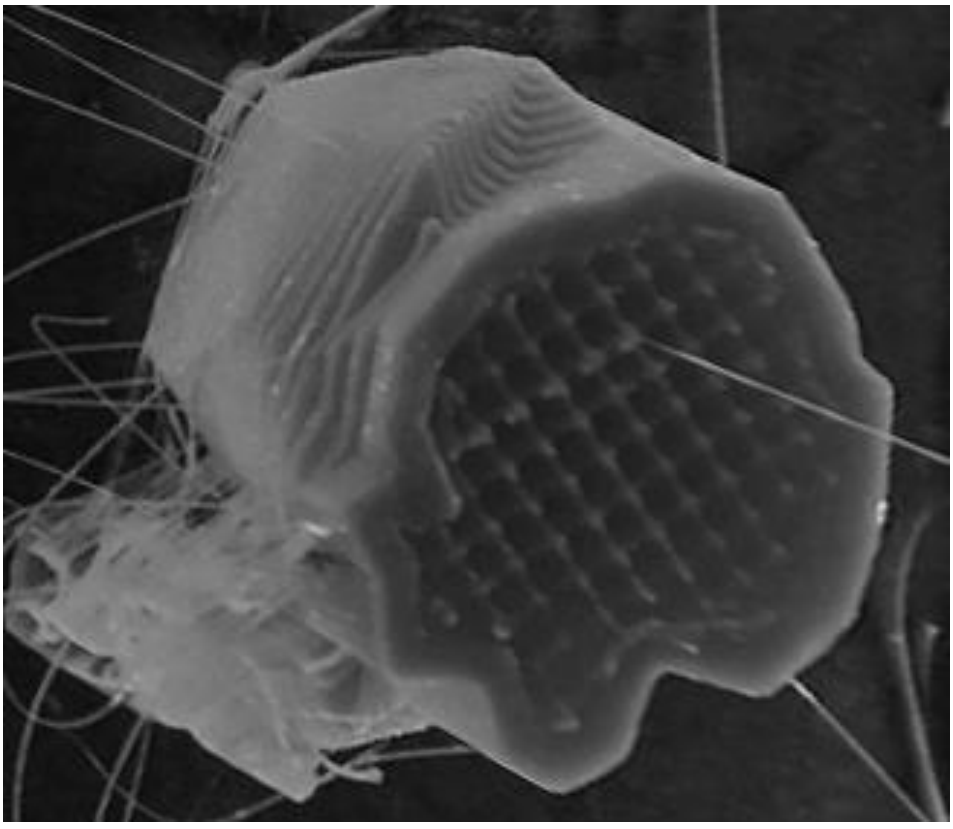

Рис. 5.4 – Віртуальний вигляд зверху

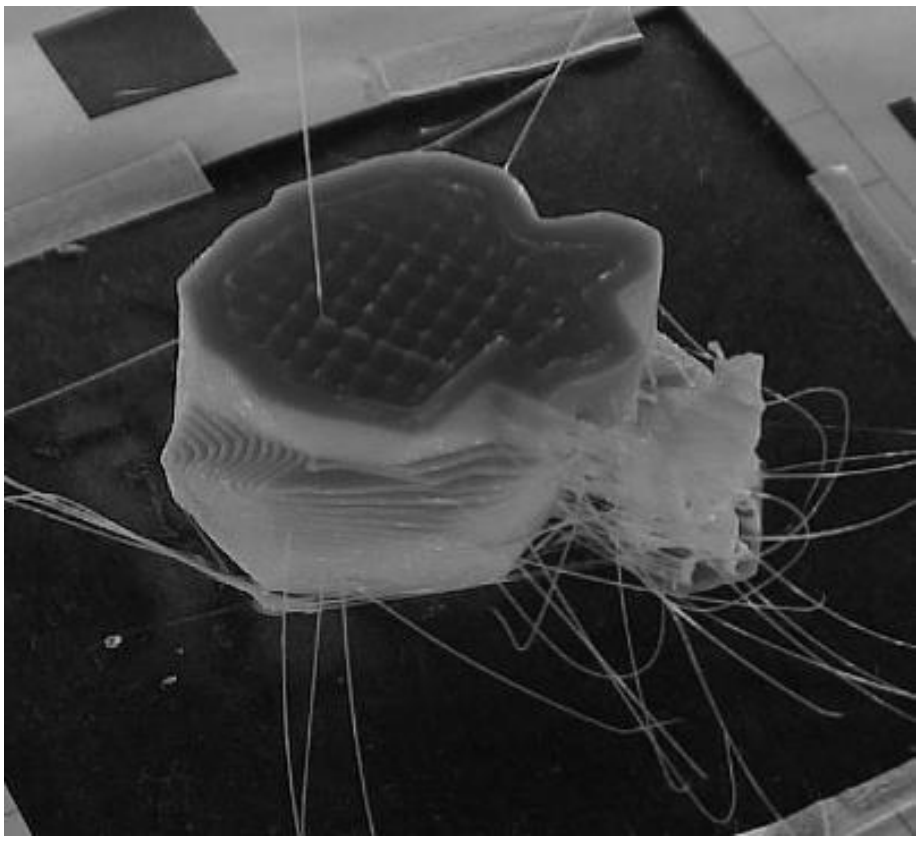

Рис. 5.5 – Псевдокутовий вигляд

Після кожного шару, на основі висоти надрукованого шару, аналітична проекційна площина зображення зсувається відповідно до номера шару, щоб коригований кадр залишався ортогональним до оптичної осі віртуальної верхньої камери. Це дозволяє сегментувати значущі області контуру та текстури на основі зображень і відомих параметрів моделі STL та G-коду друкованого об’єкта, рис. 5.6.

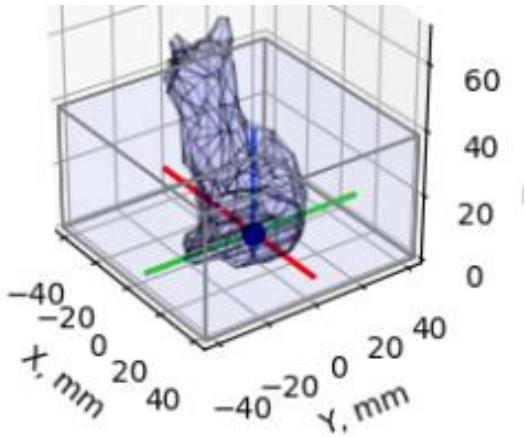

Рис. 5.6 – Модель STL з ортогональним скануванням

Наприкінці процесу друку багатошаровий набір зображень, вертикального зрізу XZ, рис. 5.7 та вертикального зрізу YZ, рис. 5.8, забезпечує додаткові дані для об’ємного аналізу надрукованої частини в майбутньому.

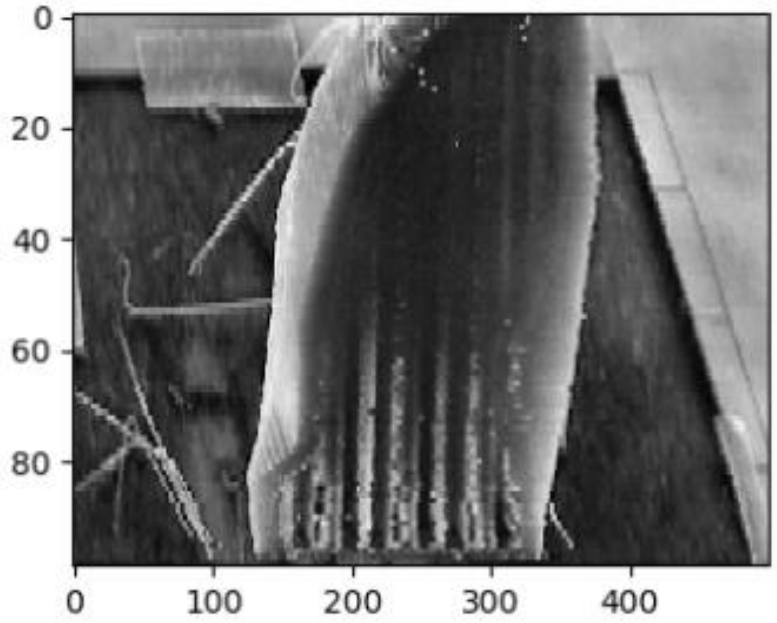

Рис. 5.7 – Вертикальний зріз XZ

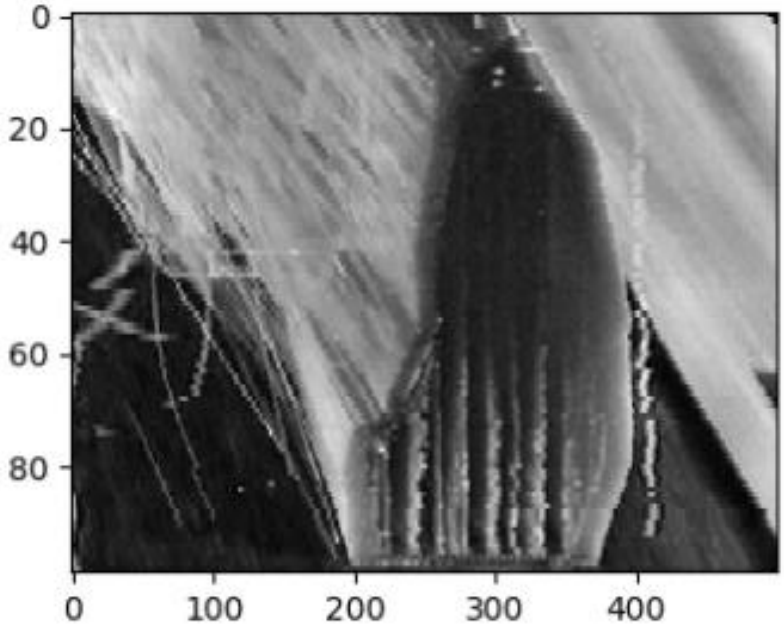

Рис. 5.8 – Вертикальний зріз YZ

5.2 Аналізу процесу 3D-друку на основі G-коду

Програмне забезпечення, розроблене на основі Python, розбирає вихідний G-код, розділяючи його на шари та сегментуючи шляхи екструдера на такі категорії, як обрамлення, заповнення, зовнішні та внутрішні стінки, підтримка тощо, рис. 5.9.

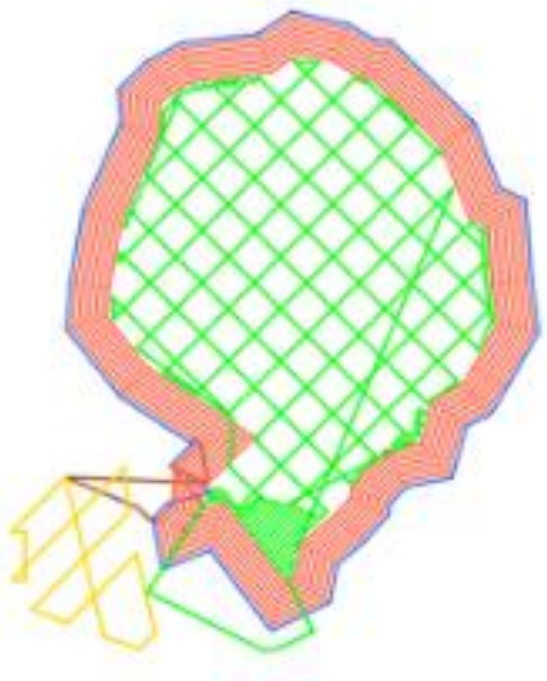

Рис. 5.9 – Вихідний G-код, розділений на шари: синій(зовнішня стінка), червоний(внутрішня стінка), зелений(заповнення), і жовтий(підтримка)

Категорії шляхів G-коду залежать від програмного алгоритму, який використовується для розділення моделі STL. Траєкторії G-коду для будь-якого шару можуть не точно збігатися з контуром шару, рис. 5.10, отриманим з моделі STL, а величина розбіжності може досягати кількох міліметрів, модель STL стає ненадійною, а координати G-коду є еталонним джерелом позиціонування. Таким чином, віртуальний вигляд зверху та псевдокутовий вигляд дозволяють оцінити відхилення за вертикальними розмірами та формою частини.

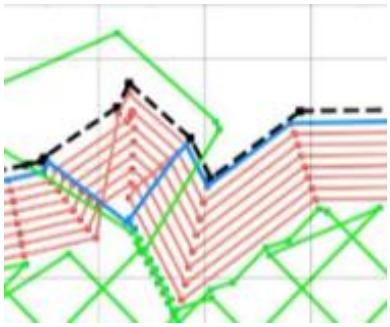

Рис. 5.10 – Можлива невідповідність STL і G-коду

Використовуючи контрольні пошарові траєкторії екструдера, отримані за допомогою аналізу G-коду, крім віртуального вигляду зверху, також можна створити псевдовид збоку. Такий підхід не дозволяє створити повну розгортку всієї бічної поверхні моделі, однак дає можливість оцінити розбіжність вертикального розміру деталі з еталонною висотою, отриманою з G-коду. Використання однієї камери замість двох (основної та додаткової для бокового огляду) зменшує обчислювальне навантаження та усуває необхідність синхронізації обробки двох зображень (а також зменшує витрати на обладнання).

Параметри температури, координати траєкторій, швидкість подачі екструдера, а також товщина надрукованих ліній і висота шару зберігаються в пам'яті програми для кожного шару. Команди друку, паузи, аналіз зображень і алгоритми прийняття рішень виконуються програмою, що забезпечує повний контроль над процесом друку. У разі критичних відхилень друк може бути призупинено, і, якщо є можливість відновлення частини, виконується послідовність коригуючих команд G-коду[21].

5.3 Процес обробки зображень

Процес обробки зображень можна розділити на три гілки, рис. 5.11, які описані далі.

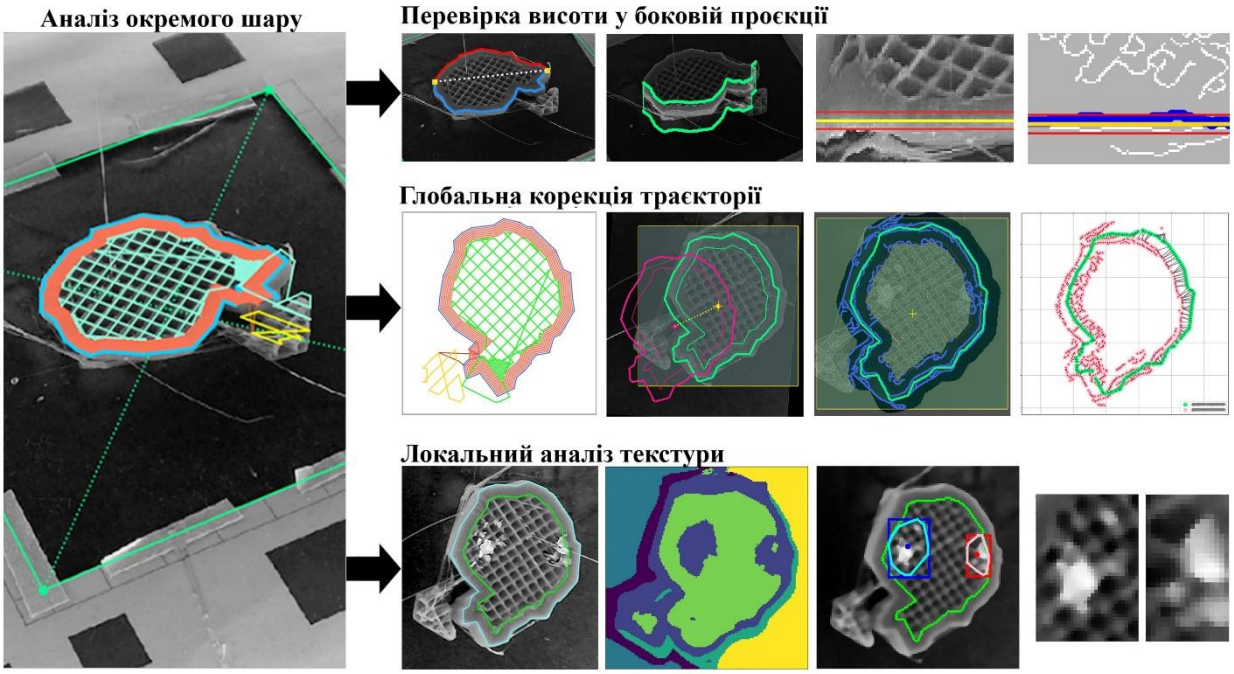

Рис. 5.11 – Конвеєр обробки зображень

Кожна з них відповідає за аналіз різних аспектів друку: перевірку висоти в псевдокутовому вигляді, відповідність глобальної траєкторії та локальну перевірку текстури. Це дозволяє враховувати як глобальні, так і локальні параметри друку.

Критерії, такі як вирівнювання платформи, втрата розмірності, залежать від конкретної моделі принтера та вручну калібруються користувачем під час першого запуску. Можна створити таблиці калібрування для визначення корекційних коефіцієнтів траєкторій G-коду. Проте на цьому етапі зазначені параметри перевіряються лише на відповідність/невідповідність заданим значенням. У разі невідповідності за вирівнюванням платформи, розмірами чи формою друк призупиняється. Такий підхід не усуває ці помилки під час друку, але дозволяє заощадити час і матеріал.

5.3.1 Перевірка висоти у боковій проєкції

Знаючи положення камери та траєкторії G-коду для поточного шару, можна проаналізувати видимість бокової частини надрукованої деталі,

розв'язавши систему рівнянь(2), яка визначає коефіцієнти нахилу та зсуву для лінійного обмеження видимості.

$$\begin{cases} y_p^{(1)} = m * x_p^{(\min)} + b \\ y_p^{(2)} = m * x_p^{(\max)} + b' \end{cases} \quad (5.2)$$

Де m і b – коєфіцієнти лінійного обмежувача видимості $x_p^{(\min)}$і $x_p^{(\max)}$ отримані з крайніх контурних точок проекції контуру G-Code на площину малюнка, $y_p^{(1)}$ і $y_p^{(2)}$ – у-координати відповідних райніх точок на площині малюнка.

Псевдовигляд збоку дозволяє контролювати висоту надрукованої частини та виявляти критичні помилки, такі як «заблоковане сопло», «нестача матеріалу», «значні деформації» тощо.

5.3.2 Глобальна корекція траєкторії

Після перевірки вертикального розміру використовується віртуальний вигляд зверху для подальшого двоетапного аналізу зовнішнього контуру та заповнення надрукованого шару. Маючи дані про відповідні траєкторії екструдера з G-Code та отриманий контур, можна визначити, чи існує невідповідність між реальними обрисами та еталонними межами, використовуючи алгоритми багатошаблонного співставлення (MTM) [12] та ітеративної найближчої точки (ICP) [7].

MTM дозволяє відстежувати значні горизонтальні та вертикальні зміщення надрукованої частини на основі бінарного шаблону шару, отриманого з траєкторій G-Code, тоді як алгоритм ICP визначає точні обертання та трансляції у межах невеликого діапазону відхилень. У результаті формується матриця трансформації, яка, помножена на просторові координати

траєкторій екструдера, усуває невеликі лінійні зміщення та спотворення масштабу надрукованого шару.

Метод MTM [12] обчислює карту кореляції між еталонним контуром шару та бінарним краєвим зображенням віртуального вигляду зверху на основі функції OpenCV **"matchTemplate"** [4] і передбачає положення шаблону (еталонного контуру G-Code) на зображенні. Оскільки алгоритм виконує пошук шляхом ковзання шаблону по зображенню, він виявляє об'єкт із подібною орієнтацією до шаблону, але може бути нечутливим до обертання.

Алгоритм ICP [7] спрямований на знаходження матриці трансформації між двома хмарами точок шляхом мінімізації квадратичної похибки між відповідними поверхнями за допомогою методу градієнтного спуску. Ітеративний алгоритм сходиться за умови, що початкові позиції хмар точок знаходяться близько одна до одної.

За наявності двох відповідних наборів точок $\{m_1, m_2, \dots, m_n\}$ і $\{p_1, p_2, \dots, p_n\}$ ми можемо знайти трансляцію t, обертання R і масштабування s, які мінімізують квадратичну суму помилки:

$$E(R, t, s) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|m_i - sR(p_i) - t\|^2, \tag{5.3}$$

де *R, t* і s – обертання, трансляція і масштабування відповідно. Оскільки операція масштабування також може транслювати об’єкт, центр проекції шару G-коду має бути розміщений у початку координат.

На основі обертання, масштабування та трансляції, отриманих за допомогою алгоритму ICP, і припускаючи, що z-рівень було знайдено під час перевірки вертикального розміру, траєкторії G-коду для наступного (k+1)-го шару $[G_x^{(k+1)\prime} G_y^{(k+1)\prime}]^T$ буде перетворено з початкових траекторій наступного шару $[G_x^{(k+1)} G_y^{(k+1)}]^T$ відповідно до рівняння:

$$\begin{bmatrix} G_x^{(k+1)\prime} \\ G_y^{(k+1)\prime} \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix} * \begin{bmatrix} G_x^{(k+1)} \\ G_y^{(k+1)} \\ 1 \end{bmatrix}. \quad (5.4)$$

5.3.3 Локальний аналіз текстури

Після аналізу контуру проводиться перевірка шару, заповненого матеріалом. Метою цього етапу є виявлення нерівномірних ділянок текстури всередині заповнення шару. На цьому етапі припускається, що вертикальний розмір деталі відповідає заданому, а правильне розташування реальних меж деталі вже визначено. Таким чином, розглядаються лише нерівності текстури в області, обмеженій зовнішньою оболонкою шару.

Текстурні ознаки, засновані на локальних ймовірнісних мірах подібності, є простими, мають низьке обчислювальне навантаження та добре підходять для невеликої кількості специфічних випадків, але можуть бути неефективними для широкого спектра реальних завдань [13]. Через складну топологію поверхні текстурні варіації можуть не бути чітко вираженими через порівняння гістограм [14].

У цій роботі для сегментації текстур було реалізовано метод, заснований на текстонах, оскільки цей підхід неодноразово демонстрував свою ефективність і масштабованість [14]. Сегментація на основі текстонів використовує фільтри Леунга-Маліка (ЛМ) [14], які працюють подібно до клітин зорової кори та дозволяють сегментувати зображення на канали узгодженої яскравості та текстури природним чином. Текстура при цьому формується через просторові варіації нормалі поверхні та відбивних властивостей.

Банк фільтрів ЛМ, рис. 5.12, складається з 48 ядер фільтрів, які є комбінацією 36 видовжених фільтрів, 8 фільтрів різниці гауссіанів і 4 низькочастотних гауссіанських фільтрів [14].

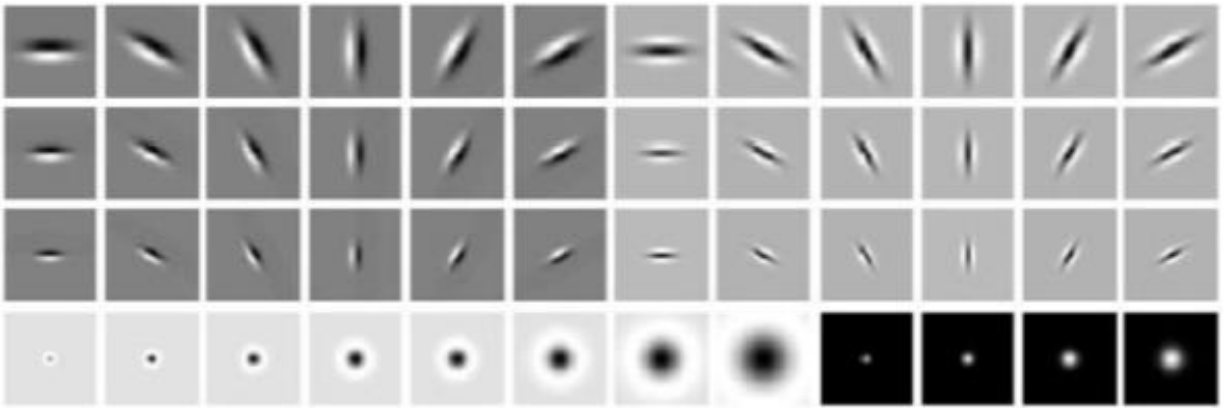

Рис. 5.12 – Банк фільтрів Леунга-Маліка

Після згортки зображення з ядрами фільтрів кожен піксель перетворюється на багатовимірний вектор відгуків фільтрів. Після кластеризації ці вектори формують набір текстонних каналів або векторів зовнішнього вигляду, які визначають розподіл зображення на сегменти.

У адитивному виробництві з використанням пластику та металу освітлення та взаємні відбиття створюють неламбертове середовище, де схожі поверхні можуть виглядати значно по-різному під різними кутами огляду [14]. Це обмежує набір можливих методів обробки зображень. Згідно з [11], відгуки фільтрів кодують інформацію про зовнішній вигляд у широкому діапазоні масштабів і можуть слугувати методом попередньої обробки, який комбінується з густими дескрипторами для ефективної класифікації текстур за умов змінного освітлення.

Без попередньої інформації про текстуру відповідним підходом до кластеризації отриманих відгуків фільтрів, а отже, і для сегментації текстур, є методи навчання без учителя. Це поширена техніка у розпізнаванні об’єктів, яка дозволяє отримувати висновки з даних, що не мають міток. Кластеризація, як найпоширеніший метод навчання без учителя, спрямована на пошук прихованих закономірностей у вихідному наборі даних і групування їх відповідно до їхніх основних характеристик та кількості кластерів **k**, заданих користувачем.

Більшість попередніх досліджень у сфері сегментації текстур базуються на кластеризації методом **k-середніх** [11]. Однак його непараметрична природа, жорсткі межі кластерів та відсутність гнучкості у формі кластерів створюють практичні проблеми і можуть не працювати ефективно в реальних умовах.

Метод кластеризації з використанням **Гауссової змішаної моделі (ГЗМ)**, реалізований у цій роботі, може дати кращі результати. Кластеризація **ГЗМ** базується на відгуках фільтрів і намагається розділити текстуру вхідного шару без міток як суміш багатовимірних гауссових ймовірнісних розподілів у регіонах, що мають спільні характеристики. Гауссова змішана модель (ГЗМ) для невідомої кількості кластерів **k** може бути представлена як суперпозиція Гауссових розподілів [15]:

$$f_{GMM}(x) = \sum_{j=1}^{k} w_j f_{N(\mu_j,\Sigma_j)}(x), where \sum_{j=1}^{k} w_j = 1, \quad (5.5)$$

яка є комбінацією зважених w_j нормальних функцій щільності $f_{N(\mu_j,\Sigma_j)}$ із вектором середнього μ_j так коваріаційною матрицею $\sum_j$.

Кластеризація на основі Гауссової змішаної моделі (**ГЗМ**) визначає максимальну правдоподібність для Гауссових моделей із прихованими змінними, використовуючи алгоритм очікування-максимізації (Expectation-Maximization, EM) [16, 17]. Цей алгоритм ітеративно оцінює середні значення, коваріації та вагові коефіцієнти таким чином, що кожен кластер асоціюється з гладкою Гауссовою моделлю.

Однак, попри ефективність описаного методу, точна кількість кластерів **k** повинна бути задана заздалегідь. Це є критичним рішенням, яке визначає успіх сегментації текстури.

Використовуючи маску заповнення, отриману з траєкторій G-коду, сегментація **ГЗМ** і аналіз дефектів виконуються виключно в області текстури заповнення шару. Оскільки результат сегментації не завжди передбачуваний, і одна аномальна ділянка може складатися з кількох сегментів, після розподілу

ГЗМ запускається агломеративна ієрархічна кластеризація (Agglomerative Hierarchical Clustering, AHC) [18].

AHC, як метод навчання без учителя, рекурсивно об’єднує пари окремих кластерів на основі їхнього розташування в межах сегментованої області. Це забезпечує узгоджену карту дефектів для аналізованої області заповнення.

5.4 Виявлення друкарських дефектів та удосконалення параметрів друку, для їх виправлення

За допомогою вищезазначених методів можна виявити значну кількість помилок друку. Основні типи помилок, разом із запропонованими коригувальними діями, представлені в табл. 5.1. Ці помилки можна виявити та/або усунути за допомогою відповідних команд G-коду без потреби в регулюванні механічних параметрів принтера або режимів нарізки.

Таблиця 5.1

Цільові відмови та коригувальні дії

Тип помилки	**Стратегія виявлення**	**Зміна параметрів**
Нестача філамента	Перевірка вертикального рівня + алгоритми MTM та ICP	Призупинити друк / Звітувати
Заблоковане сопло	Перевірка вертикального рівня + алгоритми MTM та ICP	Підвищити температуру сопла; Повторити попередній шар
Відсутній шар	Перевірка вертикального рівня + алгоритми MTM та ICP	Повторити попередній шар
Втрата точності розмірів	Алгоритми MTM та ICP	Оновити координати G-коду
Проблема з вирівнюванням платформи	Сегментація текстури	Призупинити друк / Звітувати; Ручна перекалібровка

Продовження таблиці 5.1

Цільові відмови та коригувальні дії

Проблема адгезії	Перевірка вертикального рівня початкового шару	Підвищити температуру платформи; Звітувати у разі критичного вертикального відхилення
Погане прилягання друку до платформи	Перевірка вертикального рівня + алгоритми MTM та ICP	Підвищити температуру платформи; Призупинити друк/Звітувати
Зсув або згин друку	Перевірка вертикального рівня + алгоритми MTM та ICP	Оновити координати G-коду
Слабке або недостатнє заповнення	Сегментація текстури	Підвищити температуру насадки та швидкість подачі
Деформоване заповнення	Сегментація текстури	Змінити температуру сопла та швидкість подачі
Залишки обгорілого філамента	Сегментація текстури	Виконати вирівнювання
Неповне заповнення	Сегментація текстури	Процедура заміни проблемної області
Низька якість поверхні над підтримкою	Сегментація текстури	Змінити швидкість подачі
Щілини між заповненням і оболонкою	Сегментація текстури	Змінити швидкість подачі

6 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

Алгоритм був протестований під час звичайного друку без збоїв на моделі низькополігональної лисиці розміром 42x51x70 мм [19] з наступними параметрами друку: PLA діаметром 1.75 мм, висота шару 0.4 мм, ширина лінії 0.4 мм, 30% заповнення сіткою та товщина стінки 3.2 мм. Уся модель складається з 175 шарів, але тести проводилися лише для перших 96 шарів, оскільки частина моделі була розташована поза видимою зоною.

Рівень платформи та перевірка розмірів були заздалегідь відкалібровані перед друком. Візуальний аналіз вертикального рівня, деформацій зовнішньої оболонки та текстури заповнення виконувався для кожного шару друку.

Ці експерименти у режимі нормального друку без відхилень дозволяють визначити точність і допустимі межі адаптивного алгоритму.

Розпочинаючи з перевірки вертикального рівня, алгоритм аналізує віртуальний вигляд зверху для глобального співставлення траєкторій і локального аналізу текстури. Це дозволяє враховувати як глобальні, так і локальні параметри процесів друку.

На основі еталонного контуру G-коду та системи рівнянь розрахований візуальний сепаратор використовується для створення псевдобічної проєкції, у якій вигнута частина моделі представлена як пряма лінія, рис. 6.1.

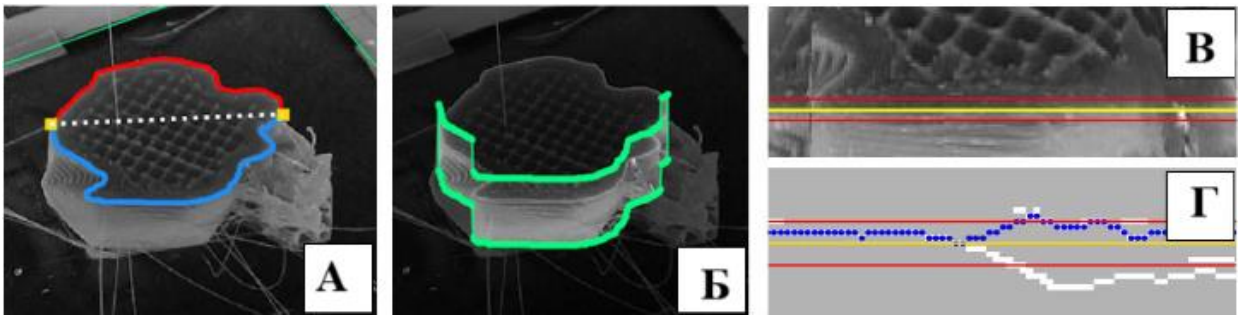

Рис. 6.1 – Генерація псевдо-бічного виду: А – обчислений лінійний роздільник видимості (білий пунктир), край для видимої області (синя) та невидимої області (червона) надрукованої деталі; Б – визначена видима

область для розгортання; В – розгорнута область з опорним вертикальним рівнем (жовтий) і максимальними помилками для двох шарів в обох напрямках (червоний); Г – виявлена помилка вертикального краю (синій) з опорною висотою шару (жовтий) і максимальними помилками для двох шарів в обох напрямках (червоний).

Завдяки рівномірному освітленню з усіх боків контраст між сусідніми гранями деталі достатній для визначення чіткої межі. Таким чином, порівняння покоординатного відхилення виявленої межі від еталонного контуру дозволяє отримати розподіл і загальну похибку вертикального рівня для кожного шару .

Як показують експериментальні результати, у режимі нормального друку середнє та загальне значення похибки не перевищують максимального відхилення, яке дорівнює висоті двох шарів. Однак для кожного окремого шару похибка вертикального рівня може перевищувати максимальне значення похибки, рівне висоті одного шару.

Роздільна здатність алгоритму виявлення дефектів залежить від роздільної здатності камери, відстані до області друку та розміру деталі. Це можна врахувати в алгоритмі таким чином, щоб разове відхилення, яке перевищує висоту одного шару, ігнорувалося, тоді як багаторазові послідовні перевищення висоти одного шару розглядалися як справжня похибка.

Аналіз глобального контуру використовує віртуальний вигляд зверху та комбінацію алгоритмів MTM і ICP.

На першому етапі алгоритм MTM визначає значні горизонтальні та вертикальні зміщення, рис. 6.2, після чого алгоритм ICP надає більш точну інформацію про обертання та зміщення в невеликому діапазоні в межах області, виявленої алгоритмом MTM, рис. 6.3.

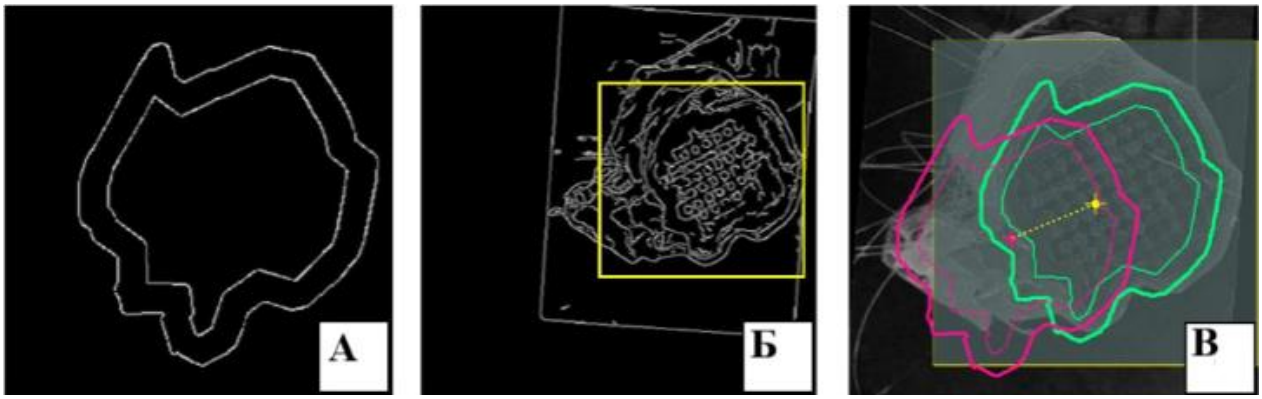

Рис. 6.2 – Виявлення глобального зміщення на основі алгоритму MTM:
А – бінарний шаблон на основі контуру; Б – друкована частина зміщена
черезе збій; В – розрахунок відстані і напрямку зміщення.

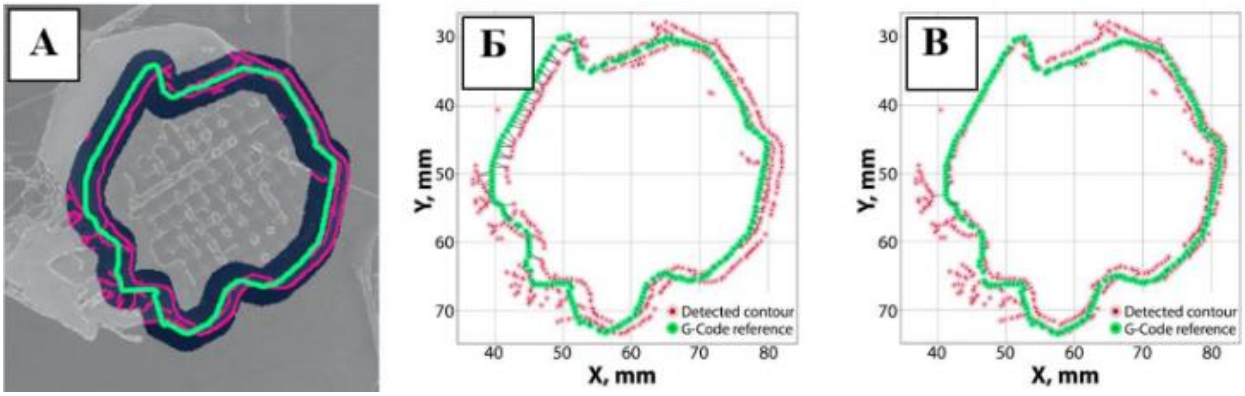

Рис. 6.3 – Глобальне узгодження траєкторії G-Code для одного шару на основі алгоритму ICP: А – опорний контур (зелений), що не збігається з виявленими точками контуру (червоний); Б – початкова ітерація ICP;
В – фінальна ітерація ICP.

Для забезпечення надійності роботи алгоритму ICP використовується обмежувальна маска, заснована на контурі шару STL, яка обмежує кількість точок краю, що враховуються під час обчислень зміщення та обертання відносно еталонного контуру G-коду, рис. 6.3(А).

У ході експериментів було виявлено, що обмежувальна маска шириною 30 пікселів (що відповідає 5.7 мм у даній конфігурації, або приблизно 10% горизонтального розміру надрукованої деталі) дозволяє отримати стабільний

результат. Максимальні значення зміщення та обертання становили 8 мм та 10 градусів відповідно.

У номінальному режимі друку спостерігалися помилки обертання до 2 градусів і помилки зміщення до 1.7 мм.

Розміри фільтра повинні корелювати з розміром домінуючих структур зображення, щоб зменшити вплив шуму під час обробки зображення.

З огляду на те, що для прискорення операції кластеризації оригінальне зображення може бути зменшено, а більший фільтр краще придушує візуальний шум, експериментально було визначено, що розмір фільтра, що становить 1/3 розміру вхідного зображення, дозволяє ефективно сегментувати текстуру шару з високою швидкістю. Таким чином, розміри вхідного зображення та фільтрів Leung-Malik становлять 150x150 пікселів і 49x49 пікселів відповідно.

Крім розміру фільтра, не менш важливим параметром є кількість очікуваних кластерів текстури. У ході експериментів було встановлено, що шість кластерів (шість очікуваних текстур на вхідному зображенні) забезпечують ефективне співвідношення швидкості та якості сегментації.

На, рис.6.4, представлений результат сегментації **ГЗМ** тестового зображення, що містить 18 зразків текстури, де розмір кожної текстурної області дорівнює розміру фільтра і становить 49x49 пікселів.

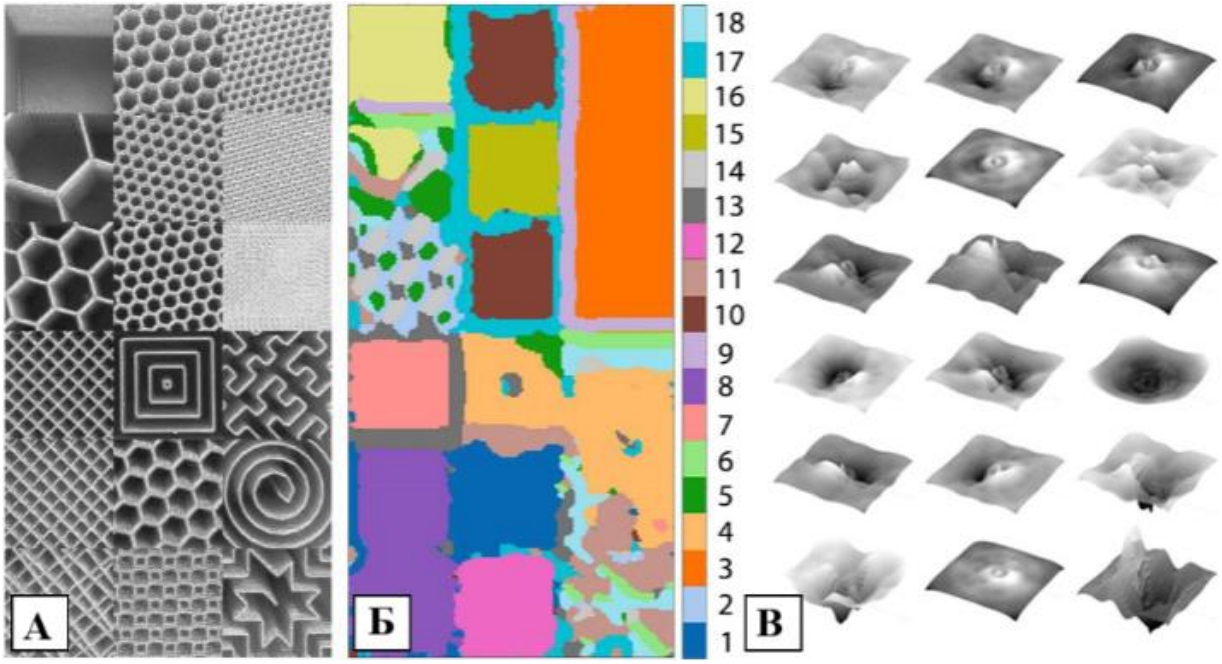

Рис. 6.4 – Розбиття зображення на текстурні канали (текстони):
А – тестове зображення з 18 різними зразками текстур; Б – сегментовані текстури; С – отримані текстурні канали.

Текстони, вектори зовнішнього вигляду або відповідні вектори відгуку фільтрів фіксують характерні форми різних матеріалів і особливостей за різних умов огляду та освітлення. На, рис. 6.4, показано 18 кластерів, що відповідають основним особливостям зображення [14], де кожен центр кластера візуалізується через псевдообернене зображення, яке кодує геометричні особливості, такі як жолоби, виступи, гребені та западини, у вигляді значка зображення [20].

Після сегментації вхідного зображення здійснюється маркування текстурних областей усередині маски заповнення, рис. 6.5.

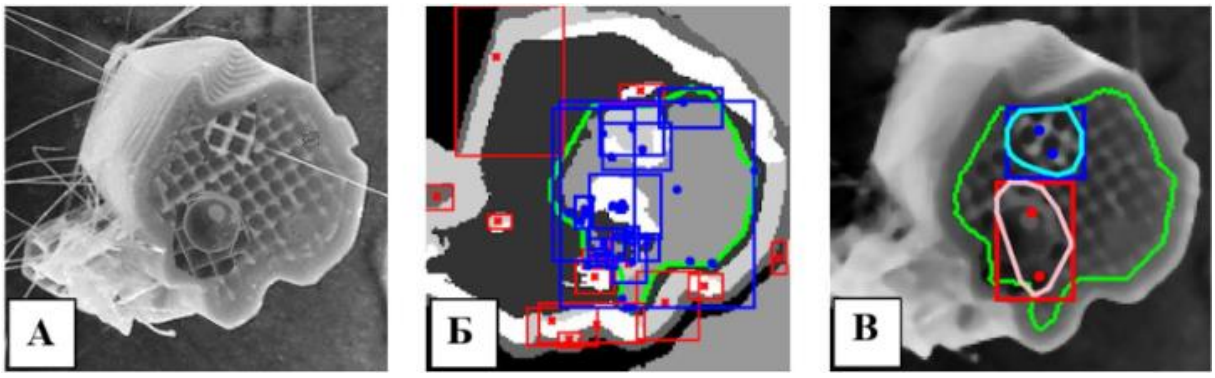

Рис. 6.5 – Результати сегментації дефектного шару: А – вихідний віртуальний вид зверху; Б – сегментоване зображення з маскою заповнення (зелена), текстурами заповнення (сині області), зовнішніми текстурами (червоні області); В – сегментовані дефекти (червоні та сині області) всередині області заповнення (зелена).

Текстурні області, які не відповідають основній текстурі заповнення, вважаються аномальними. Якщо вони мають певну форму та розмір, ці області враховуються у подальшому аналізі. Якщо загальна площа аномальних областей перевищує критичне значення в 15% від загальної площі області заповнення шару, шар вважається дефектним, і проводиться подальша ієрархічна агломеративна кластеризація без нагляду.

Оскільки після сегментації **ГЗМ** дефектна частина текстури може складатися з кількох сегментів, необхідно визначити їхню належність до тієї чи іншої групи дефектів. Для алгоритму АНС параметри кластеризації були експериментально підібрані на основі розташування центрів мас аномальних областей та відстані між ними. Алгоритм передбачає наявність однієї або двох дефектних зон заповнення та визначає, до якої із зон належать сегментовані аномалії.

Під час експериментів розглядалася загальна площа аномальних областей без урахування їхніх форм, загальна площа аномальних областей може досягати 10% або більше від області заповнення. Цей факт пояснюється проникненням текстури зовнішньої стінки в межі маски заповнення, що є

помилковим спрацюванням і може бути усунено шляхом аналізу форми цих областей.

Приклади сегментації штучно створених дефектів друку, представлені на, рис.6.6.

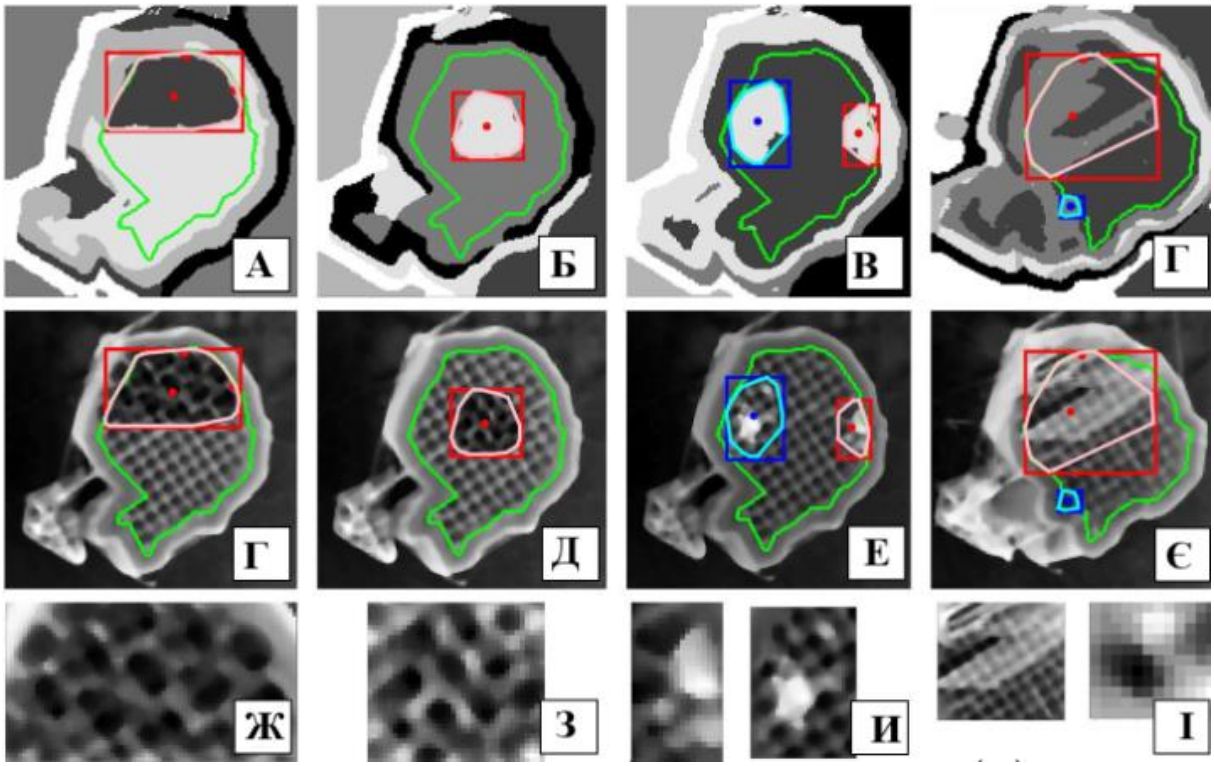

Рис. 6.6 – Виявлені області з аномальною текстурою:
(А-Г) – сегментовані текстури; (Г-Є) – виявлені дефекти;
(Ж-І) – обрізані області інтересу з дефектами (не в масштабі).

Одним із запропонованих методів усунення дефектів є операція сглажування, після якої виконується повторна сегментація та аналіз текстури. Якщо дефектна область зберігається, її необхідно розплавити та заповнити матеріалом на 100%.

На цьому етапі аналізу текстури, окрім прямої сегментації, зображення дефектних областей зберігаються в базі даних для подальшого маркування та класифікації.

Таким чином, алгоритм машинного зору, використаний у цьому дослідженні, є основою для створення алгоритму корекції збоїв для всіх методів адитивного виробництва.

ВИСКОВКИ

Розробка методу удосконалення параметрів FDM-друку на основі машинного зору є комплексною та складною задачею, оскільки інують проблеми, які не вирішені до кінця, а саме:

1. однозначно візуально визначити тип помилки,
2. встановити прямий причинно-наслідковий зв’язок між типом помилки та пов’язаним параметром друку,
3. заздалегідь визначити, яке значення параметра (коефіцієнти масштабування, швидкість подачі, температура, швидкість переміщення тощо) слід використовувати для корекції збою.

Вищенаведені експерименти базуються на припущенні, що механічні параметри принтера (стабільність збирання, наявність мастила в рухомих частинах, натяг ременів, електрична напруга драйверів крокових двигунів тощо) налаштовані та відкалібровані оптимально. Отримані експериментальні результати для випадку номінального режиму друку без відхилень дозволяють визначити точність і допуски адаптивного алгоритму.

Таким чином, на цьому етапі дослідження представлена робота є скоріше інтелектуальним інструментом для призупинення друку, що дозволяє заощаджувати час і матеріал, ніж повноцінним методом удосконалення параетрів для підвищення якості друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Graves S., Rocholl J.C. Style Delta Robot Kinematics. URL: https://reprap.org/wiki/File:Rostock_Delta_Kinematics_3.pdf. (Дата звернення: 02.09.2024).

2. Camera calibration with OpenCV: asymmetrical circle calibration pattern. URL: https://docs.opencv.org/2.4/_downloads/acircles_pattern.png. (Дата звернення: 14.09.2024).

3. Heikkila J. Geometric camera calibration using circular control points. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(10), 1066-1077. URL: https://doi.org/10.1109/34.879788. (Дата звернення: 03.10.2024).

4. OpenCV: Template Matching. URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. (Дата звернення: 11.11.2024).

5. Crivellaro A., Lepetit V. Robust 3D Tracking with Descriptor Fields, IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, 3414-3421. URL: https://doi.org/10.1109/CVPR.2014.436. (Дата звернення: 23.09.2024).

6. Canny J. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 679-698. URL: https://doi.org/10.1109/TPAMI.1986.4767851. (Дата звернення: 10.10.2024).

7. Chen Y., Medioni G. Object modelling by registration of multiple range images, Image and Vision Computing, 10(3), 145-155. URL: https://doi.org/10.1016/0262-8856(92)90066-C. (Дата звернення: 01.11.2024).

8. Besl P.J., McKay N.D. A Method for Registration of 3-D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(2), 239-256. URL: https://doi.org/10.1109/34.121791. (Дата звернення: 05.09.2024).

9. Varma M., Zisserman A. Texture classification: are filter banks necessary? Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, Madison, WI, USA, pp. II-691. URL: https://doi.org/10.1109/CVPR.2003.1211534. (Дата звернення: 30.09.2024).

10. Zhu S.-C., Guo C.-E., Wang Y., Xu Z. What are Textons? International Journal of Computer Vision, 62(1/2), 121-143. URL: https://doi.org/10.1007/s11263-005-4638-1. (Дата звернення: 12.11.2024).

11. Liu L., Chen J., Fieguth P., Zhao G., Chellappa R., Pietikäinen M. From BoW to CNN: Two Decades of Texture Representation for Texture Classification. International Journal of Computer Vision, 127(1), 74-109. URL: https://doi.org/10.1007/s11263-018-1125-z. (Дата звернення: 17.10.2024).

12. Thomas L.S.V., Gehrig J. Multi-template matching: a versatile tool for object localization in microscopy images. BMC Bioinformatics, 21, 44. URL: https://doi.org/10.1186/s12859-020-3363-7. (Дата звернення: 09.10.2024).

13. Hernandez-Rivera E., Coleman S.P., Tschopp M.A. Using similarity metrics to quantify differences in high-throughput datasets: application to X-ray diffraction patterns. ACS Comb Sci., 19(1), 25–36. URL: https://doi.org/10.1021/acscombsci.6b00142. (Дата звернення: 20.09.2024).

14. Leung T., Malik J. Representing and recognizing the visual appearance of materials using three-dimensional textons. International Journal of Computer Vision, 43, 29-44. URL: https://doi.org/10.1023/A:1011126920638. (Дата звернення: 28.10.2024).

15. Bishop C. Pattern Recognition and Machine Learning. Springer, New York, USA, 2006.

16. Dempster A.P., Laird N.M., Rubin D.B. Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. Ser. B, 39 (1977), 1-38.

17. McLachlan G.J., Krishnan T., Ng S.K. The EM Algorithm, Papers, No. 2004, 24, Humboldt-Universität zu Berlin, Center for Applied Statistics and Economics (CASE), Berlin, 2004.

18. Nielsen F. Hierarchical Clustering, in: Introduction to HPC with MPI for Data Science, Undergraduate topics in computer science, Springer, Basel, Switzerland, 2016, pp. 195-211. URL: https://doi.org/10.1007/978-3-319-21903-5_8. (Дата звернення: 07.11.2024).

19. Low Polygonal STL Fox model. URL: https://www.thingiverse.com/thing:937740, CC BY-NC-SA 3.0 license. (Дата звернення: 06.09.2024).

20. Zhu S.-C., Guo C.-E., Wang Y., Xu Z. What are Textons? International Journal of Computer Vision, 62(1/2), 121-143. URL: https://doi.org/10.1007/s11263-005-4638-1. (Дата звернення: 15.11.2024).

21. С.С. Коротков, Н.О. Лащевська, І.Б. Кузьміч, В.В. Волошин Інтеграція камери в систему FDM-друку для покращення якості друку // журнал “ТЕЛЕКОМУНІКАЦІЙНІ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ”.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

«МЕТОД УДОСКОНАЛЕННЯ ПАРАМЕТРІВ FDM-ДРУКУ НА ОСНОВІ МАШИННОГО ЗОРУ»

Виконав: студент групи ПДМ-63 Волошин Віталій Віталійович

Керівник: доктор філософії(PHD), старший викладач кафедри ІПЗ Залива Віталій Вікторович

Київ - 2025

Мета, об’єкт, предмет дослідження

Метою даного дослідження є удосконалення параметрів FDM-друку на основі машинного зору, що дозволить покращити якість друку та підвищити стабільність процесу.

Об’єкт дослідження процес друку методом наплавлення розплавленого матеріалу (FDM) з використанням адитивних технологій.

Предмет дослідження метод удосконалення параметрів FDM-друку на основі технологій машинного зору для покращення якості друкованих виробів.

Актуальність роботи

Сучасні підходи до контролю якості FDM-друку переважно базуються на ручному моніторингу або використанні попередньо заданих параметрів друку. Це обмежує можливості адаптації процесу в реальному часі та призводить до значних втрат матеріалів, часу і ресурсів.

Така інтеграція дозволяє:

Виявляти дефекти на ранніх стадіях друку.

Автоматично коригувати параметри друку в режимі реального часу.

Підвищувати точність та стабільність процесу.

Актуальність роботи полягає у необхідності розробки методу, який дозволить зменшити кількість дефектів, оптимізувати витрати матеріалів і часу, а також покращити якість друкованих виробів.

3

Проблематика FDM-друку та параметри, що впливають на якість

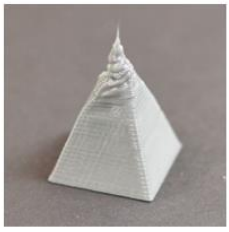

Рис. 4.1 – Неправильна температура сопла

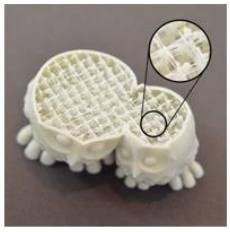

Рис. 4.2 – Дефект заповнення

Рис. 4.3 – Неправильна температура стола/сопла для першого шару/вимкнений обдув

Рис. 4.4 – Неправильна швидкість друку

Рис. 4.5 – Недоекструзія пластику

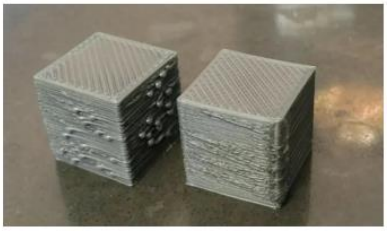

Рис. 4.6 – Переекструзія пластику

4

Метод удосконалення параметрів FDM-друку на основі машинного зору

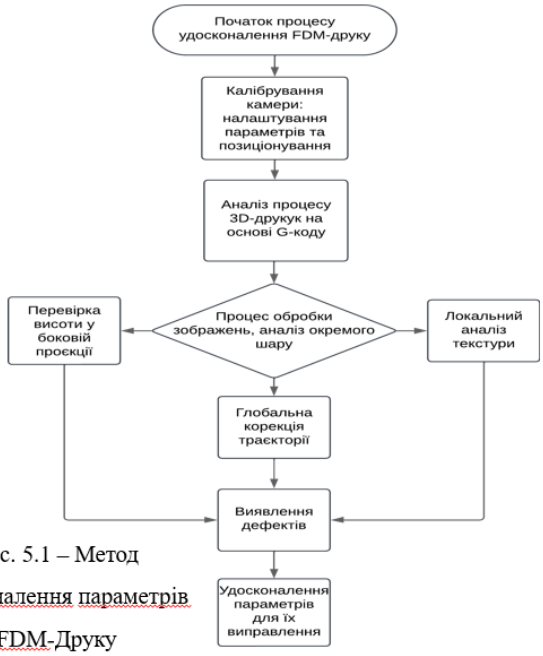

Рис. 5.1 – Метод удосконалення параметрів FDM-Друку

5

Калібрування камери та позиціонування

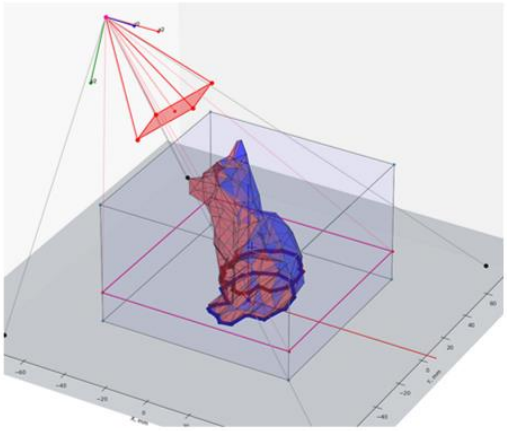

Рис. 6.1 – Положення камери відносно моделі

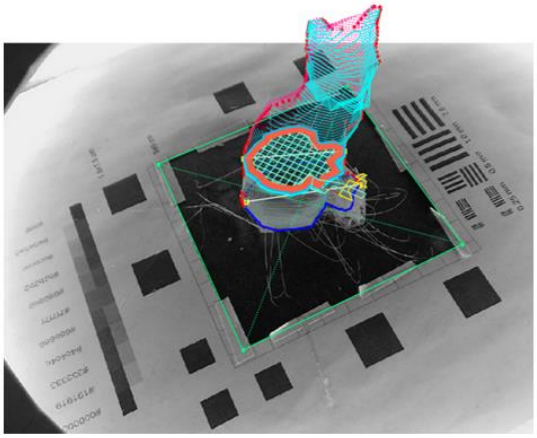

Рис. 6.2 – Траєкторії G-коду, спроектовані на вихідний кадр зображення

6

Аналіз процесу 3D-друку на основі G-коду

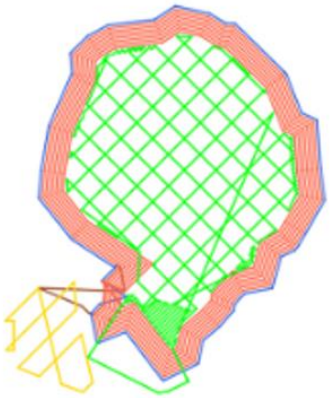

Рис. 7.1 – Вихідний G-код, розділений на шари: синій(зовнішня стінка), червоний(внутрішня стінка), зелений(заповнення), і жовтий(підтримка)

```
1. G1 X-9.92 Y-10.28 Z0.455 E0.36 F360
2. G1 X-9.98 Y-11.04 Z0.45 E0.408
3. G1 X-11.38 Y-12.53 Z0.457 E0.537
4. G1 X-11.16 Y-14.11 Z0.442 E0.637
5. G1 X-9.55 Y-15.55 Z0.408 E0.772
6. G1 X-5.28 Y-17.24 Z0.335 E1.062
7. G1 X-5.02 Y-17.13 Z0.332 E1.079
8. G1 X-4.71 Y-17.54 Z0.325 E1.111
9. G1 X-1.7 Y-20.77 Z0.257 E1.389
10. G1 X2.04 Y-20.05 Z0.211 E1.628
11. G1 X3.02 Y-19.58 Z0.201 E1.696
12. G1 X4.62 Y-11.52 Z0.244 E2.213
13. G1 X6.29 Y-10.1 Z0.232 E2.351
```

Рис. 7.2 – Приклад G-коду, де G1 - Команда лінійного перемішення, X, Y, Z – координати, до яких необхідно переміститися, E – величина екструзії (обсяг витискуваного матеріалу), F – швидкість перемішення (у міліметрах на хвилину).

7

Процес обробки зображень

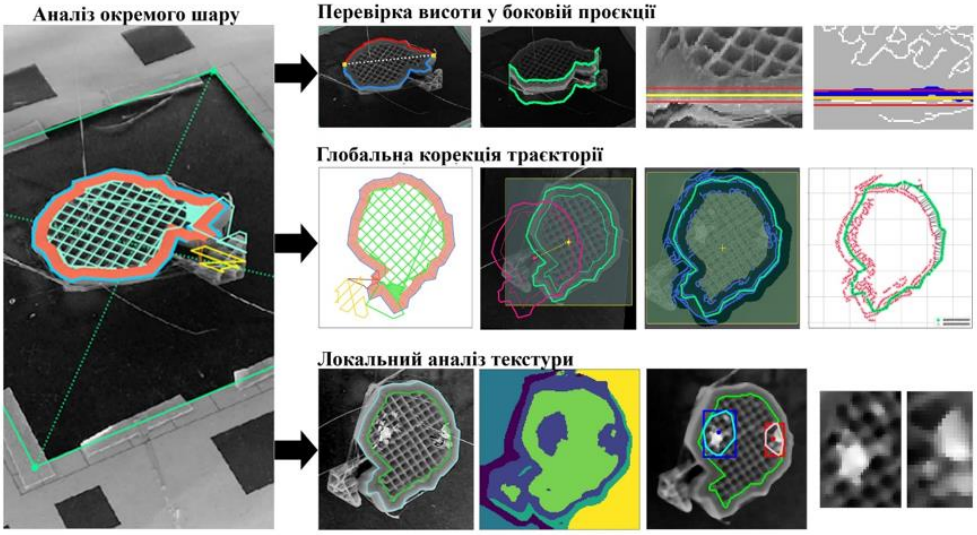

Рис. 8.1 – Конвеєр обробки зображень

8

Виявлення друкарських дефектів та удосконалення параметрів друку, для їх виправлення

Тип помилки	Стратегія виявлення	Зміна параметрів
Нестача філамента	Перевірка вертикального рівня + алгоритми MTM та ICP	Призупинити друк / Звітувати
Заблоковане сопло	Перевірка вертикального рівня + алгоритми MTM та ICP	Підвищити температуру сопла; Повторити попередній шар
Відсутній шар	Перевірка вертикального рівня + алгоритми MTM та ICP	Повторити попередній шар
Втрата точності розмірів	Алгоритми MTM та ICP	Оновити координати G-коду
Проблема з вирівнюванням платформи	Сегментація текстури	Призупинити друк / Звітувати; Ручна перекалібровка
Проблема адгезії	Перевірка вертикального рівня початкового шару	Підвищити температуру платформи; Звітувати у разі критичного вертикального відхилення
Погане прилягання друку до платформи	Перевірка вертикального рівня + алгоритми MTM та ICP	Підвищити температуру платформи; Призупинити друк/Звітувати
Зсув або згин друку	Перевірка вертикального рівня + алгоритми MTM та ICP	Оновити координати G-коду
Слабке або недостатнє заповнення	Сегментація текстури	Підвищити температуру насадки та швидкість подачі
Деформоване заповнення	Сегментація текстури	Змінити температуру сопла та швидкість подачі
Залишки обгорілого філамента	Сегментація текстури	Виконати вирівнювання
Неповне заповнення	Сегментація текстури	Процедура заміни проблемної області
Низька якість поверхні над підтримкою	Сегментація текстури	Змінити швидкість подачі
Щілини між заповненням і оболонкою	Сегментація текстури	Змінити швидкість подачі

Табл. 9.1 – Цільові відмови та коригувальні дії

9

Експериментальні результати

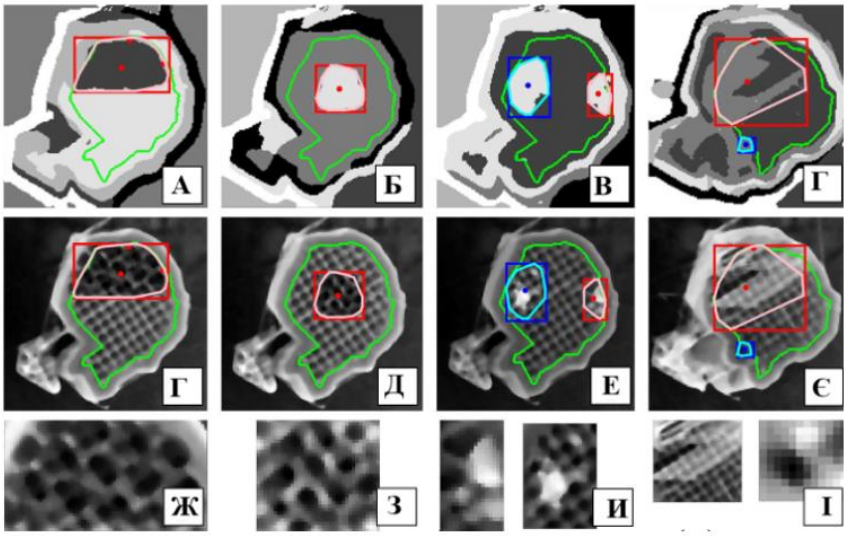

Рис. 10.1 – Виявлені області з аномальною текстурою:

(А-Г) – сегментовані текстури;

(Г-Є) – виявлені дефекти;

(Ж-І) – обрізані області інтересу з дефектами (не в масштабі).

10

Порівняльний аналіз

•Вартість 1 кг PLA-філаменту: 450 грн.
•Вага філаменту на одну модель: 16.5 г (або 0.0165 кг).
•Час друку однієї моделі: 43 хв.

Критерій	Метод на основі машинного зору	Метод генетичних алгоритмів (ГА)
Частка браку	5%	30%
Загальна вага (кг)	0.017325	0.02145
Вартість філаменту (грн)	7.796	9.653
Час друку (хв)	43	49.45

Табл. 11.1 – Порівняння двох методів

11

Висновки

В ході виконання магістерської роботи були зроблені наступні висновки:

1. **Існують труднощі для розробки методу удосконалення параметрів FDM-друку**: Завдання є комплексним через труднощі з візуальним визначенням типу помилки, встановленням причинно-наслідкових зв'язків між помилками та параметрами друку, а також попереднім визначенням оптимальних параметрів для корекції помилок.

2. **Умови проведення експериментів**: Для досягнення коректних результатів передбачено, що всі механічні параметри принтера оптимально налаштовані. Це дозволило визначити точність і допустимі відхилення адаптивного алгоритму в номінальних умовах друку.

3. **Результати дослідження**: На даному етапі метод є інструментом для призупинення друку, спрямованим на економію часу і матеріалів, але ще не досяг статусу повноцінної системи оптимізації параметрів для підвищення якості друку. Водночас, його ефективність у порівнянні з іншими методами становить у середньому на 10–25%, що вже є значним результатом і відкриваю перспективу для подальших досліджень.

12

Апробація

Статті:

1. С.С. Коротков, Н.О. Лащевська, І.Б. Кузьміч, В.В. Волошин

Інтеграція камери в систему FDM-друку для покращення якості друку // подана до друку

Тези доповідей:

1. Волошин В.В., Залива В.В. Проблематика FDM-друку. // Всеукраїнська Науково-практична конференція «Проблеми комп'ютерної інженерії». – Київ: ДУІКТ, 2024.

13

ДЯКУЮ ЗА УВАГУ!

14

ДОДАТОК Б. ФРАГМЕНТИ ОСНОВНИХ ПРОГРАМНИХ МОДУЛІВ

```
# Імпортуємо бібліотеку numpy для роботи з масивами та числовими операціями
import numpy as np
np.bool = np.bool_

# Імпортуємо модулі для роботи з STL-файлами (коментаровано, якщо не потрібні)
#from stl import mesh

# Імпортуємо модуль sys для роботи з системними викликами
import sys

# Імпортуємо OpenCV для роботи з обробкою зображень
import cv2

# Імпортуємо бібліотеку time для вимірювання часу виконання
import time

# --- Для роботи з STL-файлами
import meshcut  # Для виконання розрізів STL-файлів
from stl import mesh  # Для читання та роботи з STL-об'єктами
import numpy.linalg as la  # Для лінійної алгебри (обчислення векторів, матриць тощо)

# --- Для роботи з послідовним монітором (Serial Monitor)
import threading  # Для багатопоточності
import queue  # Для обробки черг
import serial  # Для взаємодії з послідовним портом

# --- Для роботи з парсингом G-code
from pygcode import *  # Основна бібліотека для парсингу G-code
from pygcode import Line  # Клас для роботи з рядками G-code
from pygcode import Machine, GCodeRapidMove  # Для управління машиною та швидкими рухами

# --- Імпортуємо модулі з PyQt5 для створення графічного інтерфейсу користувача (GUI)
from PyQt5.QtCore import *  # Основні класи ядра PyQt5
from PyQt5.QtGui import *  # Для графічного оформлення (іконки, шрифти тощо)

# Додаткові модулі PyQt5 для створення елементів GUI
from PyQt5.QtCore import Qt, QSize  # Для визначення параметрів GUI (розміри, положення)
from PyQt5 import QtWidgets, QtGui  # Основні віджети для GUI
from PyQt5.QtWidgets import QApplication, QWidget, QSlider, QComboBox, QInputDialog, QLineEdit, QFileDialog
from PyQt5.QtWidgets import QMainWindow, QPlainTextEdit, QCheckBox

# Для роботи з іконками в інтерфейсі
from PyQt5.QtGui import QIcon

# --- Імпортуємо Matplotlib для візуалізації
from matplotlib import pyplot as plt  # Основний модуль для побудови графіків
from matplotlib.patches import Polygon  # Для роботи з багатокутниками

# Модулі для роботи з тривимірною графікою
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import Axes3D  # Для побудови тривимірних графіків
from matplotlib.collections import PolyCollection  # Колекція для 3D-об'єктів
```

```
from matplotlib import colors as mcolors  # Робота з кольорами
import matplotlib.cm as cm  # Карти кольорів
from mpl_toolkits.mplot3d.art3d import Poly3DCollection, Line3DCollection  # Колекції для полігонів і ліній у 3D

# Для інтеграції Matplotlib у GUI PyQt5
from matplotlib.backends.qt_compat import QtCore, QtWidgets, is_pyqt5
if is_pyqt5():
    from matplotlib.backends.backend_qt5agg import (
        FigureCanvas, NavigationToolbar2QT as NavigationToolbar)  # Інструменти навігації
else:
    from matplotlib.backends.backend_qt4agg import (
        FigureCanvas, NavigationToolbar2QT as NavigationToolbar)

# Для створення графіків у Matplotlib
from matplotlib.figure import Figure

# Імпортуємо PIL (Pillow) для роботи із зображеннями (обробка, збереження, зміна формату)
from PIL import Image

#====================================================================================

# Глобальні змінні

# Флаг, що використовується для відстеження стану "TT"
flag_TT = 0

# Флаг, що сигналізує про готовність до друку
flag_READY_TO_PRINT = 0

# Флаг, що вказує на команду до початку друку
flag_COMMAND_TO_PRINT = 0

# Список для збереження ліній команд G-code, які потрібно відправити
line_command_bank = []

# Строкова змінна для зберігання команди, що буде відправлена
cmd_to_send = ""

# Лічильник для відстеження номеру поточної лінії G-code
gcode_line_number = 0
# ------------------------------------------------------------


class ThreadingTasks(QtCore.QObject):
    def __init__(self):
        super().__init__()
        print("ThreadingTasks initialized")

        # --- Ініціалізація необхідних змінних
        self.rcv = []  # Список для отриманих даних
        self.timer_id = 0  # Ідентифікатор таймера
        # self.th_port = serial.Serial("/dev/ttyACM0", baudrate=9600, timeout=0)  # Послідовний порт (налаштувати за потреби)

    # --- Метод для тестування багатопоточної роботи
    def test_func(self, numbers):
        """Тестовий метод для обчислення кубів чисел"""
        print("test func activated")
        for n in numbers:
            time.sleep(1)  # Затримка у 1 секунду
            print('cube:', n * n * n)

    # --- Обробка подій таймера
    def timerEvent(self, event):
        """Обробка отриманих даних через таймер"""
        global flag_READY_TO_PRINT
```

```
        self.rcv = self.th_port.read(255).decode('utf-8')
# Читання даних з порту
        print(self.rcv)  # Виведення отриманих даних
        str_split = self.rcv.split()  # Розділення
отриманих даних
        for item in str_split:
            if item == 'ok':  # Якщо отримано "ok"
                print('=== OK ===')
                flag_READY_TO_PRINT = 1  # Сигнал
готовності до друку
            else:
                print('=== NOT OK ===')
                flag_READY_TO_PRINT = 0


    # --- Метод друку шару
    def printingLayer(self):
        """Відправка команд G-code для друку
шару"""
        print("Layer is printing")
        global flag_COMMAND_TO_PRINT,
flag_READY_TO_PRINT
        global line_command_bank, cmd_to_send,
gcode_line_number


        while gcode_line_number <
len(line_command_bank):  # Перевірка, чи
залишилися команди
            if flag_READY_TO_PRINT == 1:  # Якщо
пристрій готовий до друку
                flag_READY_TO_PRINT = 0  #
Скидання флага готовності
                cmd_to_send =
line_command_bank[gcode_line_number]  #
Поточна команда
                self.th_port.write(('\n' + str(cmd_to_send) +
'\n').encode())  # Надсилання команди

print(f'[{gcode_line_number}/{len(line_command_b
ank) - 1}] {cmd_to_send}')
                gcode_line_number += 1  # Перехід до
наступної команди
```

```
class App(QWidget):


    def __init__(self):
        super().__init__()
        self.title = 'СИСТЕМА КЕРУВАННЯ'
        self.th_obj = ThreadingTasks()
        self.initUI()


    def initUI(self):
        super().__init__()
        self._main = QtWidgets.QWidget()
        self.setWindowTitle(self.title)


        # ---------- КОМПОНЕНТИ ----------------------
----------------------------


        # Порожній простір
        self.lbl_blank_space = QtWidgets.QLabel(' ')


        # Логотип
        self.lbl_LOGO =
QtWidgets.QLabel('КЕРУВАННЯ')
        self.lbl_LOGO.setStyleSheet("QLabel {color:
#000000}")
        self.lbl_LOGO.setFont(QtGui.QFont('Sans', 22,
QtGui.QFont.Bold))

self.lbl_LOGO.setAlignment(QtCore.Qt.AlignCenter
| QtCore.Qt.AlignVCenter)


        # Поле для введення імені файлу
        self.textbox_FILENAME =
QtWidgets.QLineEdit()
        self.btn_SET_FILENAME =
QtWidgets.QPushButton('&Встановити ім\'я
файлу')


        # Кнопка для завантаження STL-файлу
```

```
        self.btn_LOAD_STL = QtWidgets.QPushButton('&Завантажити STL-файл')

        # Кнопка для завантаження GCode
        self.btn_PARSE_GCODE = QtWidgets.QPushButton('&Завантажити GCode')

self.btn_PARSE_GCODE.setStyleSheet("QPushButton {background-color: #FEE101;}")

        # Текстове поле для аналізу GCode
        self.textfield_GCODE_PARSER = QPlainTextEdit(self)

self.textfield_GCODE_PARSER.insertPlainText("Аналізатор GCode\n")

self.textfield_GCODE_PARSER.setReadOnly(True)

self.textfield_GCODE_PARSER.setLineWrapMode(QPlainTextEdit.NoWrap)

self.textfield_GCODE_PARSER.setFont(QtGui.QFont('Courier', 6))

        # Інформація про GCode
        self.lbl_GCODE_INFO_SECTION = QtWidgets.QLabel('Загальна кількість шарів = ...\nАктивний шар = ...\nКількість команд = ...')

self.lbl_GCODE_INFO_SECTION.setStyleSheet("QLabel {; color: #000000;}")

self.lbl_GCODE_INFO_SECTION.setFont(QtGui.QFont('Courier', 10))

        # Повзунок для шарів GCode
        self.lbl_GCODE_LAYER_SLIDER = QtWidgets.QLabel('Повзунок шарів GCode')

self.lbl_GCODE_LAYER_SLIDER.setStyleSheet("QLabel {background-color: #CCCCCC; color: #FFFFFF}")

self.lbl_GCODE_LAYER_SLIDER.setFont(QtGui.QFont('Sans', 10, QtGui.QFont.Bold))

self.lbl_GCODE_LAYER_SLIDER.setAlignment(QtCore.Qt.AlignCenter | QtCore.Qt.AlignVCenter)

        self.slider_L = QSlider(Qt.Horizontal)
        self.slider_L.setFocusPolicy(Qt.StrongFocus)

self.slider_L.setTickPosition(QSlider.TicksAbove)
        self.slider_L.setTickInterval(2)
        self.slider_L.setSingleStep(1)

        # Кнопки для візуалізації і друку активного шару
        self.btn_VISUALIZE_LAYER = QtWidgets.QPushButton('&Візуалізувати активний шар')
        self.btn_PRINT_ACTIVE_LAYER = QtWidgets.QPushButton('&Друкувати активний шар')

self.btn_PRINT_ACTIVE_LAYER.setStyleSheet("QPushButton {background-color: #FEE101;}")

        # Кнопка запуску серійного монітора
        self.btn_RUN_SERIAL_MONITOR = QtWidgets.QPushButton('&Запустити серійний монітор')

        # Поле для серійного монітора
        self.textfield_SERIAL_MONITOR = QPlainTextEdit(self)

self.textfield_SERIAL_MONITOR.insertPlainText("Серійний монітор\n")

self.textfield_SERIAL_MONITOR.setLineWrapMode(QPlainTextEdit.NoWrap)

self.textfield_SERIAL_MONITOR.setFont(QtGui.QFont('Courier', 6))

        # Заголовок командного блоку

self.lbl_SERIAL_COMMAND_BLOCK_HEAD = QtWidgets.QLabel('Командний блок')
```

```
self.lbl_SERIAL_COMMAND_BLOCK_HEAD.setStyleSheet("QLabel {background-color: #CCCCCC; color: #FFFFFF}")

self.lbl_SERIAL_COMMAND_BLOCK_HEAD.setFont(QtGui.QFont('Sans', 10, QtGui.QFont.Bold))

self.lbl_SERIAL_COMMAND_BLOCK_HEAD.setAlignment(QtCore.Qt.AlignCenter | QtCore.Qt.AlignVCenter)


    # Інформація про температуру
    self.lbl_SERIAL_MONITOR_TEMPERATURE = QtWidgets.QLabel('Температура = ')

self.lbl_SERIAL_MONITOR_TEMPERATURE.setStyleSheet("QLabel {; color: #000000;}")

self.lbl_SERIAL_MONITOR_TEMPERATURE.setFont(QtGui.QFont('Courier', 10))


    # Поле введення команд
    self.line_edit_COMMAND_LINE = QLineEdit(self)
    self.btn_COMMAND_SEND = QtWidgets.QPushButton('&Відправити команду')


    # Кнопка відкриття камери
    self.btn_OPEN_CAMERA = QtWidgets.QPushButton('&Відкрити камеру')

self.btn_OPEN_CAMERA.setStyleSheet("QPushButton {background-color: #A7A7AD;}")




    ## ---------- КОМПОНОВАННЯ ----------
    layout_MAIN_H = QtWidgets.QHBoxLayout()
    layout_V0 = QtWidgets.QVBoxLayout()
    layout_V1 = QtWidgets.QVBoxLayout()
    layout_V2 = QtWidgets.QVBoxLayout()
    layout_V3 = QtWidgets.QVBoxLayout()
    layout_H_X = QtWidgets.QHBoxLayout()
    layout_H_Y = QtWidgets.QHBoxLayout()
    layout_H_Z = QtWidgets.QHBoxLayout()
    layout_instant_XYZ = QtWidgets.QHBoxLayout()
    layout_H_FILENAME = QtWidgets.QHBoxLayout()


    # -------- V0
    layout_V0.addWidget(self.lbl_LOGO)


layout_H_FILENAME.addWidget(self.textbox_FILENAME)

layout_H_FILENAME.addWidget(self.btn_SET_FILENAME)
    layout_V0.addLayout(layout_H_FILENAME)


    layout_V0.addWidget(self.btn_LOAD_STL)

layout_V0.addWidget(self.btn_PARSE_GCODE)

layout_V0.addWidget(self.lbl_GCODE_INFO_SECTION)

layout_V0.addWidget(self.lbl_GCODE_LAYER_SLIDER)
    layout_V0.addWidget(self.slider_L)

layout_V0.addWidget(self.textfield_GCODE_PARSER)

layout_V0.addWidget(self.btn_VISUALIZE_LAYER)

layout_V0.addWidget(self.btn_PRINT_ACTIVE_LAYER)

layout_V0.addWidget(self.btn_RUN_SERIAL_MONITOR)

layout_V0.addWidget(self.textfield_SERIAL_MONITOR)

layout_V0.addWidget(self.lbl_SERIAL_COMMAND_BLOCK_HEAD)
```

```
layout_V0.addWidget(self.line_edit_COMMAND_LINE)

layout_V0.addWidget(self.btn_COMMAND_SEND)
        #
layout_V0.addWidget(self.lbl_SERIAL_MONITOR_TEMPERATURE)

layout_V0.addWidget(self.btn_OPEN_CAMERA)


        # -------- MAIN LAYOUT
        layout_MAIN_H.addLayout(layout_V0)
        layout_MAIN_H.addLayout(layout_V1)
        layout_MAIN_H.addLayout(layout_V2)
        layout_MAIN_H.addLayout(layout_V3)
        self.setLayout(layout_MAIN_H)


        # ---------- З'ЄДНАННЯ ----------
------------------------------

self.btn_SET_FILENAME.clicked.connect(self.btn_SET_FILENAME_function)  # Встановлення імені файлу

self.btn_LOAD_STL.clicked.connect(self.btn_LOAD_STL_function)         # Завантаження STL-файлу


self.btn_PARSE_GCODE.clicked.connect(self.btn_PARSE_GCODE_function)    # Завантаження GCode

self.slider_L.valueChanged.connect(self.slider_L_change)              # Зміна активного шару

self.btn_VISUALIZE_LAYER.clicked.connect(self.btn_VISUALIZE_LAYER_function)  # Візуалізація шару

self.btn_PRINT_ACTIVE_LAYER.clicked.connect(self.btn_PRINT_ACTIVE_LAYER_function)  # Друк активного шару



self.btn_RUN_SERIAL_MONITOR.clicked.connect(self.btn_RUN_SERIAL_MONITOR_function)  # Запуск серійного монітора

self.btn_COMMAND_SEND.clicked.connect(self.btn_COMMAND_SEND_function)            # Відправлення команди



self.btn_OPEN_CAMERA.clicked.connect(self.btn_OPEN_CAMERA_function)  # Відкриття камери


    # ---------- ЗМІННІ ----------
------------------------------

    font = cv2.FONT_HERSHEY_SIMPLEX


    # ---- Прапори (flags)
    flag_UPD_CANVAS = 1              # Оновлення полотна
    flag_STL_LOADED = 0              # Завантаження STL
    flag_GCODE_PARSED = 0            # Розбір GCode


    # ---- Параметри GCode
    parsed_Num_of_layers = 0         # Кількість шарів
    active_layer = 0                 # Активний шар
    LAYER_THICKNESS = 0.4            # Товщина шару (мм)


    layer_bank = []                  # Банк шарів
    word_bank = []                   # Банк команд
    line_bank = []                   # Банк ліній


    X_active_bank = []  # Координати X активного шару
    Y_active_bank = []  # Координати Y активного шару
    Z_active_bank = []  # Координати Z активного шару
    G_active_bank = []  # G-команди
    E_active_bank = []  # E-команди
    F_active_bank = []  # F-команди
```

```
    command_bank = []  # Команди для активного шару


    # ---- Глобальні змінні
    XY_data = []
    SIDE_data = []


    tensor_XY = []
    tensor_SIDE = []
    stl_file = []


    filename = "default"  # Стандартне ім'я файлу


    # ---- Для OpenCV
    cMo = np.array([
        [5.96398265e-01, -2.34244280e-01, -2.12089702e+02],
        [2.21850690e-01,  1.00361026e+00, -2.02148236e+02],
        [2.28244003e-05,  7.88925038e-04,  1.00000000e+00]
    ])



    # ---------- ВИЗНАЧЕННЯ ФУНКЦІЙ ----------
    # Функція для встановлення імені файлу
    def btn_SET_FILENAME_function(self):
        self.filename = self.textbox_FILENAME.text()
        # Глобальний прапорець для перевірки зміни імені файлу
        global flag_TT
        if flag_TT == 0:
            flag_TT = 1
        else:
            flag_TT = 0


    # Функція для завантаження STL-файлу
    def btn_LOAD_STL_function(self):
        self.flag_STL_LOADED = 1
        figure = plt.figure()
        axes = mplot3d.Axes3D(figure)
        # Завантаження STL-файлу і додавання векторів на графік
        your_mesh = mesh.Mesh.from_file(self.filename + '.stl')
        self.stl_file = your_mesh


axes.add_collection3d(mplot3d.art3d.Poly3DCollection(your_mesh.vectors, alpha=0.2, facecolor='royalblue'))

axes.add_collection3d(mplot3d.art3d.Line3DCollection(your_mesh.vectors, alpha=0.3, linewidths=1, color='black', linestyle=':'))


        # Обробка STL-файлу для подальшої візуалізації
        verts = your_mesh.vectors.reshape(-1, 3)
        faces = np.arange(len(verts)).reshape(-1, 3)
        np.int = int
        verts, faces = meshcut.merge_close_vertices(verts, faces)
        mesh_plane = meshcut.TriangleMesh(verts, faces)


        plane_sect = (0, 0, self.active_layer * self.LAYER_THICKNESS)
        plane_norm = (0, 0, 1)
        plane_norm /= la.norm(plane_norm)
        stl_plane = meshcut.Plane(plane_sect, plane_norm)
        P = meshcut.cross_section_mesh(mesh_plane, stl_plane)


        # Додавання точок та ліній для активного шару
        for i in range(np.shape(P[0])[0]):
            axes.scatter(P[0][i][0], P[0][i][1], P[0][i][2], color='red', s=5)
            axes.plot([P[0][i][0], P[0][i-1][0]],
                      [P[0][i][1], P[0][i-1][1]],
```

```
                [P[0][i][2], P[0][i-1][2]], color='red',
linewidth=2)


        axes.text(50, 20, self.active_layer *
self.LAYER_THICKNESS, s=f'Активний шар
{self.active_layer}', fontsize=10)


        if self.flag_GCODE_PARSED == 1:
            for i in range(len(self.X_active_bank)):
                axes.plot([self.X_active_bank[i],
self.X_active_bank[i-1]],
                          [self.Y_active_bank[i],
self.Y_active_bank[i-1]],
                          [self.Z_active_bank[i],
self.Z_active_bank[i-1]], color='b')


        # Автоматичне масштабування
        scale = your_mesh.points.flatten('A')
        axes.auto_scale_xyz(scale, scale, scale)
        axes.set_xlabel('X, мм')
        axes.set_ylabel('Y, мм')
        axes.set_zlabel('Z, мм')
        axes.set_title('Модель STL')
        # Відображення графіку
        plt.show()


    # Функція для парсингу GCODE
    def btn_PARSE_GCODE_function(self):
        with open(self.filename + '.gcode', 'r') as fh:
            for line_text in fh.readlines():
                line = Line(line_text)
                w = line.block.words
                if np.shape(w)[0] == 0:
                    pass
                else:
                    self.word_bank.append(w)

self.layer_bank.append(self.parsed_Num_of_layers)
                    self.line_bank.append(line_text)
                if line.comment:
                    if
line.comment.text.startswith("LAYER:"):
                        self.parsed_Num_of_layers += 1
                        print(self.parsed_Num_of_layers)

self.lbl_GCODE_INFO_SECTION.setText(f"Загаль
на кількість шарів =
{self.parsed_Num_of_layers}\n"
                                            "Активний
шар = ...\n"
                                            "Кількість
команд = ...")
        self.flag_GCODE_PARSED = 1


    # Зміна активного шару через слайдер
    def slider_L_change(self):
        global line_command_bank
        global gcode_line_number
        gcode_line_number = 0
        self.active_layer = int(self.slider_L.value() *
self.parsed_Num_of_layers / 100)

self.lbl_GCODE_INFO_SECTION.setText(f"Загаль
на кількість шарів =
{self.parsed_Num_of_layers}\n"
                                    f"Активний шар =
{self.active_layer}\n"
                                    f"Кількість команд =
{len(line_command_bank)}")
        # Очищення банків даних для нового шару
        self.X_active_bank = []
        self.Y_active_bank = []
        self.Z_active_bank = []
        self.G_active_bank = []
        self.E_active_bank = []
        self.F_active_bank = []
        self.command_bank = []
        line_command_bank = []


        # Збирання даних для активного шару
        for i in range(len(self.layer_bank)):
            if self.layer_bank[i] == self.active_layer:
```

```
line_command_bank.append(self.line_bank[i])
            for j in range(len(self.word_bank[i])):
self.command_bank.append(str(self.word_bank[i][j])
)
                if
str(self.word_bank[i][j]).startswith('G'):
self.G_active_bank.append(float(str(self.word_bank[i
][j])[1:]))
                if
str(self.word_bank[i][j]).startswith('X'):
self.X_active_bank.append(float(str(self.word_bank[i
][j])[1:]))
                if
str(self.word_bank[i][j]).startswith('Y'):
self.Y_active_bank.append(float(str(self.word_bank[i
][j])[1:]))
                if
str(self.word_bank[i][j]).startswith('Z'):
self.Z_active_bank.append(float(str(self.word_bank[i
][j])[1:]))
                if
str(self.word_bank[i][j]).startswith('E'):
self.E_active_bank.append(float(str(self.word_bank[i
][j])[1:]))
                if str(self.word_bank[i][j]).startswith('F'):
self.F_active_bank.append(float(str(self.word_bank[i
][j])[1:]))

        # Виведення команд активного шару
        self.textfield_GCODE_PARSER.clear()
        for line in line_command_bank:
self.textfield_GCODE_PARSER.insertPlainText(line)

    # ---------- ФУНКЦІЇ КЕРУВАННЯ
ІНТЕРФЕЙСОМ ----------------------------------------
---

    # Функція для візуалізації активного шару
    def btn_VISUALIZE_LAYER_function(self):
        # Створення графіку
        fig = plt.figure(figsize=(6, 6), dpi=80)
        ax = fig.add_subplot(111, projection='3d')

        # Побудова ліній між точками активного
шару
        for i in range(len(self.X_active_bank)):
            ax.plot([self.X_active_bank[i],
self.X_active_bank[i-1]],
                    [self.Y_active_bank[i],
self.Y_active_bank[i-1]],
                    [self.Z_active_bank[i],
self.Z_active_bank[i-1]], color='b')

        # Налаштування міток і назви графіку
        ax.set_xlabel('X, мм')
        ax.set_ylabel('Y, мм')
        ax.set_zlabel('Z, мм')
        ax.set_title(f'Шар {self.active_layer}')
        plt.show()

    # Функція для друку активного шару
    def btn_PRINT_ACTIVE_LAYER_function(self):
        global flag_COMMAND_TO_PRINT
        flag_COMMAND_TO_PRINT = 1
        print("****")
        # Створення потоку для друку шару
        t2 = threading.Thread(name="Thread 2",
target=self.th_obj.printingLayer, args=())
        t2.setDaemon(True)
        t2.start()

    # Функція для запуску серійного монітора
    def
btn_RUN_SERIAL_MONITOR_function(self):
        # Запуск таймера для оновлення даних
серійного монітора
        self.th_obj.timer_id =
self.th_obj.startTimer(100,
timerType=QtCore.Qt.PreciseTimer)
```

```
    # Функція для відправки команди через серійний порт
    def btn_COMMAND_SEND_function(self):
        # Зчитування тексту команди з поля введення
        to_send = self.line_edit_COMMAND_LINE.text()
        self.port.write(('\n' + str(to_send) + '\n').encode())
        print(f'--> Відправлено команду: {to_send}')

    # Функція для відкриття камери
    def btn_OPEN_CAMERA_function(self):
        # Відкриття камери
        cap = cv2.VideoCapture(0)
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 800)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 600)

        if not cap.isOpened():
            print("Помилка відкриття відео потоку")
            return

        frame_width = int(cap.get(3))
        frame_height = int(cap.get(4))
        print(frame_width, frame_height)

        while cap.isOpened():
            ret, frame = cap.read()
            if ret:
                # Налаштування перспективи та візуалізація
                pts_top_1 = np.float32([[48, 122], [487, 35], [134, 445], [621, 337]])
                pts_top_2 = np.float32([[0, 0], [500, 0], [0, 500], [500, 500]])
                cM_top = cv2.getPerspectiveTransform(pts_top_1, pts_top_2)
                top_view = cv2.warpPerspective(frame[:, :, 0], cM_top, (500, 500))
                top_view = cv2.cvtColor(top_view, cv2.COLOR_GRAY2RGB)

                sc = 13.2
                sh = 240
                for i in range(len(self.X_active_bank)):
                    cv2.line(top_view,
                             (int(self.X_active_bank[i] * sc + sh + 10), int(self.Y_active_bank[i] * sc + sh - 5)),
                             (int(self.X_active_bank[i-1] * sc + sh + 10), int(self.Y_active_bank[i-1] * sc + sh - 5)),
                             (0, 0, 255), 1)

                cv2.putText(frame, "Основна камера: Повний огляд", (20, 30), self.font, 0.6, (255, 255, 255), 1, cv2.LINE_AA)
                cv2.imshow("Кадр", frame)
                cv2.moveWindow("Кадр", 400, 50)

                cv2.imshow("Вид зверху", top_view)
                cv2.moveWindow("Вид зверху", 1400, 50)

                # Вихід при натисканні 'q'
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    cap.release()
                    cv2.destroyAllWindows()
                    break

# ---------- ОСНОВНА ЧАСТИНА ПРОГРАМИ ----------
def main():
    app = QApplication(sys.argv)
    ex = App()
    ex.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```