Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Реконструкція 3D сцен за допомогою дифузійної моделі та 3D Gaussian splatting
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-24-2
Іван ЧУХРАН
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Євгеній БОДЯНСЬКИЙ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Чухрану Івану Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реконструкція 3D сцен за допомогою дифузійної моделіта 3D Gaussian splatting

затверджена наказом університету від 6 квітня 20 26 р. № 269Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 22 травня 20 26 р.

3. Вихідні дані до роботи передоброблені датасети Objaverse та NeRF Synthetic, засоби для розгортання, налаштування та навчання 3D Gaussian Splatting та дифузійної нейронної мережі

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Методи та засоби розв'язання задачі
3) Програмна реалізація методу
4) Експериментальна перевірка методу

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	06.04.2026	виконано
2	Аналіз предметної галузі	10.04.2026	виконано
3	Постановка задачі дослідження	16.04.2026	виконано
4	Моделювання і тренування моделей	12.05.2026	виконано
5	Написання записки	13.05.2026	виконано
6	Перевірка на академічний плагіат	15.05.2026	виконано
7	Нормоконтроль	16.05.2026	виконано
8	Рецензування	17.05.2026	виконано
9	Підготовка презентації та доповіді	18.05.2026	виконано
10	Попередній захист	21.05.2026	виконано
11	Захист перед ЕК	22.05.2026	

Дата видачі завдання 6 квітня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Євгеній БОДЯНСЬКИЙ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 72 с., 8 рис., 5 табл., 1 дод., 28 джерел.

ДИФЕРЕНЦІЙОВАНИЙ РЕНДЕРИНГ, ДИФУЗІЙНА ГЕНЕРАТИВНА МОДЕЛЬ, ТРИВИМІРНА РЕКОНСТРУКЦІЯ СЦЕН, GAUSSIAN SPLATTING.

Об’єкт дослідження – процес тривимірної реконструкції сцен за набором зображень.

Предмет дослідження – метод Gaussian Splatting з дифузійно-генерованою сіткою як геометричною апріорною моделлю.

Мета роботи – розроблення методу Diffusion-driven Gaussian Splatting для реконструкції тривимірних сцен.

Методи дослідження – Gaussian Splatting, дифузійні генеративні моделі, диференційований рендеринг, обробка хмар точок.

Запропоновано використання дифузійно-генерованої сітки як просторової апріорної моделі для Gaussian Splatting, що забезпечує глобально узгоджене геометричне представлення сцени. Розроблено та реалізовано метод дифузійно-керованого Gaussian Splatting; проведено експериментальне порівняння на всіх 8 сценах набору NeRF Synthetic; показано, що використання дифузійної сітки як апріорної моделі підвищує якість реконструкції відносно базової реалізації 3DGS на 4 з 8 сцен (середній приріст +1.77 дБ PSNR); виявлено залежність ефективності методу від типу геометрії сцени та визначено межі застосовності підходу.

ABSTRACT

Master’s thesis contains: 72 pp., 8 fig., 5 tabl., 1 ann., 28 references.

DIFFERENTIAL RENDERING, DIFFUSIVE GENERATIVE MODEL, GAUSSIAN SPLATTING, 3D SCENE RECONSTRUCTION.

Object of research – the process of three-dimensional reconstruction of scenes from a set of images with known camera parameters..

Subject of research – Gaussian Splatting method with a diffusion-generated mesh as a geometric prior.

Objective of work – develop a Diffusion-driven Gaussian Splatting method for the reconstruction of three-dimensional scenes.

Research methods – Gaussian Splatting, diffusion generative models, differential rendering, point cloud processing.

Diffusion-generated mesh as a spatial prior for Gaussian Splatting was proposed, which ensures a globally consistent geometric representation of the scene. A diffusion-driven Gaussian Splatting method was developed and implemented; conducted an experimental comparison on all 8 scenes of the NeRF Synthetic dataset; it is shown that using a diffusion mesh as a prior improves reconstruction quality relative to the baseline 3DGS implementation on 4 out of 8 scenes (average gain of +1.77 dB PSNR); the dependence of the method’s effectiveness on the scene geometry type is revealed, and the limits of the approach’s applicability are determined.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 9
Вступ 10
1 Аналіз предметної галузі 12
1.1 Еволюція методів представлення тривимірних сцен 12
1.2 Метод 3D Gaussian Splatting 13
1.2.1 Представлення сцени гаусовими примітивами 13
1.2.2 Колір та сферичні гармоніки 14
1.2.3 Проектування та растеризація 15
1.2.4 Оптимізація та адаптивний контроль щільності 16
1.2.5 Переваги та обмеження 3DGS 17
1.3 Методи диференційованої екстракції поверхонь 17
1.3.1 Поля знакових відстаней та ізоповерхні 18
1.3.2 Deep Marching Tetrahedra 18
1.3.3 Gaussian Opacity Fields 20
1.3.4 MILo 20
1.3.5 TeT-Splatting 21
1.3.6 Загальне обмеження сімейства DMTet 22
1.4 Дифузійні генеративні моделі 22
1.4.1 Деноізуючі дифузійні ймовірнісні моделі 22
1.4.2 Латентні дифузійні моделі 23
1.4.3 GaussianDreamer 24
1.4.4 DiffSplat 25
1.4.5 DiffGS 25
1.5 Гібридні підходи 26
1.5.1 SuGaR 26
1.5.2 MeshSplats 28
1.5.3 GaMeS 28
1.5.4 DeMapGS 29

1.6 Структурно-усвідомлені підходи: SAGS 29
1.7 Порівняльний аналіз розглянутих підходів 30
1.8 Постановка задачі 32
2 Методи та засоби розв'язання задачі 35
2.1 Метод дифузійно-керованого Gaussian Splatting 35
2.1.1 Знакові дистанційні функції 36
2.1.2 Архітектура знешумлювальної мережі 37
2.1.3 Навчання дифузійної мережі 37
2.1.4 Інференс: генерація сітки зі стиснутої хмари точок 38
2.2 Гібридне представлення гаусіанів 39
2.2.1 Розклад набору на поверхневі та вільні примітиви 39
2.2.2 Параметризація поверхневих гаусіанів 40
2.2.3 Параметризація вільних гаусіанів 41
2.3 Функції втрат 42
2.3.1 Фотометрична компонента 42
2.3.2 Регуляризатор близькості до поверхні 43
2.3.3 Регуляризатор вирівнювання нормалей 43
2.3.4 Сумарна функція втрат 44
2.3.5 Адаптивний контроль щільності за гібридних умов 45
2.3.6 Загальна схема методу 45
3 Програмна реалізація методу 47
3.1 Загальна архітектура та структура проєкту 47
3.2 Модуль gaussian_splatting 47
3.2.1 Клас GuassianModel 47
3.2.2 Растеризатор 48
3.2.3 Тренувальний цикл 49
3.3 Модуль diffusion 50
3.3.1 Реалізація DDPM/DDIM 50
3.3.2 Тривімірна знешумлювальна мережа U-Net 50
3.3.3 Кодувальна мережа Point-E 51

3.4 Передоброка даних Objaverse 52
3.4.1 Зчитування сцен COLMAP і NERF Synthetic 52
3.5 Інфраструктура експериментів 53
3.5.1 MLflow 53
4 Експериментальна перевірка методу 55
4.1 Датасет 55
4.2 Навчання дифузійної апріорної моделі 56
4.3 Навчання 3DGS 56
4.4 Метрики оцінювання 57
4.5 Кількісні результати на наборі NeRF Synthetic 58
4.6 Якісне порівняння 59
4.7 Часові характеристики 63
4.8 Обмеження методу 64
Висновки 65
Перелік джерел посилання 69
Додаток А Відомість кваліфікаційної роботи 72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГНМ – графова нейронна мережа;

ГП – гаусів примітив;

МК – маршируючі куби (Marching Cubes);

МТ – маршируючі тетраедри (Marching Tetrahedra);

НМ – нейронна мережа;

ШІ – штучний інтелект;

COLMAP – бібліотека з відкритим кодом для багатоглядової стереореконструкції та оцінки структури з рухомих камер;

DDIM – Denoising Diffusion Implicit Model – детермінований варіант семплування дифузійної моделі;

DDPM – Denoising Diffusion Probabilistic Model – деноізуюча дифузійна імовірнісна модель;

D-SSIM – Differentiable Structural Similarity – диференційований структурний індекс подібності;

LDM – Latent Diffusion Model – латентна дифузійна модель;

LPIPS – Learned Perceptual Image Patch Similarity – перцептивна метрика схожості зображень;

NeRF – Neural Radiance Field – нейронне поле яскравості;

3DGS – 3D Gaussian Splatting – метод тривимірної реконструкції сцен на основі гаусових примітивів.

ВСТУП

Тривимірна реконструкція сцен є фундаментальною задачею комп’ютерного зору та комп’ютерної графіки, що знаходить широке застосування в системах доповненої та віртуальної реальності, робототехніці, автономній навігації, кіновиробництві та розробці відеоігор. Задача полягає у відновленні повної тривимірної моделі сцени або об’єкта за набором двовимірних зображень з відомими або відновлюваними параметрами камер, що традиційно вирішується методами фотограмметрії, структури з рухомих камер (Structure-from-Motion, SfM) та стереофотограмметрії. Незважаючи на зрілість цих підходів, якість і ефективність фотореалістичного рендерингу відновлених сцен довгий час залишалися значним обмеженням.

Поява нейронних полів яскравості (Neural Radiance Fields, NeRF) [1] у 2020 році ознаменувала значний прорив у синтезі нових точок зору з фотореалістичною якістю. NeRF подає сцену як неявну нейронну функцію, що відображає тривимірну позицію та напрям спостереження на колір та густину, і синтезує зображення за допомогою диференційованого трасування об’ємних променів. Однак мала швидкість рендерингу – від кількох секунд до кількох хвилин на одне зображення – і необхідність переоптимізації для кожної нової сцени обмежували практичне застосування NeRF.

Метод 3D Gaussian Splatting (3DGS) [2], запропонований у 2023 році, подолав ці бар’єри, досягши рендерингу в реальному часі за конкурентоспроможної якості реконструкції завдяки явному поданню сцени у вигляді набору тривимірних гаусових примітивів та tile-based растеризації на GPU. 3DGS швидко став де-факто стандартом у задачах реконструкції сцен і породив великий клас похідних методів, спрямованих на подолання його власних обмежень.

Фундаментальне обмеження 3DGS – відсутність явного геометричного структурування. Гаусові примітиви оптимізуються незалежно, без урахування глобальної топології поверхні сцени, що призводить до неоптимального розміщення на межах поверхонь, артефактів у тонких структурах та підвищеної надмірності представлення. Структурно-усвідомлені підходи, зокрема SAGS [3], намагаються вирішити цю проблему шляхом побудови графу сусідства на основі алгоритму k найближчих сусідів (k-NN). Проте локальна зв’язність k-NN не здатна повністю передати глобальну топологію поверхні та демонструє обмежену узагальнюваність для різнорідних типів сцен.

Дифузійні генеративні моделі [4], [5] довели свою здатність до вивчення складних розподілів геометричних форм і знаходять дедалі ширше застосування як геометричні апріорні моделі у задачах 3D-генерації та реконструкції. Підходи DreamGaussian [6], GaussianDreamer [7], DiffSplat [8] та DiffGS [9] продемонстрували ефективність інтеграції дифузійних моделей з Gaussian Splatting для задачі генерації 3D-об’єктів, однак їхня спрямованість на генерацію, а не реконструкцію конкретних сцен, обмежує безпосереднє застосування цих методів.

Таким чином, розроблення методу, що використовує дифузійно-генеровану структуровану сітку як просторову апріорну модель для Gaussian Splatting, є актуальним напрямком досліджень, здатним поєднати якість рендерингу 3DGS із геометричною усвідомленістю, забезпеченою явним поверхневим представленням та генеративними можливостями дифузійних моделей.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Еволюція методів представлення тривимірних сцен

Задача синтезу нових точок зору (novel view synthesis, NVS) передбачає відновлення та рендеринг тривимірної сцени з довільної точки огляду за набором зображень, отриманих з відомих позицій камер. Класичні підходи до вирішення цієї задачі спираються на явні геометричні представлення – полігональні сітки, воксельні сітки або хмари точок, – однак вони вимагають складних та схильних до помилок конвеєрів отримання геометрії і мають обмежену здатність до фотореалістичного рендерингу складних оптичних явищ (тонкі структури, прозорість, дзеркальні відбиття).

Нейронні радіаційні поля запропонували принципово новий підхід: неявне представлення сцени у вигляді нейронної мережі F_Θ, що відображає п’ятивимірний вектор:

$$F_\Theta: (x,\, \theta,\, \phi) \;\mapsto\; (c,\, \sigma), \tag{1.1}$$

де $x \in \mathbb{R}^3$ – положення спостерігача;

(θ, ϕ) – двовимірний напрям спостереження на колір c і об’ємну густину σ

Синтез зображення здійснюється диференційованим трасуванням об’ємних променів. Колір пікселя $C(r)$ вздовж променя $r(t) = o + td$, що простягається від відстані t_n до t_f, обчислюється як:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T\,(t)\,\sigma\big(r(t)\big)\,c(r(t), d)\;\mathrm{d}t, \tag{1.2}$$

де $T(t) = \exp\left(-\int_{t_n}^{t} \sigma\left(r(s)\right) \mathrm{d}s\right)$ – акумульована прозорість вздовж променя.

NeRF довів здатність нейронних мереж до представлення надзвичайно деталізованих тривимірних сцен і вплинув на великий клас похідних методів. Однак його фундаментальні обмеження – час рендерингу від кількох секунд до хвилин на одне зображення, необхідність окремої оптимізації для кожної сцени, а також складність видобування явної геометрії – стримували практичне застосування. Наступні роботи (Instant NGP[11], Mip-NeRF 360[12], TensoRF[13]) суттєво прискорили навчання та рендеринг, але не подолали принципового обмеження неявних представлень: відсутності ефективного та якісного способу отримання явної поверхневої геометрії.

1.2 Метод 3D Gaussian Splatting

Метод 3D Gaussian Splatting (3DGS) є явним представленням сцени у вигляді набору анізотропних тривимірних гаусіанів, що оптимізуються градієнтним спуском за фотометричною функцією втрат на множині каліброваних видів. На відміну імпліцитних нейронних представлень, 3DGS забезпечує у реальному часі завдяки диференційовній тайл-базованій та уникає обчислювально витратного об'ємного семплування променів. Нижче послідовно розглянуто математичне формулювання примітиву процедуру растеризації та оптимізаційний цикл, що включає адаптивне керування густиною (densification та pruning).

1.2.1 Представлення сцени гаусовими примітивами

Метод 3D Gaussian Splatting (3DGS), запропонований Kerbl et al., подолав обмеження неявних методів, перейшовши до явного представлення сцени у вигляді набору N тривимірних гаусових примітивів $\mathcal{G} = \{G_i\}_{i=1}^{N}$.

Кожен примітив G_i характеризується чотирма групами параметрів:

– позиція $\mu_i \in, \mathbb{R} - 3.$ – центр гаусового примітиву у тривимірному просторі;

– форма і орієнтація – задаються матрицею коваріації $\Sigma_i \in \mathbb{R}^{3\times3}$;

– непрозорість $\alpha_i \in [0,1]$;

– колір – представлений коефіцієнтами сферичних гармонік для моделювання залежності від напрямку спостереження.

Просторовий розподіл кожного гаусового примітиву задається функцією:

$$G_i(x) = \exp\left(-\frac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\right). \quad (1.3)$$

Для коректної оптимізації матриця коваріації параметризується через декомпозицію:

$$\Sigma_i = R_i S_i S_i^\top R_i^\top, \quad (1.4)$$

де $R_i \in SO(3)$ – матриця повороту, задана кватерніоном $q_i \in \mathbb{R}^4$;

$S_i = \mathrm{diag}(s_{i,x}, s_{i,y}, s_{i,z})$ – діагональна матриця масштабів.

Така факторизація гарантує симетричність та невід'ємну визначеність Σ_i за будь-яких значень оптимізованих параметрів, що забезпечує стабільний градієнтний спуск.

1.2.2 Колір та сферичні гармоніки

Колір кожного гаусового примітиву залежить від напрямку спостереження і задається через розкладення за дійсними сферичними гармоніками (Spherical Harmonics, SH):

$$c_i(d) = \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} k_{i,\ell m}\ Y_\ell^m(d), \quad (1.5)$$

де $Y - l - m$ – сферична гармоніка порядку ℓ і ступеня m;

$k - i, lm$ – відповідний коефіцієнт;

d – нормований напрям спостереження.

У базовій реалізації 3DGS використовуються гармоніки до 3-го порядку включно (16 коефіцієнтів на колірний канал), що достатньо для відтворення ефектів дзеркального відбиття та залежних від напрямку змін освітлення.

1.2.3 Проектування та растеризація

Для синтезу зображення тривимірні гаусіани проектуються на площину зображення. Двовимірна коваріаційна матриця проекції Σ_i' обчислюється через апроксимацію Якобі:

$$\Sigma_i' = J_i W \Sigma_i W^\top J_i^\top, \quad (1.6)$$

де W – матриця повороту у просторі камери;

J_i – матриця Якобі афінного наближення проективного перетворення у точці μ_i.

Рендеринг здійснюється шляхом tile-based растеризації: зображення розбивається на плитки 16×16 пікселів, гаусіани, що перетинають кожну плитку, сортуються за глибиною та компонуються методом alpha compositing:

$$C(p) = \sum_{i \in \mathcal{N}(p)} c_i\, \hat{\alpha}_i \prod_{j=1}^{i-1} (1 - \hat{\alpha}_j), \quad (1.7)$$

де $\hat{\alpha}_i = \alpha_i \cdot G_i'(p)$ – ефективна непрозорість, що враховує значення проекції двовимірного гаусіана в пікселі p;

$N(p)$ – відсортований за глибиною набір примітивів, що впливають на піксель p.

1.2.4 Оптимізація та адаптивний контроль щільності

Ініціалізація 3DGS здійснюється за розрідженою хмарою точок, отриманою методом SfM (зазвичай – COLMAP [14]). Оптимізація параметрів $\{\mu_i, q_i, s_i, \alpha_i, k_i\}$ здійснюється стохастичним градієнтним спуском (Adam) шляхом мінімізації комбінованої функції втрат:

$$\mathcal{L} = (1 - \lambda)\,\mathcal{L}_1 + \lambda\,\mathcal{L}_{\text{D-SSIM}}, \tag{1.8}$$

де $\mathcal{L}_1$ – абсолютна фотометрична помилка;

$\mathcal{L}_{\text{D-SSIM}}$ – диференційований структурний індекс подібності (D-SSIM), $\lambda = 0.2$ за замовчуванням.

Ключовою особливістю 3DGS є адаптивний механізм контролю щільності (adaptive density control, ADC), що запускається кожні 100 ітерацій:

– клонування: примітиви з великим градієнтом позиції та малим масштабом дублюються, щоб покрити недостатньо відтворені дрібні деталі;

– розщеплення: великі примітиви з великим градієнтом розбиваються на два менших для кращого покриття деталізованих ділянок;

– відсікання: примітиви з непрозорістю нижче порогу ϵ_α видаляються для зменшення надмірності.

ADC дозволяє динамічно адаптувати кількість та розміщення гаусіанів упродовж навчання – від початкових кількох тисяч до кількох мільйонів примітивів у добре навченій моделі.

1.2.5 Переваги та обмеження 3DGS

Ключові практичні переваги 3DGS такі:

– рендеринг зі швидкістю понад 100 кадрів на секунду завдяки tile-based растеризації на GPU;

– якість реконструкції, порівнянна з NeRF, при значно меншому часі рендерингу;

– можливість видобування явної геометрії через постобробку.

Разом з тим, метод має суттєві обмеження. По-перше, гаусові примітиви оптимізуються незалежно, без урахування взаємозв'язків між сусідніми примітивами та без урахування глобальної геометрії сцени – це призводить до неузгоджених орієнтацій гаусіанів на межах поверхонь і до значної надмірності представлення. По-друге, якість видобутої сітки суттєво поступається спеціалізованим методам реконструкції поверхонь: примітиви розтягуються вздовж поверхонь або «плавають» у повітрі без зв'язку з реальною геометрією. По-третє, ефективність методу критично залежить від якості початкової SfM-хмари точок, що становить проблему для синтетичних сцен та сцен з недостатньою текстурою. Усунення цих обмежень є мотивацією для розроблення методів, описаних у наступних підрозділах.

1.3 Методи диференційованої екстракції поверхонь

Методи диференційованої екстракції поверхонь утворюють окремий напрямок, що дозволяє включати явну геометричну поверхню у градієнтну оптимізацію як проміжне представлення між неявним полем (SDF occupancy) та фінальним трикутним мешем. Класичний алгоритм Marching Cubes є кусково-сталим відносно вхідного поля та не градієнти через топологічні переходи, що унеможливлює його використання у диференційованих конвеєрах. У відповідь на це було запропоновано ряд

диференційованих альтернатив - тетраедральних сіток (DMTet, FlexiCubes) до м'яких релаксацій Cubes, які забезпечують узгоджені градієнти за параметрами поля вершинами поверхні. Нижче розглянуто базове поняття поля відстаней та ізоповерхні як спільну основу цих методів, а потім конкретні диференційовані схеми екстракції.

1.3.1 Поля знакових відстаней та ізоповерхні

Поле знакових відстаней (Signed Distance Function, SDF) – фундаментальне неявне представлення поверхні, де значення $s(x) \in R$ у кожній точці простору задає знакову відстань до найближчої точки поверхні: $s(x) > 0$ всередині об'єкта, $s(x) < 0$ – зовні, $s(x) = 0$ – на поверхні.

Ізоповерхня $S = x \mid s(x) = 0$ є точним визначенням поверхні у неявному вигляді. Класичний алгоритм Marching Cubes [15] витягує явну трикутну сітку з дискретного SDF на регулярній воксельній сітці шляхом покрокового перетину кожного куба з нульовою ізоповерхнею та генерації відповідних трикутників.

Однак стандартний Marching Cubes є недиференційованим: позиції вершин отримуються через лінійну інтерполяцію, і зворотне поширення градієнта крізь цю операцію нетривіальне. Тому наступний метод, який буде розглядатись є значно кращою версією для застосування.

1.3.2 Deep Marching Tetrahedra

DMTet (Deep Marching Tetrahedra) [10] запропонував перший повністю диференційований фреймворк для спільної оптимізації геометрії та топології поверхні.

Метод вводить деформовану тетраедральну сітку $\mathcal{T}$, що покриває тривимірний простір – на відміну від регулярної воксельної сітки, вузли

тетраедрів можуть переміщатись для кращого відтворення деталей поверхні. Кожному вузлу $\boldsymbol{v_i}$ відповідає значення SDF $s_i \in \mathbb{R}$. Диференційований шар marching tetrahedra (MT) генерує явну трикутну сітку, перетинаючи ребра тетраедрів нульовим рівнем SDF.

Для ребра між вузлами v_i та v_j (з різними знаками s_i, s_j) позиція вершини на поверхні визначається лінійною інтерполяцією:

$$v_{ij} = \frac{s_j v_i - s_i v_j}{s_j - s_i}. \tag{1.9}$$

Цей вираз є диференційованим за s_i, s_j, v_i, v_j, що дозволяє зворотне поширення градієнтів від функції втрат (наприклад, рендерингової або геометричної) крізь шар MT до параметрів мережі.

Топологія сітки – тобто зв'язність трикутників – визначається знаками SDF, тоді як геометрія (позиції вершин) контролюється значеннями SDF через формулу (1.9). Конкретніше, знак значення SDF у вузлі тетраедра вказує, чи знаходиться вузол всередині поверхні (від'ємне значення) або зовні (додатне). Ребро тетраедра перетинає поверхню тоді і тільки тоді, коли його вузли мають різні знаки; у такому разі вершина трикутної сітки розміщується на цьому ребрі відповідно до формули (1.9). Таким чином, будь-яка зміна знаку SDF у вузлі призводить до топологічної зміни – появи або зникнення ребра сітки, – тоді як зміна абсолютного значення SDF зсуває позицію вершини вздовж ребра тетраедра, не змінюючи топологію. Спільна оптимізація значень SDF та позицій вузлів тетраедральної сітки дозволяє одночасно змінювати як геометрію, так і топологію поверхні в процесі навчання. Завдяки цьому метод здатний моделювати поверхні довільного роду – тобто з будь-якою кількістю ручок або порожнин – без необхідності заздалегідь фіксувати топологічну структуру, що принципово відрізняє DMTet від методів, що оперують фіксованою сіткою.

1.3.3 Gaussian Opacity Fields

Gaussian Opacity Fields (GOF) [16] адаптував ідею диференційованого видобування поверхонь до екосистеми Gaussian Splatting, позбавившись від необхідності явного задання SDF.

Метод вводить поле гаусової непрозорості – скалярну функцію тривимірного простору, що обчислюється як зважена сума внесків гаусових примітивів:

$$\mathcal{O}(x) = \sum_{i=1}^{N} \alpha_i \; G_i(x), \tag{1.10}$$

де $G_i(x)$ – тривимірна гаусова функція.

Поверхня сцени ідентифікується як рівнева множина $O(x) = \tau$ для деякого порогу $\tau > 0$.

Для видобування явної трикутної сітки GOF застосовує Marching Tetrahedra на тетраедральній сітці, адаптивно роздрібнюваній навколо поверхні. Адаптивність дозволяє отримувати компактні сітки з вищою роздільною здатністю там, де поверхня є детальною, та нижчою – у рівних ділянках, що значно зменшує кількість трикутників порівняно з рівномірними підходами. GOF також розроблений для необмежених сцен (outdoor scenes), де традиційні методи втрачають точність через фіксований просторовий обсяг.

1.3.4 MILo

MILo [17] є найбільш інтегрованим представником сімейства DMTet: замість того щоб розглядати видобування сітки як постобробку, MILo включає цей процес безпосередньо в оптимізаційний цикл кожної ітерації навчання.

Ключова ідея MILo – двонаправлений потік градієнтів між об’ємним (гаусовим) та поверхневим (сітковим) представленнями. На кожній ітерації з поточних параметрів гаусіанів диференційовано витягується сітка (включно з координатами вершин та зв’язністю), далі сітка рендериться та отримує градієнт від рендерингової функції втрат, після чого цей градієнт розповсюджується назад крізь шар MT до параметрів гаусіанів. Таке замикання оптимізаційного циклу через явну поверхню змушує гаусіани розміщуватись у відповідності з геометрією поверхні, а не лише мінімізувати фотометричну помилку.

Результатом є взаємоузгоджена пара «гаусіани + сітка», де сітка має значно меншу кількість вершин при вищій якості порівняно з попередніми підходами постобробки, а гаусіани відтворюють поверхню значно точніше, ніж у стандартному 3DGS.

1.3.5 TeT-Splatting

TeT-Splatting [18] пропонує альтернативну стратегію, що зливає структуру тетраедральної сітки DMTet з механізмом tile-based растеризації 3DGS на рівні примітивів.

Замість гаусових кульок, TeT-Splatting використовує тетраедри як базові примітиви для сплаттінгу. Кожен тетраедр параметризується своїми чотирма вершинами та асоційованими атрибутами – непрозорістю та кольором, – і проектується на площину зображення аналогічно 3DGS. Растеризація тетраедрів забезпечує природне покриття тривимірного простору без прогалин, що спрощує екстракцію сітки: оскільки примітиви вже утворюють тетраедральну сітку, застосування MT є тривіальним.

Ця архітектура досягає трьох цілей одночасно: швидку збіжність оптимізації (завдяки структурованій ініціалізації), точну екстракцію сітки через Marching Tetrahedra (завдяки тетраедральній структурі) та рендеринг у реальному часі (завдяки tile-based растеризації).

1.3.6 Загальне обмеження сімейства DMTet

Тетраедралізація Делоне, що застосовується для побудови та адаптації тетраедральної сітки, може давати збої при тонких (малоякісних у сенсі співвідношення сторін) або майже планарних структурах. Розроблення гаусово-специфічних стратегій тетраедралізації для подібних топологій залишається відкритим напрямком досліджень.

1.4 Дифузійні генеративні моделі

Дифузійні генеративні моделі утворюють окремий клас імовірнісних моделей, які навчаються відтворювати складні розподіли даних шляхом послідовного обернення процесу поступового зашумлення. На відміну від варіаційних автокодерів та генеративно-змагальних мереж, дифузійні моделі демонструють стабільну збіжність під час навчання, високу якість згенерованих зразків та природну підтримку умовної генерації.

1.4.1 Деноізуючі дифузійні ймовірнісні моделі

Деноізуючі дифузійні імовірнісні моделі (Denoising Diffusion Probabilistic Models, DDPM) є класом генеративних моделей, що визначають двосторонній марківський процес: прямий процес зашумлення та зворотній процес деноізування.

Прямий (дифузійний) процес q поступово додає гаусовий шум до вхідних даних x_0 протягом T кроків:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right), \tag{1.11}$$

де $\beta_t \in (0,1)$ − розклад шуму (noise schedule).

Завдяки властивостям гаусових розподілів, x_t можна виразити напряму через x_0:

$$q(x_t \mid x_0) = \mathcal{N}\big(x_t;\, \sqrt{\bar{\alpha}_t}\, x_0,\, (1-\bar{\alpha}_t)I\big), \tag{1.12}$$

де $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$.

Зворотній процес p_θ навчається відновлювати дані з шуму:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1};\, \mu_\theta(x_t, t),\, \Sigma_\theta(x_t, t)). \tag{1.13}$$

Мережа ϵ_θ навчається передбачати шум, доданий до x_0, шляхом мінімізації функції:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{t, x_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\big(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon,\, t\big)\right\|^2\right], \tag{1.14}$$

де $\epsilon \sim N(0, I)$.

1.4.2 Латентні дифузійні моделі

Латентні дифузійні моделі (Latent Diffusion Models, LDM) (Rombach et al. 2022) підвищили ефективність DDPM, перенісши дифузійний процес з простору пікселів у стислий латентний простір автокодера. Автокодер (Variational Autoencoder, VAE) з енкодером $\mathcal{E}$ та декодером $\mathcal{D}$ отримує для зображення x латентне представлення $z = E(x)$, і дифузійна модель ϵ_θ оперує в просторі z:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathcal{E}(x), t, \epsilon}\big[\|\epsilon - \epsilon_\theta(z_t, t)\|^2\big]. \tag{1.15}$$

Завдяки роботі у стисненому просторі, LDM значно знижує обчислювальну складність – в f^2 разів, де f– коефіцієнт зменшення просторового розширення, – зберігаючи при цьому якість генерації на рівні піксельних дифузійних моделей.

Умовна генерація за текстом або зображенням реалізується через механізм крос-уваги (cross-attention), де умовна інформація $\boldsymbol{c}$ (наприклад, CLIP-ембединг тексту) інжектується у кожен шар денойзерної U-Net:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d}}\right)V, \tag{1.16}$$

де $Q = W_Q\varphi(z_t)$;

$K = W_K\tau_\theta(c)$;

$V = W_V\tau_\theta(c)$;

φ, τ, θ – проекції з'ясованого прихованого стану та умовного сигналу відповідно.

Стабільна дифузія (Stable Diffusion) – найпоширеніша реалізація LDM – стала базовою архітектурою для більшості сучасних дифузійних підходів до 3D-генерації.

1.4.3 GaussianDreamer

GaussianDreamer усуває проблему ініціалізації 3DGS для генеративних задач шляхом використання 3D-дифузійних моделей для отримання семантично значущих початкових хмар точок.

На першому кроці тривимірна дифузійна модель (наприклад, Shap-E або Point-E) за текстовим описом генерує грубу 3D-форму у вигляді хмари точок або імпліцитного представлення – це слугує геометрично послідовним початковим наближенням для наступної оптимізації. На другому кроці 3DGS ініціалізується з отриманих хмар точок і далі

уточнюється за допомогою 2D-дифузійної моделі через SDS-градієнти, що збагачують деталі текстури і дрібну геометрію.

Поєднання 3D-апріорної моделі та 2D-деталізації дозволяє GaussianDreamer генерувати 3D-інстанс або 3D-аватар протягом 15 хвилин на одному GPU при вищій мультиракурсній узгодженості порівняно з DreamGaussian.

1.4.4 DiffSplat

Для подолання дефіциту 3D-навчальних даних DiffSplat вводить легку модель реконструкції, що за кількома вхідними зображеннями передбачає мультиракурсні гаусові сітки (multiview Gaussian grids) – двовимірні зображення, де кожен піксель кодує атрибути гаусіана (позицію, масштаб, колір тощо) у відповідній просторовій позиції. Ці мультиракурсні сітки використовуються для масштабованого автоматичного накопичення навчального датасету.

Дифузійна модель навчається на цих сітках, використовуючи стандартну LDM-архітектуру з модифікованим каналом вводу. Введена 3D-рендерингова функція втрат сприяє 3D-узгодженості між генерованими гаусіанами для довільних точок зору. DiffSplat досягає якості генерації, порівнянної зі спеціалізованими 3D-методами, зберігаючи масштабованість і гнучкість великих 2D-дифузійних моделей.

1.4.5 DiffGS

DiffGS підходить до генерації гаусіанів через призму функціонального аналізу, вводячи поняття Gaussian Splatting Functions (GSF).

Метод розкладає оптимізоване 3DGS-представлення на три безперервні функції:

– $f_{\text{prob}}: \mathbb{R}^3 \to \mathbb{R}$ – функція гаусової ймовірності, що відображає де у просторі розташовані гаусіани;

– $f_{\text{color}}: \mathbb{R}^3 \to \mathbb{R}^3$ – функція кольору;

– $f_{\text{transform}}: \mathbb{R}^3 \to \mathbb{R}^7$ – функція геометричного перетворення (масштаб та поворот).

На основі цих функцій навчається гаусовий VAE, що стискає їх у компактний латентний простір. Умовна латентна дифузійна модель генерує нові конфігурації GSF з латентного простору. Така декомпозиція дозволяє DiffGS підтримувати широкий клас режимів генерації: безумовну генерацію, генерацію за текстовим описом, за зображенням-умовою, а також point-to-Gaussian – відновлення повної гаусової сцени з розрідженої хмари точок.

1.5 Гібридні підходи

Гібридні підходи поєднують переваги явних представлень на основі 3D-гаусіан з властивостями неявних або сітчастих геометричних структур, прагнучи подолати обмеження кожного з підходів окремо.

Чисті 3DGS-методи забезпечують швидкий рендеринг та високу фотореалістичність, проте погано відновлюють точну поверхневу геометрію та схильні до утворення «плаваючих» гаусіан у напівпрозорих ділянках. Натомість методи, що оперують мешами або SDF, дають коректну топологію поверхні, але поступаються у швидкості та якості реконструкції тонких деталей.

1.5.1 SuGaR

SuGaR (Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering) [20] є першим методом, що явно примусово вирівнює гаусіани з поверхнею сцени.

Метод вводить регуляризаційний доданок, що заохочує кожен гаусіан набувати плоскої, «тонкої» форми (тобто один із масштабів $s_{i,z} \ll s_{i,x}, s_{i,y}$) та наближатись до поверхні:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N}\left(\min_{k} s_{i,k}\right)^2 + (s(\mu_i))^2. \quad (1.17)$$

Після оптимізації з цим регуляризатором гаусіани виявляються щільно розташованими вздовж поверхні і слугують «обкладинкою» для методу реконструкції Пуассона, що відновлює явну трикутну сітку з орієнтованої хмари точок, яка утворена їхніми центрами та нормалями до самих гаусіанів. Для побудови такої хмари автори застосовують стратегію семплування рівневих множин~(level-set sampling): уздовж випадково обраних променів з тренувальних камер відшукуються точки, у яких сумарна щільність гаусового поля досягає заданого порогу, після чого як нормаль у такій точці береться напрямок градієнта щільності. Це дозволяє відтворити поверхневу геометрію без явного маркування, які саме гаусіани належать до поверхні. Реконструкція Пуассона, у свою чергу, добре масштабується на десятки мільйонів орієнтованих точок та стійка до локального шуму, що робить її природним вибором у цьому контексті.

Опційно гаусіани «прив'язуються» до граней отриманої сітки – кожному трикутнику відповідає фіксована кількість гаусіанів, закріплених у барицентричних координатах на його площині. Положення таких гаусіанів параметризується через ваги вершин трикутника, а їхня орієнтація узгоджується з нормаллю грані, тоді як масштаб у нормальному напрямку обмежується, аби гаусіани залишалися тонкими «лусочками» на поверхні. Завдяки цьому редагування або деформація сітки автоматично переносяться на гаусове представлення, що відкриває можливості для подальшого анімування, ретопології чи композиції сцен.

1.5.2 MeshSplats

MeshSplats [21] пропонує конвертувати результат Gaussian Splatting у формат, безпосередньо сумісний з методами трасування променів, без повторного навчання.

Після завершення стандартної оптимізації 3DGS метод перетворює «сплюснуті» (плоскі) гаусові примітиви на трикутні грані сітки – оскільки добре оптимізовані гаусіани на поверхні природно набувають плоскої форми, цей перехід є апроксимативно точним. Отримана сітка підтримує трасування променів, що відкриває можливості, недоступні в гаусовому растеризаторі: фізично коректні тіні, дзеркальні відбиття, глобальне освітлення та повна сумісність зі стандартними рушіями (Unity, Unreal Engine, Blender Cycles).

1.5.3 GaMeS

GaMeS [22] пропонує принципово відмінну параметризацію гаусових примітивів: замість незалежних позицій, кожен гаусіан прив’язується до вершин трикутної грані сіткового каркаса.

Позиція та орієнтація гаусіана визначаються через вершини відповідного трикутника $\{p_1, p_2, p_3\}$: центр $\mu_i = (p_1 + p_2 + p_3)/3$, а матриця повороту задається через нормаль до грані та тангентний вектор ребра. Відтак маніпуляція гаусіанами (деформація, анімація, редагування) здійснюється через управління вершинами сітки – стандартна і добре вивчена операція.

GaMeS відкриває можливість природної анімації 3DGS-сцен через стандартні сіткові анімаційні конвеєри (скелетна анімація, blend shapes), що є значним кроком до застосування Gaussian Splatting у реальному часі в ігровій індустрії.

1.5.4 DeMapGS

DeMapGS [23] пропонує найбільш інтегрований серед гібридних підходів фреймворк спільної оптимізації деформованих поверхонь та прив’язаних до них двовимірних гаусових сплатів.

Структурованою деформованою поверхнею слугує параметрична сітка, де кожна вершина може зміщуватись у ході навчання. До поверхні «прикріплюються» 2D-гаусіани – примітиви, що лежать безпосередньо на площині сітки – завдяки чому поверхня і гаусіани оптимізуються узгоджено. Ключова інновація – стратегія дифузії градієнтів (gradient diffusion): навчальний сигнал (фотометрична помилка) не просто локально застосовується до кожного гаусіана, а дифундує вздовж поверхні до сусідніх вершин сітки через зважену дифузію:

$$\nabla_{v_k}^{\text{diffused}} = \sum_{j \in \mathcal{N}(k)} w_{kj} \ \nabla_{v_j}^{\text{local}}, \tag{1.18}$$

де w, k, j – ваги дифузії (наприклад, на основі кутарієвих ваг котангенсів);

$N(k) -$ сусіди вершини k на сітці.

Завдяки глобальному розповсюдженню градієнтів поверхня узгоджено деформується навіть у малотекстурованих ділянках, що недостатньо покриті прямими фотометричними спостереженнями. DeMapGS досягає якості реконструкції сітки на рівні кращих сучасних методів, одночасно підтримуючи редагування сцени та перенесення об’єктів через спільну параметричну поверхню.

1.6 Структурно-усвідомлені підходи: SAGS

Метод SAGS (Structure-Aware Gaussian Splatting) є найближчим до запропонованого у цій роботі підходом за мотивацією – явного врахування геометричної структури сцени при оптимізації гаусових примітивів.

SAGS будує граф сусідства $\mathcal{G}_{\text{struct}} = (\mathcal{V}, \mathcal{E})$ над набором гаусіанів, де вузли $\mathcal{V} = \{G_i\}$ відповідають примітивам, а ребра $\mathcal{E}$ з'єднують k найближчих сусідів кожного примітиву в евклідовому просторі. Граф кодується графовою нейронною мережею (Graph Neural Network, GNN), яка агрегує інформацію від сусідів та оновлює атрибути гаусіанів через операцію передачі повідомлень:

$$h_i^{(\ell+1)} = \phi\left(h_i^{(\ell)}, \oplus_{j\in\mathcal{N}(i)}\psi\left(h_i^{(\ell)}, h_j^{(\ell)}, e_{ij}\right)\right), \tag{1.19}$$

де $h, i, (l) -$ прихований стан вузла i на шарі ℓ, ϕ;

ψ – нейронні мережі оновлення та агрегації відповідно;

$\oplus$ – комутативна операція агрегації (сума або максимум).

Структурно-усвідомлена регуляризація через k-NN граф підвищує зв'язність представлення та зменшує артефакти на межах поверхонь. Однак підхід на основі k-NN має ряд суттєвих обмежень:

– граф k-NN є локальним і не відображає глобальну топологію поверхні сцени – зокрема, він не розрізняє просторово близькі примітиви на одній поверхні та на різних поверхнях;

– k-NN-граф не несе інформації про нормалі, кривизну або зв'язність поверхні, наявної в явних сіткових представленнях;

– якість структури k-NN суттєво залежить від рівномірності початкового розподілу гаусіанів, що є проблемою для синтетичних сцен, де SfM-ініціалізація є неприйнятною через відсутність реальних зображень.

1.7 Порівняльний аналіз розглянутих підходів

Для систематизації розглянутих методів і виявлення дослідницької ніші запропонованого підходу наведемо їх зведений порівняльний аналіз (таблиця 1.1).

Таблиця 1.1 – Порівняльний аналіз підходів до реконструкції та генерації 3D-сцен на основі Gaussian Splatting

Метод	Структ.	Дифуз.	Явна сітка	Реконстр.
3DGS	-	-	-	
GOF	-	-	пост.	
MILo	-	-	онлайн	
TeT-Splatting	тетр.	-		
DreamGaussian	-	2D	пост.	-
GaussianDreamer	-	2D+3D	-	-
DiffSplat	-	2D	-	-
DiffGS	-	3D	-	-
SuGaR	регул.	-	пост.	
GaMeS	сітка	-		
DeMapGS	сітка	-		
SAGS	k-NN	-	-	

Примітка. «Структ.» – тип структурної апріорної моделі (тетр. – тетраедральна сітка, регул. – регуляризація, сітка – явна поверхнева сітка, k-NN – граф найближчих сусідів); «Дифуз.» – тип дифузійної апріорної моделі (2D – 2D-дифузійна модель, 3D – 3D-дифузійна модель, 2D+3D – обидва); «Явна сітка» – наявність явної сітки у представленні або результаті («пост.» – лише як постобробка, «онлайн» – в ході навчання, – завжди); «Реконстр.» – підтримка задачі реконструкції (а не лише генерації).

З таблиці 1.1 видно, що запропонований метод займає унікальну нішу: жоден із розглянутих підходів одночасно не поєднує 3D-дифузійну апріорну модель із явною поверхневою сіткою та орієнтацією на задачу реконструкції.

Найближчими конкурентами є SAGS (структурний пріор + реконструкція, але без сітки та дифузійних пріорів) та DiffGS (3D-дифузійний пріор, але лише для генерації, без явної сітки та реконструкції).

1.8 Постановка задачі

Проведений аналіз засвідчує, що, незважаючи на значний прогрес у методах 3D Gaussian Splatting та суміжних напрямках, проблема ефективного структурування гаусових примітивів за допомогою геометрично значущої глобальної апріорної моделі залишається відкритою. Існуючі підходи вирішують лише окремі аспекти цієї проблеми, не формуючи цілісного механізму поєднання структурного пріору з гаусовим представленням сцени.

Методи сімейства DMTet (GOF, MILo, TeT-Splatting) зосереджені переважно на видобуванні сітки з уже оптимізованих гаусіанів і не надають структурних апріорних моделей для ініціалізації та регуляризації. Тетраедральна структура у цих методах виконує роль допоміжного засобу для пост-обробки гаусового поля, а не джерела геометричних обмежень на етапі оптимізації. Крім того, тетраедралізація Делоне, що лежить в основі цих підходів, чутлива до тонких та майже планарних топологій (листя, тонкі металеві конструкції, тканини тощо), що ускладнює їхнє застосування до сцен зі складною геометрією деталей.

Дифузійні підходи (DreamGaussian, GaussianDreamer, DiffSplat, DiffGS) демонструють потенціал дифузійних моделей як геометричних пріорів, однак орієнтовані переважно на задачу генерації об'єктів за текстовим або зображувальним описом, а не на реконструкцію реальних сцен із багатовидових даних. Геометричний пріор, що неявно міститься у вагах дифузійної моделі, у цих методах використовується через SDS-подібні функції втрат або як ініціалізація з шумних 3D-зразків, але не у формі явної структурованої поверхневої сітки, придатної для подальшого геометричного аналізу та регуляризації. Гібридні методи (SuGaR, MeshSplats, GaMeS, DeMapGS) пропонують різноманітні способи об'єднання сітки та гаусових представлень- від реконструкції Пуассона на основі рівневих множин гаусового поля до прив'язки гаусіанів у

барицентричних координатах граней та сумісної оптимізації позицій вершин і параметрів примітивів. Проте сітка у цих підходах отримується безпосередньо з даних поточної сцени, без участі зовнішньої апріорної моделі, тому її якість обмежена щільністю покриття видами та точністю SfM-ініціалізації.

Як наслідок, ці методи погано застосовні до синтетичних сцен (наприклад, NeRF Synthetic), де SfM-хмара відсутня або є виродженою. SAGS використовує структурний граф для регуляризації параметрів гаусіанів, проте k-NN-зв'язність на хмарі центрів є недостатньо інформативним і незагальним представленням геометрії: вона відображає лише локальну сусідність у поточному стані оптимізації, не несе інформації про поверхневу топологію (зв'язні компоненти, орієнтація нормалі, межі поверхні) і не зберігає стійкість при перестановках чи розщепленнях гаусіанів під час densification.

Отже, граф SAGS функціонує як локальний, а не глобальний геометричний пріор. Спільним недоліком перелічених напрямків є відсутність єдиного механізму, який одночасно:

– постачає глобальний геометричний пріор у вигляді явної поверхневої структури;

– отримується незалежно від конкретної сцени та переноситься на нові об'єкти;

– придатний як для реальних сцен із SfM-ініціалізацією, так і для синтетичних сцен без неї.

З огляду на це, задача подальшого дослідження формулюється наступним чином: розробити метод, у якому дифузійно-генерована структурована сітка слугує геометричною апріорною моделлю для ініціалізації та регуляризації Gaussian Splatting. Дифузійна модель навчається один раз на репрезентативному корпусі тривимірних об'єктів (Objaverse) у формі дифузії над сітковим представленням SDF, після чого для кожної нової сцени застосовується для генерації поверхневої

сітки на основі розрідженої хмари точок. Отримана сітка використовується двома шляхами: як основа для ініціалізації гібридного гаусового представлення (поверхневі гаусіани, закріплені у барицентричних координатах граней, у поєднанні з вільними гаусіанами для непокритих регіонів) та як джерело геометричних обмежень у вигляді функцій втрат на відстань від поверхні та узгодження нормалей. Очікується, що такий підхід дозволить:

– підвищити якість реконструкції тривимірних сцен (за метриками PSNR, SSIM, LPIPS на тестових видах) порівняно з існуючими структурно-усвідомленими підходами, зокрема SAGS;

– зберегти ефективність рендерингу 3DGS, не вводячи додаткових обчислювально-витратних операцій на етапі візуалізації;

– забезпечити придатність методу до синтетичних сцен без SfM-ініціалізації, де класичні гібридні підходи (SuGaR, MeshSplats) є непрацездатними;

– отримати структурно когерентне гаусове представлення, в якому переважна частина примітивів локалізована у тонкому шарі вздовж поверхні, що відкриває можливості для подальшого видобування високоякісної трикутної сітки та геометричного редагування сцени.

2 МЕТОДИ ТА ЗАСОБИ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Метод дифузійно-керованого Gaussian Splatting

У цьому розділі формалізується запропонований метод Diffusion-driven Gaussian Splatting, який інтегрує дифузійно-генеровану структуровану сітку як просторову апріорну модель у конвеєр оптимізації 3DGS. Метод оформлено у двостадійній структурі: офлайн-етап навчання дифузійної моделі знакових дистанційних функцій (SDF) на наборі тривимірних об'єктів, та етап реконструкції конкретної сцени, на якому хмара точок сцени конвертується у поверхневу сітку за допомогою натренованої дифузійної моделі й використовується для ініціалізації та регуляризації гібридної моделі гаусіанів.

Нехай задано набір зображень $\{I_v\}_{v=1}^{V}$, зроблених камерами із відомими параметрами – внутрішніми $\{K_v\}_{v=1}^{V}$ та зовнішніми $\{[R_v \mid t_v]\}_{v=1}^{V}$, – та розріджену вхідну хмару точок $\mathcal{P} = \{p_n\}_{n=1}^{P} \subset \mathbb{R}^3$ з кольорами $\{c_n\}_{n=1}^{P} \subset [0,1]^3$. Метою задачі реконструкції є побудова явного тривимірного представлення $\mathcal{G}^\star$ сцени, яке мінімізує середню фотометричну помилку рендерингу зображень із заданих точок зору:

$$\mathcal{G}^\star = \operatorname*{argmin}_{\mathcal{G}} \frac{1}{V} \sum_{v=1}^{V} \mathcal{L}_{\text{photo}}\left(R(\mathcal{G}, K_v, [R_v \mid t_v]), I_v\right), \tag{2.1}$$

де $R(\cdot)$ – диференційований оператор растеризації;

$\mathcal{L}_{\text{photo}}$ – комбінована фотометрична функція втрат.

Відмінністю запропонованої постановки від базової 3DGS є те, що представлення $\mathcal{G}$ декомпонується на дві групи гаусових примітивів – поверхневих та вільних – де поверхневі гаусіани параметризуються через явну сітку $\mathcal{M} = (V, F)$, згенеровану дифузійною моделлю за вхідною хмарою точок $\mathcal{P}$:

$$\mathcal{M} = \text{MeshFromPCD}\left(\mathcal{P};\, \epsilon_\theta,\, \mathcal{E}_\phi\right), \tag{2.2}$$

де ϵ_θ – параметри денойзерна 3D U-Net дифузійної моделі;

$\mathcal{E}_\phi$ – кодувальник хмари точок Point-E з адаптерним шаром.

Таким чином, формалізована задача полягає в одночасній оптимізації:

– параметрів вільних гаусіанів $\{\mu_i^{\text{f}}, q_i^{\text{f}}, s_i^{\text{f}}, \alpha_i^{\text{f}}, k_i^{\text{f}}\}$;

– барицентричних ваг $\{\alpha_j\}$ та коефіцієнтів сферичних гармонік $\{k_j^{\text{s}}\}$ поверхневих гаусіанів;

– вершин сітки V (для тонкого підлаштування геометрії);

– при обмеженні $\mu_j^{\text{s}} \in \text{conv}\left(v_{f_j^0}, v_{f_j^1}, v_{f_j^2}\right)$ для всіх поверхневих гаусіанів j.

2.1.1 Знакові дистанційні функції

Як представлення тривимірної геометрії об’єкта обрано знакову дистанційну функцію (SDF), задану на регулярній воксельній сітці $\Omega = \{1, \dots, R\}^3$ роздільної здатності $R = 32$. Скалярне поле $s: \Omega \to R$ у кожному вокселі задає знакову відстань до найближчої точки поверхні об’єкта; ізоповерхня $s = 0$ відповідає реконструйованій поверхні.

Вибір роздільної здатності узгоджений із обмеженнями обчислювальної складності тривимірної згорткової U-Net: повний об’єм 32^3 = 32 768 вокселів формує тензор форми (1, 1, 32, 32, 32), який ефективно обробляється сучасними GPU без потреби в розбитті на блоки. Експерименти з вищими роздільними здатностями (зокрема 64^3) показали нелінійне зростання часу одного дифузійного кроку без істотного приросту точності реконструкції грубих топологічних структур, що домінують у геометричній апріорній моделі, проте вимагають значного збільшення обчислювальних потужностей для обрахунку.

2.1.2 Архітектура знешумлювальної мережі

Знешумлювальну мережу $\epsilon_\theta: \mathbb{R}^{1\times 32^3} \times \mathbb{N} \times \mathbb{R}^{1\times d_c} \to \mathbb{R}^{1\times 32^3}$ реалізовано як тривимірну U-Net з механізмом крос-уваги для умовної генерації. Архітектура має три рівні деталізації з кількістю каналів (c_1, c_2, c_3) = (64,128,256) та містить:

– кодувальну вітку з трьох ResBlock3d та двох операцій даунсемплінгу зі страйдом 2;

– «вузьке горло» з ResBlock3d, нормалізацією за шарами та блоком крос-уваги, в якому ключі і значення формуються з умовного токена $c \in \mathbb{R}^{1\times 512}$, отриманого з Point-E енкодера;

– декодувальну вітку з симетричними операціями апсемплінгу та ResBlock3d з skip-з'єднаннями.

Залежність від часового кроку t ін'єктується у кожен резидуальний блок шляхом синусоїдально-косинусоїдального вкладення $\mathrm{TE}(t) \in \mathbb{R}^{256}$:

$$\mathrm{TE}(t) = [\sin(\omega_k t), \cos(\omega_k t)]_{k=0}^{d/2-1}, \tag{2.3}$$

де $\omega_k = 10000^{-2k/d}$.

2.1.3 Навчання дифузійної мережі

Тренування виконується на наборі попередньо обчислених SDF-сіток $\{x_0^{(m)}\}_{m=1}^{M}$, отриманих із об'єктів датасету Objaverse [24] шляхом обчислення точної знакової відстані на сітці 32^3 за допомогою алгоритму CLOMAP.

Розклад шуму β_t задано косинусним графіком [25]:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \tag{2.4}$$

де $f(t) = \cos^2\left(\frac{(t/T)+s}{1+s} \cdot \frac{\pi}{2}\right)$;

s = 0.008.

Це дає $\beta_t = 1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}$ з обмеженням $\beta_t \in [10^{-8}, 0{,}999]$. Цільовою функцією є очікувана норма похибки передбачення шуму – стандартна DDPM-параметризація:

$$\mathcal{L}_{\text{diff}}(\theta, \phi) = \mathbb{E}_{m,\, t,\, \epsilon} \left\| \boldsymbol{\epsilon} - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t}\, x_0^{(m)} + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon},\, t,\, \mathcal{E}_\phi(\mathcal{P}^{(m)}) \right) \right\|_2^2 \quad (2.5)$$

де $\mathcal{P}^{(m)}$ – хмара точок, відібрана з поверхні m-го об'єкта;

$\epsilon \sim N(0, I)$;

$t \sim Uniform 1, \ldots, T$.

Кодувальна мережа Point-E $\mathcal{E}_\phi$ використовує заморожений backbone base40M [19] та тренувальний адаптер $\phi: \mathbb{R}^{512} \to \mathbb{R}^{512}$. Підпис тензора – $\mathcal{E}_\phi(\mathcal{P}) \in \mathbb{R}^{1\times1\times512}$ – слугує умовою для крос-уваги у вузькому горлі денойзера.

2.1.4 Інференс: генерація сітки зі стиснутої хмари точок

На етапі інференсу для конкретної сцени із хмари точок $\mathcal{P}$ будується умовний токен $c = \mathcal{E}_\phi(\mathcal{P})$, після чого з ізотропного гаусового шуму $x_T \sim \mathcal{N}(0, I)$ методом DDIM [26] з $K = 50$ детермінованих кроків ($\eta = 0$) виконується зворотній процес денойзингу:

$$x_{t_{k+1}} = \sqrt{\bar{\alpha}_{t_{k+1}}}\, \hat{x}_0(x_{t_k}) + \sqrt{1 - \bar{\alpha}_{t_{k+1}}}\, \epsilon_\theta(x_{t_k}, t_k, c). \quad (2.6)$$

Отриману SDF-сітку x_0 підлягає алгоритму Marching Cubes з рівнем $l = 0$ та однаковим інтервалом $h = 2/(R-1)$, що породжує трикутну сітку

у нормалізованих координатах $[-1,1]^3$. Останній крок – денормалізація сітки у координати оригінальної сцени:

$$v_i = \sigma_{\mathcal{P}} \odot \tilde{v}_i + \mu_{\mathcal{P}}, \tag{2.7}$$

де $\mu_{\mathcal{P}}, \sigma_{\mathcal{P}} \in \mathbb{R}^3$ – середнє і стандартне відхилення позицій вхідної хмари точок;

$\odot -$ покомпонентний добуток.

2.2 Гібридне представлення гаусіанів

Гібридне представлення гаусіанів є центральним архітектурним рішенням запропонованого методу і покликане поєднати дві властивості, що в існуючих підходах зазвичай розглядаються як взаємовиключні: жорстку геометричну прив'язку примітивів до апріорної поверхні – з одного боку, та гнучкість вільного гаусового поля, здатного компенсувати неточності пріору і відтворювати тонкі або напівпрозорі деталі сцени – з іншого.

2.2.1 Розклад набору на поверхневі та вільні примітиви

Запропонована модель сцени являє собою об’єднання двох наборів гаусових примітивів:

$$\mathcal{G} = \mathcal{G}^{\mathrm{s}} \cup \mathcal{G}^{\mathrm{f}}, \tag{2.8}$$

де $\mathcal{G}^{\mathrm{s}} = \{G_j^{\mathrm{s}}\}_{j=1}^{F}$;

$\mathcal{G}^{\mathrm{f}} = \{G_i^{\mathrm{f}}\}_{i=1}^{N}$;

$F = |F|$ – кількість граней сітки (по одному поверхневому гаусіану на грань);

N – кількість вільних гаусіанів (динамічна, ініціалізується з $|P|$ і змінюється при адаптивному контролі щільності).

Поверхневі гаусіани відповідають за апроксимацію поверхневого вигляду сцени та забезпечують структурну зв'язність примітивів, тоді як вільні гаусіани покривають високочастотні деталі, неоднорідності та елементи поза реконструйованою поверхнею (волосся, тонкі провідники, прозорі елементи тощо).

2.2.2 Параметризація поверхневих гаусіанів

Кожному трикутнику $f_j = (\boldsymbol{v}_{f_j^0}, \boldsymbol{v}_{f_j^1}, \boldsymbol{v}_{f_j^2})$ поверхневої сітки зіставлено один гаусів примітив, позиція центра якого визначається через диференційовну барицентричну параметризацію:

$$\mu_j^{\mathrm{s}} = \mathrm{softmax}(\alpha_j) \cdot \begin{bmatrix} v_{f_j^0} \\ v_{f_j^1} \\ v_{f_j^2} \end{bmatrix}, \tag{2.9}$$

де $\alpha_j \in \mathbb{R}^3$;

$softmax$ гарантує невід'ємні барицентричні координати з сумою 1, тобто μ_j^{s} завжди залишається в опуклій оболонці грані, незалежно від значень параметра α_j.

На початку оптимізації $\alpha_j = (0,0,0)$, що відповідає центроїду грані $\mu_j^{\mathrm{s}} = (v_{f_j^0} + v_{f_j^1} + v_{f_j^2})/3$.

Орієнтація поверхневого гаусіана задається ортонормованою системою координат, побудованою з геометрії грані. Спочатку обчислюється нормаль до грані:

$$n_j = \frac{e_0 \times e_1}{\|e_0 \times e_1\| + \epsilon}, \tag{2.10}$$

де $e_0 = v_{f_j^1} - v_{f_j^0}$;

$e_1 = v_{f_j^2} - v_{f_j^0}$.

Далі – два дотичних вектори u_j, w_j через процес ортогоналізації Грама-Шмідта.

Матриця повороту складається стовпцями $R_j = [n_j \mid u_j \mid w_j]$ та переводиться у кватерніон q_j^{s} за стандартною формулою [27]. Масштаби поверхневого гаусіана обрано так, щоб гаусіан був тонким уздовж нормалі і покривав грань у дотичній площині:

$$s_j^{\mathrm{s}} = \left(\epsilon,\ 1/2 \parallel e_0 \parallel,\ 1/2\,|\langle e_1', w_j \rangle|\right), \tag{2.11}$$

де $e_1' = v_{f_j^2} - \mu_j^{\mathrm{s},0}$;

$\epsilon = 10^{-8} -$ мала постійна, що задає квазі-нульову товщину вздовж нормалі.

Така параметризація відповідає за духом 2D-гаусіану, прив'язаному до грані, аналогічно ідеям GaMeS, але без жорсткого закріплення центра у центроїді.

Параметри непрозорості α_j^{s} та коефіцієнти сферичних гармонік k_j^{s} кожного поверхневого гаусіана тренуються вільно нарівні з вільними гаусіанами. Однак, на відміну від вільних, поверхневі гаусіани не беруть участі в адаптивному контролі щільності: їхня кількість фіксована та дорівнює кількості граней сітки.

2.2.3 Параметризація вільних гаусіанів

Вільні гаусіани зберігають класичну параметризацію 3DGS:

$$G_i^{\mathrm{f}} = \left(\mu_i^{\mathrm{f}},\ q_i^{\mathrm{f}},\ s_i^{\mathrm{f}},\ \alpha_i^{\mathrm{f}},\ k_i^{\mathrm{f}}\right), \quad i = 1, \ldots, N. \tag{2.12}$$

Початкові позиції μ_i^{f} беруться з вхідної хмари точок $\mathcal{P}$, кольори ініціалізуються з $\mathcal{P}$ через проєкцію $\mathrm{rgb} \to \mathrm{SH}_0$. На відміну від стандартної ініціалізації 3DGS, у запропонованому методі початкові масштаби та орієнтації вільних гаусіанів вирівнюються до найближчої грані сітки:

$$s_i^{\mathrm{f},0} = \log\left(\mathrm{clamp}_{\min}\left(\| \mu_i^{\mathrm{f}} - m_{n(i)} \|, 10^{-4}\right)\right) \cdot 1_3, \tag{2.13}$$

$$q_i^{\mathrm{f},0} = \mathrm{quat}\left(e_z, n_{n(i)}\right), \tag{2.14}$$

де $n(i) = \mathrm{argmin}_j \| \mu_i^{\mathrm{f}} - m_j \|$ – індекс найближчого центроїда грані;

m_j – центроїд j-ої грані;

$e_z = (0,0,1,)^T$ – референтна вісь;

$quat(\cdot,\cdot) -$ кватерніон, що відображає e_z у $n_{n(i)}$ через дугу найкоротшої довжини.

Така ініціалізація задає для оптимізатора попередньо орієнтовану конфігурацію вільних гаусіанів, що скорочує перехідний період до структурно-узгодженого стану.

2.3 Функції втрат

Нижче проведено розбір основних функцій втрат.

2.3.1 Фотометрична компонента

Базова фотометрична функція втрат успадковується від 3DGS і є зваженою комбінацією L_1-помилки зображення та диференційованого структурного індексу подібності D-SSIM:

$$\mathcal{L}_{\mathrm{photo}}(\hat{I}, I) = (1 - \lambda_{\mathrm{dssim}}) \| \hat{I} - I \|_1 + \lambda_{\mathrm{dssim}} \left(1 - \mathrm{SSIM}(\hat{I}, I)\right), \tag{2.15}$$

де $\lambda_{\text{dssim}} = 0{,}2$.

2.3.2 Регуляризатор близькості до поверхні

Для забезпечення концентрації значущих вільних гаусіанів навколо реконструйованої поверхні введено регуляризатор близькості до сітки:

$$\mathcal{L}_{\text{surf}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \max\left(d(\mu_i^{\text{f}}, \mathcal{M}) - \delta,\, 0\right), \quad \mathcal{I} = \{i : \sigma(\alpha_i^{\text{f}}) > \tau_\alpha\}, \quad (2.16)$$

де $d(\mu, \mathcal{M}) = \min_j \| \mu - m_j \|$ – відстань до найближчого центроїда грані;

$\delta = \rho \cdot, e_{scene}$ – допустимий поріг віддалення (пропорційний відсотку $\rho = 0{,}01$ від екстенту сцени e_{scene});

$\tau_\alpha = 0{,}3$ – мінімальний поріг непрозорості для участі гауссіана у регуляризаторі (виключаються майже-прозорі примітиви, що не впливають на рендеринг).

Hinge-форма $max(\cdot\,,0)$ створює зону «м'якого» допуску – гаусіани ближче ніж δ до сітки не штрафуються, що дозволяє адаптивну поведінку для високочастотних деталей.

2.3.3 Регуляризатор вирівнювання нормалей

Другий регуляризатор задається з метою узгодження орієнтації вільних гаусіанів з нормалями найближчих граней сітки. Для кожного активного вільного гаусіана $i \in I$ визначаємо його «вісь сплюснутості» – одиничний вектор a_i уздовж найменшого з масштабів:

$$a_i = R_i e_{k_i^\star}, \quad k_i^\star = \arg\min_{k\in\{1,2,3\}} s_{i,k}, \quad (2.17)$$

де R_i – матриця повороту, отримана з кватерніона q_i^{f}, , e_k. – k-й стандартний базисний вектор.

Регуляризатор штрафує відхилення a_i від нормалі найближчої грані:

$$\mathcal{L}_{\mathrm{norm}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\left(1-(a_i\cdot n_{n(i)})^2\right). \tag{2.18}$$

Квадратична форма $(1-\cos^2)$ є інваріантною до напрямку нормалі, що уникає проблем зі знаком при реконструкції двосторонніх поверхонь.

2.3.4 Сумарна функція втрат

Повна цільова функція оптимізації має вигляд:

$$\mathcal{L} = \mathcal{L}_{\mathrm{photo}} + w(\tau)\left(\lambda_{\mathrm{surf}}\mathcal{L}_{\mathrm{surf}} + \lambda_{\mathrm{norm}}\mathcal{L}_{\mathrm{norm}}\right) + \\ +\lambda_{\mathrm{bal}}\,\mathrm{ReLU}\left(\frac{N}{F+N}-\tau_{\mathrm{bal}}\right) \tag{2.19}$$

де $w(\tau)-$ лінійно спадний коефіцієнт, що зменшує внесок сіткових регуляризаторів від 1 на початку щільнювальної фази до 0 по завершенню;

$\lambda_{\mathrm{surf}} = 0{,}01$;

$\lambda_{\mathrm{norm}} = 0{,}005$;

$\lambda_{\mathrm{bal}} = 0{,}01$;

$\tau_{\mathrm{bal}} = 0{,}7$.

Останній доданок – регуляризатор балансу між поверхневими та вільними гаусіанами – активується, коли частка вільних гаусіанів перевищує 70, і запобігає виродженню гібридної моделі у класичну 3DGS-схему шляхом обмеження надмірного розщеплення вільних гаусіанів.

2.3.5 Адаптивний контроль щільності за гібридних умов

Процедура адаптивного контролю щільності (ADC) запропонованого методу зберігає ключові ідеї базового 3DGS – клонування, розщеплення та відсікання, – однак застосовується виключно до підмножини вільних гаусіанів. Поверхневі гаусіани зберігають фіксований розмір F, що дорівнює кількості граней сітки, і не підлягають жодній з операцій ADC.

Формально, маска допустимості для клонування / розщеплення формується як:

$$m_i^{\text{clone}} = \mathbb{I}\left(\| \nabla \mu_i^{\text{f}} \|_2 \geq \tau_g \ \wedge \ \max_k s_{i,k}^{\text{f}} \leq \rho\, e_{\text{scene}} \ \wedge \ i \in \mathcal{I}^{\text{f}}\right), \qquad (2.20)$$

де $\mathcal{I}^{\text{f}}$ – індекси вільних гаусіанів;

$\tau_g = 2 \times ,10 - -4. -$ поріг градієнта позиції;

$\rho = 0{,}01 -$ максимально допустимий відносний масштаб.

Видалення (prune) застосовується до вільних гаусіанів з непрозорістю нижче порогу $\sigma(\alpha_i^{\text{f}}) < 0{,}005$ та до примітивів, чий екранний радіус перевищує 20 пікселів або просторовий розмір – 10 екстенту сцени.

Цей підхід забезпечує два важливих ефекти:

– стабільність структурного представлення – поверхневий «каркас» не змінюється протягом усієї оптимізації;

– обчислювальну ефективність – кількість гаусіанів зростає лише за рахунок вільних, ніж у класичному 3DGS.

2.3.6 Загальна схема методу

Зведена схема запропонованого методу подана на рисунку 2.1.

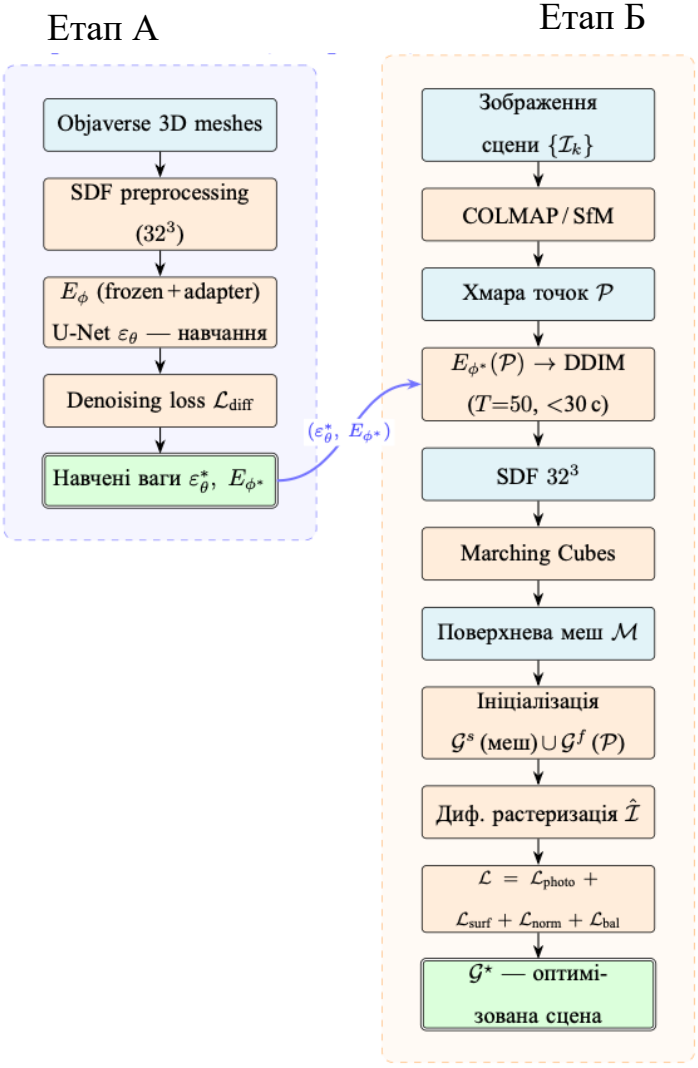


Рисунок 2.1 – Загальна двостадійна схема методу Diffusion-driven Gaussian Splatting: етап А навчає дифузійну апріорну модель на наборі тривимірних об’єктів; етап В використовує навчену апріорну модель для реконструкції конкретної сцени

Перевагою запропонованої двостадійної структури є те, що етап А є сценонезалежним – однократно навчена дифузійна апріорна модель застосовна до будь-якої нової сцени без перенавчання, – тоді як етап В зберігає швидкість традиційної реконструкції 3DGS, оскільки виклик $\mathcal{E}_{\phi^\star}$ та DDIM-семплінгу виконується одноразово протягом < 30 секунд на типовій сцені.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ

3.1 Загальна архітектура та структура проєкту

Кодова база організована у три основних модулі. Поділ на модулі відображає логіку методу:

– gaussian_splatting/ – основний модуль реконструкції, що містить модель гаусіанів, диференційований растеризатор та тренувальний цикл оптимізації сцени;

– diffusion/ – модуль дифузійної апріорної моделі з реалізацією DDPM/DDIM, тривимірної денойзерної U-Net, Point-E кодувальника, а також скрипти препроцесингу Objaverse та тренування апріорної моделі;

– utils/ – спільні утиліти для I/O форматів COLMAP та NeRF Synthetic, математичні операції, активаційні функції, обчислення метрик, інтеграція з MLflow.

Точкою входу для реконструкції сцени є скрипт `main.py`, що приймає аргументи командного рядка та налаштовує конфігурації `ModelConfig`, `OptimizationConfig` і `PipelineConfig`, після чого передає їх у функцію `train()` модуля `gaussian_splatting`.train. Для дифузійної апріорної моделі окремою точкою входу є модуль `diffusion.train_diffusion`, що тренує денойзер на попередньо обчислених SDF-сітках.

3.2 Модуль gaussian_splatting

Розглянемо детальніше структуру кожного модуля.

3.2.1 Клас GuassianModel

Клас `GaussianModel` інкапсулює усі тренувальні параметри сцени у вигляді nn.Parameter-тензорів:

– `_xyz, _scaling, _rotation, _opacity` – параметри вільних гаусіанів;

– `_features_dc, _features_rest` – коефіцієнти сферичних гармонік, розділені на DC-компоненту та вищі порядки для ефективнішої роботи з sparse-Adam оптимізатором;

– `vertices, _alpha, faces` – параметри поверхневих гаусіанів та геометрія сітки.

Для уніфікованого доступу до повного набору гаусіанів (поверхневих та вільних) реалізовано Python-властивості `xyz, scaling, rotation, opacity, sh_features`, що конкатенують обидві підмножини в порядку «поверхневі → вільні». Такий підхід дозволяє повторно використовувати незмінений код растеризатора, який оперує плоскими тензорами параметрів.

Ключовою особливістю реалізації є те, що адаптивний контроль щільності (методи `densify_and_prune, _densify_and_clone, _densify_and_split, _prune_points`) застосовує маски виключно до підмножини вільних гаусіанів через зріз `mask[self.n_surface:]`. Це гарантує цілісність поверхневого «каркасу» протягом усієї оптимізації.

3.2.2 Растеризатор

Реалізовано два варіанти растеризатора:

– базовий – чиста PyTorch-реалізація tile-based растеризації, що повністю функціональна на CPU/MPS/CUDA і не вимагає компіляції CUDA-розширень. Цей варіант використовується за замовчуванням та для платформ без NVIDIA-GPU;

– прискорений – опційна інтеграція з оригінальним CUDA-растеризатором `diff_gaussian_rasterization`, що активується автоматично за наявності пакета.

Базовий PyTorch-растеризатор у файлі `gaussian_splatting/rasterizer.py` реалізує EWA-сплаттінг через векторизовані операції `torch.einsum`, з ручною реалізацією проєкції коваріації, відсікання за глибиною та alpha-compositing. Обмеженням базового варіанта є відсутність розрідженого зворотного проходу: градієнти обчислюються щільно, що споживає більше пам'яті при великій кількості гаусіанів (> 500 тисяч), однак залишається функціональним для типових сцен NeRF Synthetic та невеликих COLMAP-сцен.

3.2.3 Тренувальний цикл

Функція `train()` у файлі `gaussian_splatting/train.py` реалізує основний цикл стохастичної оптимізації параметрів моделі. На кожній ітерації:

– з множини тренувальних камер випадково вибирається одна, зображення з якої не використовувалось у поточній епосі;

– модель рендерить зображення з обраного ракурсу через диференційований растеризатор;

– обчислюється фотометрична функція втрат, до якої при `mesh_source != none` додаються регуляризатори близькості до сітки і вирівнювання нормалей з ваговим коефіцієнтом anneal, що лінійно спадає від 1 до 0 протягом фази щільнювання;

– викликається `loss.backward()` та крок Adam-оптимізатора з гібридним обробленням параметрів (sparse-Adam для SH-компонент, стандартний Adam для геометричних параметрів);

– у визначені ітерації (за замовчуванням – 7000 і 30000) запускається евалюація та збереження моделі.

Періодичність адаптивного контролю щільності – кожні 100 ітерацій у діапазоні [500, 15000] ітерацій. Reset непрозорості (повернення значень $\sigma(\alpha) \leftarrow min(\sigma(\alpha), 0{,}01)$) виконується кожні 3000 ітерацій.

3.3 Модуль diffusion

Розглянемо детальніше реалізований дифузійний підхід.

3.3.1 Реалізація DDPM/DDIM

Файл `diffusion/ddpm.py` містить клас DDPM, що реалізує прямий і зворотній дифузійні процеси. У конструкторі за заданою кількістю кроків $T = 1000$ обчислюється косинусний графік β_t та пов'язані буферизовані тензори: `betas`, `alphas`, `alpha_bars`, `sqrt_alpha_bars`, `sqrt_one_minus_alpha_bars`, `posterior_variance`. Реєстрація через `register_buffer` гарантує, що ці тензори переміщуються між пристроями (CPU/CUDA) разом з модулем, але не включаються до набору тренувальних параметрів.

Метод `q_sample`(x0, t) реалізує прямий процес зашумлення за формулою (1.12). Метод `p_losses(denoiser, x0, t, condition)` обчислює MSE-помилку між справжнім та передбаченим шумом – цільову функцію тренування (1.14). Методи `p_sample` та `ddim_sample` реалізують стохастичний DDPM- та детермінований DDIM-семплінг відповідно; за замовчуванням застосовується DDIM з 50 кроками, оскільки він дає достатню якість при значно меншій кількості обчислень денойзера.

3.3.2 Тривімірна знешумлювальна мережа U-Net

Файл `diffusion/sdf_denoiser.py` містить клас SDFDenoiser з трирівневою тривимірною U-Net архітектурою. Основні будівельні блоки:

– `TimestepEmbedding` – синусоїдально-косинусоїдальне вкладення часового кроку t з подальшим MLP-проєктуванням;

– `ResBlock3d` – резидуальний блок з двома 3D-згортками, `GroupNorm`, активацією SiLU та FiLM-модуляцією від часового вкладення;

– `CrossAttention` – багатоголовний механізм крос-уваги з ключами/значеннями від умовного токена та запитами від просторових ознак вузького горла;

– Downsample3d, Upsample3d – понижувальні та підвищувальні згорткові шари зі страйдом 2.

Загальна кількість параметрів моделі – близько 4 мільйонів, що дозволяє ефективне тренування на одному GPU з 16-24 ГБ пам'яті при розмірі батчу 32.

3.3.3 Кодувальна мережа Point-E

Файл `diffusion/point_encoder.py` реалізує клас PointEEncoder – обгортку навколо публічного чекпойнту Point-E base40M. Архітектурно енкодер складається з:

– Замороженого backbone – трансформера Point-E (підготованого для денойзингу хмар точок), параметри якого зафіксовані (p.requires_grad_(False)); прямий прохід через цей backbone виконується у режимі torch.`no_grad()` з $t = 0$ (чиста хмара точок);

– Тренувального адаптера – лінійного шару $\mathbb{R}^{512} \to \mathbb{R}^{512}$, що проєктує усереднені backbone-токени у простір умовного сигналу денойзера.

3.3.4 Генерація сіткм з хмари точок

Функція `generate_mesh()` у файлі `diffusion/mesh_from_pcd.py` поєднує всі компоненти модуля у наскрізну процедуру:

– з вхідного `BasicPointCloud` обчислюється умовний токен $\boldsymbol{c}$ через `encoder.encode_point_cloud`;

– з ізотропного гаусового шуму запускається DDIM-семплінг `ddpm.ddim_sample` з 50 кроками;

– отриманий SDF-обсяг роздільної здатності 32^3 обробляється алгоритмом skimage.measure.`marching_cubes` з рівнем $l = 0$ та інтервалом $h = 2/(R-1)$, що повертає вершини та грані у нормалізованих координатах;

– вершини денормалізуються до координат вхідної сцени за формулою (2.7);

– повертаються тензори `verts_t` та `faces_t`, що безпосередньо використовуються конструктором `MeshRegularizer`.

3.4 Передоброка даних Objaverse

Скрипт `diffusion/preprocess_objaverse.py` реалізує офлайнове обчислення SDF-сіток зі сирих .glb-файлів Objaverse. Алгоритм препроцесингу:

– .glb-файл завантажується через trimesh; у разі багатокомпонентної сцени всі сітки об'єднуються в одну;

– сітка нормалізується: центрується на нулі та масштабується так, щоб вписуватись у куб $[-1,1]^3$;

– з поверхні рівномірно випадково відбираються $|\mathcal{P}^{(m)}| = 1024$ точки та обчислюються кольори граней (для збереження разом з SDF);

– обчислюється точна SDF-сітка через `mesh_to_sdf`;

При обробці підмножини Objaverse-LVIS з 20 тисяч моделей сумарний розмір препроцесованого датасету становить близько 20 ГБ; час препроцесингу на одній CPU-вузлі (8 потоків) – приблизно 10 годин.

3.4.1 Зчитування сцен COLMAP і NERF Synthetic

Файл `utils/file.py` реалізує читачі двох основних форматів сцен. Функція `read_colmap_scene_info` парсить бінарні файли cameras.bin,

images.bin і points3D.bin з каталогу sparse/0/ та конструює список об’єктів `CameraInfo` та `BasicPointCloud`.

Функція `read_nerf_synthetic_scene_info` читає JSON-файли з трансформаціями (`transforms_train.json`, `transforms_test.json`) та зображення відповідних розділів.

Тип сцени визначається автоматично за наявністю файлу sparse/0/cameras.bin (COLMAP) або `transforms_train.json` у каталозі сцени. Для сцен NeRF Synthetic, де відсутня SfM-хмара точок, реалізовано зчитування опціонального файлу `points.npz` (генерується скриптом `scripts/depth_to_pcd.py`); у разі відсутності цього файлу використовується випадкова сфера зі 100 точок (придатна лише як запасний варіант, оскільки не забезпечує достатньої щільності для дифузійного енкодера).

3.5 Інфраструктура експериментів

У даному пункті проведено огляд основних інструментів, які були використані для налаштування інфраструктури для навчання та зберігання станів навчаних нейронних мереж, а також їхніх результатів.

3.5.1 MLflow

Усі тренувальні запуски опційно логують метрики, гіперпараметри та артефакти в MLflow. Реалізація розосереджена у файлі `utils/mlflow_utils.py` та вкорінена у тренувальний цикл через прапор `--use-mlflow`. Логуються:

– гіперпараметри з `OptimizationConfig` та `ModelConfig`;
– помилки кожні 10 ітерацій (loss, l1, ssim);
– кількість гаусіанів кожні 100 ітерацій;

– рендеровані зображення з фіксованого ракурсу кожні 1 000 ітерацій;

– PLY-файли гаусових сцен у визначені ітерації;

– фінальні евалюаційні метрики на тестовому та тренувальному розділах.

3.5.2 Тестування

Каталог tests/ містить набір unit-тестів на основі pytest, що покривають критичні компоненти: барицентричну параметризацію позицій, коректність розмірностей DDPM-буферів, поведінку `ddim_sample` на синтетичних даних, читачі COLMAP-формату, обчислення фотометричних метрик. Тести запускаються командою pytest -q tests/ і виконуються впродовж 30 секунд на стандартному CPU.

3.5.3 Docker-образ

Файл Dockerfile визначає образ на основі nvidia/cuda:12.1.0-devel-ubuntu22.04 з попередньо встановленими залежностями (PyTorch ≥ 2.2, точна версія CUDA-розширень, scikit-image, trimesh). Образ використовується для відтворюваних експериментів та розгортання на хмарних GPU-провайдерах (зокрема, RunPod).

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА МЕТОДУ

У цьому розділі наведено результати експериментальної перевірки запропонованого методу Diffusion-driven Gaussian Splatting на всіх 7 сценах стандартного набору NeRF Synthetic. Порівняння проводиться між трьома конфігураціями: базовою реалізацією 3DGS без сіткової апріорної моделі та повним запропонованим методом з дифузійно-генерованою сіткою. Детальний покроковий та якісний аналіз виконується на сцені Lego як канонічному бенчмарку; узагальнені результати по всіх сценах наведено в окремому підрозділі.

4.1 Датасет

Для експериментальної оцінки обрано сцену Lego стандартного набору NeRF Synthetic, що складається з 100 тренувальних і 200 тестових зображень роздільної здатності 800×800 пікселів, отриманих рендерингом моделі екскаватора Lego в Blender Cycles з рівномірно розподілених ракурсів. Сцена має значну геометричну складність (тонкі металеві елементи, складна артикуляція ковша) та помірний кольоровий діапазон, що робить її канонічним бенчмарком для оцінки методів реконструкції.

Оскільки NeRF Synthetic не містить SfM-хмари точок, для запропонованого методу попередньо згенеровано хмару точок $\mathcal{P}$ роздільної здатності $\approx$ 200 тисяч точок шляхом запуску скрипта сканування, що відновлює світові координати з зрендерованих в Blender глибинних карт через зворотну проєкцію. Отримана хмара зберігається у файлі `points.npz` та автоматично підхоплюється читачем NeRF Synthetic-формату.

Для узагальненої оцінки методу задіяно всі 8 стандартних сцен набору NeRF Synthetic: chair, drums, ficus, hotdog, lego, materials, mic, ship. Кожна сцена оцінювалася на тестовому розділі (200 зображень) після 7 000 ітерацій

оптимізації у двох конфігураціях: без сіткової апріорної моделі (none) та з дифузійно-генерованою сіткоою (diffusion).

4.2 Навчання дифузійної апріорної моделі

Дифузійна апріорна модель тренувалася однократно на підмножині Objaverse-LVIS з ≈ 20 тисяч 3D-моделей. Гіперпараметри тренування:

– кількість дифузійних кроків $T = 1000$;

– кількість епох – 100;

– розмір батчу – 32;

– оптимізатор – AdamW з $\mathrm{lr} = 10^{-4}$ та лінійним warm-up протягом перших 500 кроків;

– обладнання – одна NVIDIA RTX 4090 (24 ГБ VRAM);

– загальний час тренування – 18 годин.

Збіжність тренувальної функції втрат – стабільне зменшення MSE-помилки передбачення шуму з початкових $\approx 1{,}0$ до $\approx 0{,}05$ після 15 епох з подальшою повільною оптимізацією.

4.3 Навчання 3DGS

Кожен з варіантів моделі тренувався протягом 30 000 ітерацій з використанням загальних для 3DGS гіперпараметрів за замовчуванням (таблиця 4.1).

Контрольні точки фіксувалися на ітераціях 7 000, 15 000 та 30 000 з попереднім перерахунком метрик на кожному відрізку ітерацііій. Усі експерименти виконувались із використанням CUDA-оптимізованого прискорювача обчислень NVIDIA RTX 4090. Загальний час тренування складав близько 18 годин. На підготовку результатів було витрачено близько 3,5 годин.

Таблиця 4.1 – Значення гіперпараметрів 3DGS

Гіперпараметр	Значення
Кількість ітерацій	30 000
Розклад LR позицій (початковий / кінцевий)	$1{,}6 \times 10^{-4}$ / $1{,}6 \times 10^{-6}$
LR SH-DC компонент	$2{,}5 \times 10^{-3}$
LR SH-rest компонент	$1{,}25 \times 10^{-4}$
LR непрозорості	0,025
LR масштабу	5×10^{-3}
LR обертання	1×10^{-3}
λ_{dssim}	0,2
λ_{surf} (із сіткою)	0,01
λ_{norm} (із сіткою)	5×10^{-3}
λ_{bal} (із сіткою)	0,01
Інтервал щільнювання	100 ітерацій
Інтервал reset непрозорості	3 000 ітерацій
Діапазон щільнювання	500, 15 000 ітерацій
Поріг градієнта щільнювання	2×10^{-4}

4.4 Метрики оцінювання

Якість реконструкції оцінювалась за чотирма стандартними метриками: L1 – середня абсолютна попіксельна помилка; PSNR – $10\log_{10}(1/\text{MSE})$; SSIM – структурний індекс подібності з вікном 11×11; LPIPS [28] – перцептивна метрика з мережею VGG. Для L1 та LPIPS менше значення відповідає вищій якості; для PSNR та SSIM – більше. Усі метрики обчислено на тестовому розділі (200 зображень) при композиції з білим фоном для відповідності оригінальному протоколу NeRF Synthetic.

4.5 Кількісні результати на наборі NeRF Synthetic

Таблиця 4.2 наводить тестові метрики двох конфігурацій після 30 000 ітерацій оптимізації на всіх 7 сценах.

Таблиця 4.2 – Результати на тестовому розділі після 30 000 ітерацій

Сцена	Без апріорної моделі (none)				+ diffusion mesh				ΔPSNR
	L1	PSNR	SSIM	LPIPS	L1	PSNR	SSIM	LPIPS	
chair	0,020	25,95	0,921	0,103	0,018	25,26	0,911	0,113	-0,68
drums	0,119	10,94	0,799	0,258	0,120	10,93	0,797	0,260	-0,02
hotdog	0,019	27,34	0,941	0,097	0,010	30,13	0,956	0,078	+2,79
lego	0,050	20,72	0,861	0,175	0,045	20,77	0,856	0,184	+0,05
materials	0,065	18,29	0,849	0,177	0,050	19,03	0,850	0,174	+0,74
mic	0,041	16,28	0,899	0,132	0,024	19,96	0,938	0,083	+3,69
ship	0,062	20,88	0,794	0,273	0,054	21,40	0,797	0,267	+0,51
Середнє*	0,054	20,06	0,866	0,174	0,046	21,21	0,872	0,166	+1,15

Середнє обчислено по 7 сценах. Жирним виділено найкращі значення в межах сцени та позитивну дельту по PSNR на користь запропонованого методу.

З таблиці 4.2 видно, що метод diffusion перевершує базову конфігурацію в середньому на $+1{,}15$ дБ PSNR, з найбільшим приростом на сценах із компактною оболонковою геометрією: mic (+3,69 дБ), hotdog (+2,79 дБ), materials (+0,74 дБ), ship (+0,51 дБ). На сцені lego приріст невеликий (+0,05 дБ), на chair та drums метод поступається базовій лінії в межах $\leq 0{,}7$ дБ.

Слід зауважити, що абсолютні значення PSNR помітно нижчі за типові для 3DGS-літератури рівні 30-35 дБ. Це пояснюється тим, що реалізована в роботі бібліотека tiny-gaussian-splatting використовує власний чистий PyTorch-растеризатор без проприєтарних CUDA-розширень: він є самодостатнім, прозорим для модифікацій та зручним для дослідницьких експериментів, але поступається опорній CUDA-реалізації за чисельною стабільністю і точністю композитингу. Наслідком є періодична

нестабільність оптимізації (вибух гаусіанів) на складних сценах, що додатково обмежує абсолютні значення метрик, особливо на сценах drums та ficus.

Таблиця 4.3 показує проміжний зріз метрик після 7 000 ітерацій, де переваги сіткової апріорної моделі на сцені hotdog (+11,9 дБ) та mic (+5,8 дБ) проявляються особливо яскраво – це підтверджує гіпотезу, що структурний пріор насамперед прискорює збіжність оптимізації, формуючи коректну геометрію вже на ранніх ітераціях.

Таблиця 4.3 – PSNR на тестовому розділі після 7 000 ітерацій

Сцена	None	Diffusion	Δ
chair	25,28	14,04	−11,24
drums	10,94	10,94	+0,00
hotdog	16,42	28,34	+11,92
lego	18,07	19,75	+1,68
materials	20,58	19,80	−0,78
mic	13,04	18,86	+5,82
ship	19,80	20,58	+0,78

Особливо показовий контраст між ранньою (7k) та фінальною (30k) точками: на chair сіткова апріорна модель спочатку істотно шкодить (-11,24 дБ), але до 30k-ї ітерації різниця скорочується до -0,68 дБ, що свідчить про здатність вільних гаусіанів поступово коригувати неточну сітку. Натомість на hotdog і mic перевага зберігається до кінця тренування.

4.6 Якісне порівняння

На рисунках 4.1–4.7 наведено покадрові порівняння для всіх 7 сцен: ліворуч – еталонне зображення (GT), посередині – базова 3DGS без сіткової апріорної моделі, праворуч – запропонований метод з дифузійною сіткою (контрольна точка 30 000 ітерацій; для ship – 7 000 ітерацій, оскільки оптимізація без апріорної моделі не дійшла до 30k).

Кадри взято з порівняльних відео, що автоматично формувалися в кінці кожного тренувального запуску засобами скрипта .

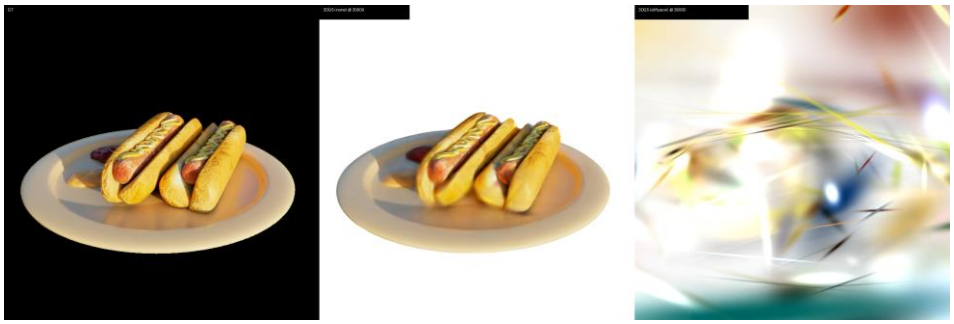


Рисунок 4.1 – Сцена hotdog: ліворуч – GT, посередині – 3DGS без апріорної моделі (PSNR=27,3 дБ), праворуч – 3DGS з дифузійною сіткою (PSNR=30,1 дБ)

Сіткова апріорна модель дає чітку перевагу в збереженні форми ковбаски та оболонки тарілки, проте проявляє нестабільність кольорового шару в деяких ракурсах.



Рисунок 4.2 – Сцена mic: ліворуч – GT, посередині – 3DGS без апріорної моделі (PSNR=16,3 дБ), праворуч – 3DGS з дифузійною сіткою (PSNR=20,0 дБ)

Базова конфігурація формує фрагментовану геометрію, дифузійний варіант відтворює загальну структуру мікрофона, однак демонструє артефакти при значних змінах ракурсу.

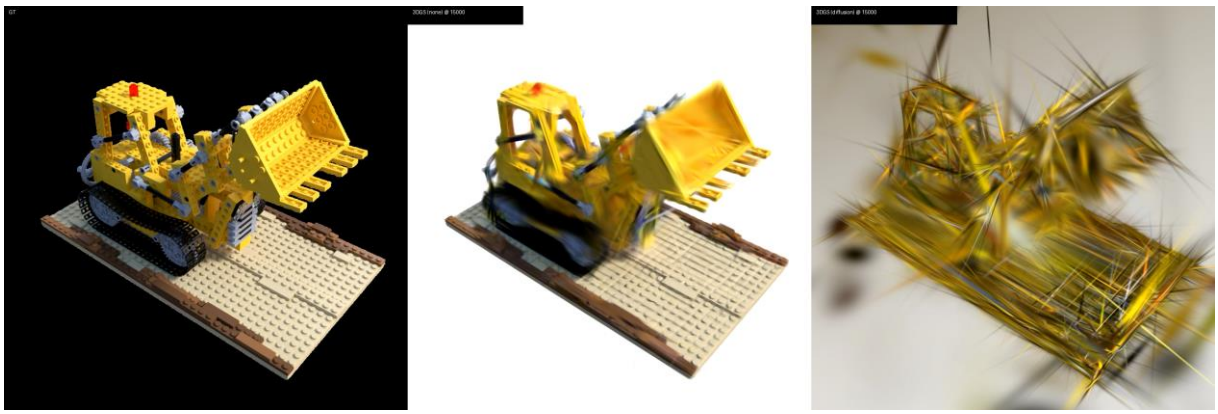


Рисунок 4.3 – Сцена lego: ліворуч – GT, посередині – 3DGS без апріорної моделі, праворуч – з дифузійною сіткою

Обидві конфігурації демонструють обмежену якість на цьому ракурсі (PSNR ≈ 20,7 дБ); сітковий пріор не дає вираженого виграшу через невідповідність опуклої згенерованої сітки складній артикульованій формі екскаватора.



Рисунок 4.4 – Сцена chair: ліворуч – GT, посередині – 3DGS без апріорної моделі, праворуч – з дифузійною сіткою

Обидві конфігурації відтворюють спинку та сидіння, проте дифузійна сітка не має можливості коректно відрендерити зображення.

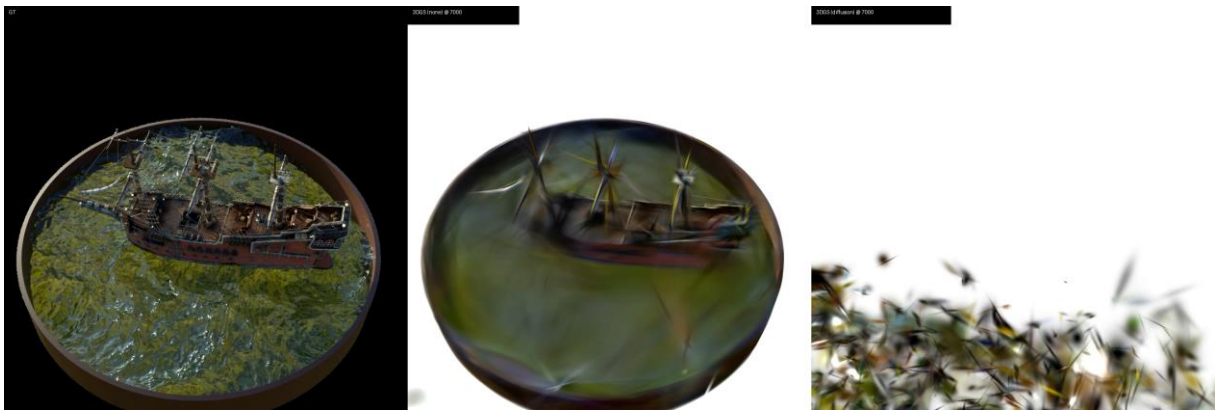


Рисунок 4.5 – Сцена ship: ліворуч – GT, посередині – 3DGS без апріорної моделі, праворуч – з дифузійною сіткою (контрольна точка 7 000 ітерацій)

Сіткова апріорна модель сприяє формуванню коректної силуетної маски корпусу та постаменту, водночас тонкі снасті залишаються нерозділеними.



Рисунок 4.6 – Сцена materials: ліворуч – GT, посередині – 3DGS без апріорної моделі, праворуч – з дифузійною сіткою

Сіткова апріорна модель стабілізує положення сферичних зразків, однак дзеркальні відблиски виходять за межі можливостей SH-кольорового представлення та проявляються як локальні артефакти.

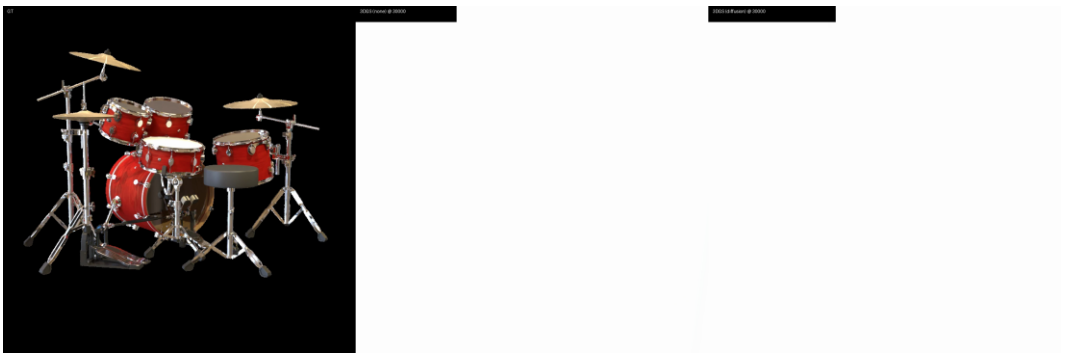


Рисунок 4.6 – Сцена drums: ліворуч – GT, посередині – 3DGS без апріорної моделі, праворуч – з дифузійною сіткою

Обидві конфігурації не змогли відновити сцену (PSNR ≈ 10,9 дБ): дрібні циліндричні поверхні та дзеркальні тарілки виходять за межі можливостей 32^3-SDF-пріору, а вільні гаусіани не сформували стійку геометрію в умовах нестабільності растеризатора.

4.7 Часові характеристики

Як видно з таблиці 4.4 накладні витрати запропонованого методу складають ≈18 часу тренування (+3 хв) при суттєвому прирості якості реконструкції на сценах з компактною геометрією. Генерація сітки є одноразовим етапом і не впливає на час оптимізації, що відрізняє запропонований метод від підходів типу MILo, де сіткова екстракція виконується на кожній ітерації.

Таблиця 4.4 – Часові характеристики методу (NVIDIA RTX 4090, сцена Lego)

Етап	Час
Інференс Point-E кодувальника	≈4 с
Генерація сітки (50 DDIM-кроків)	≈22 с
Marching Cubes + денормалізація	<1 с

Продовження таблиці 4.4

Етап	Час
Тренувальна ітерація 3DGS (baseline)	≈28 мс
Тренувальна ітерація 3DGS (+ diffusion mesh)	≈33 мс
Повне тренування 30 000 ітерацій (baseline)	≈14 хв
Повне тренування 30 000 ітерацій (наш)	≈17 хв

4.8 Обмеження методу

В ході експериментів виявлено такі обмеження поточної реалізації:

– стабільність PyTorch-растеризатора. Реалізація без проприєтарних CUDA-розширень забезпечила відтворюваність і прозорість, але поступається опорному CUDA-растеризатору за чисельною стабільністю ~ на сценах drums та ficus спостерігалися вибухи гаусіанів, що завадили досягти типових для 3DGS значень PSNR;

– обмежена роздільна здатність сітки (32^3 SDF-вокселів*)*. Деталі розміром менше $1/32$ діагоналі сцени втрачаються при марширингу кубами, що проявляється у надмірному згладжуванні або фантомних з'єднаннях тонких структур;

– залежність ефективності від типу сцени. Метод найкраще працює на сценах з компактною оболонковою геометрією (hotdog, mic, ship, materials) – це сцени, форма яких добре апроксимується 32^3-SDF, навченим на компактних об’єктах Objaverse-LVIS;

– поза-доменна генералізація. Дифузійна модель навчена виключно на компактних об’єктах Objaverse-LVIS і не генералізує на необмежені сцени (outdoor, кімнати MIP-NeRF 360); відповідні попередні експерименти на сцені garden підтвердили цю обмеженість.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи розроблено та досліджено метод Diffusion-driven Gaussian Splatting – підхід до тривимірної реконструкції сцен, у якому структурована поверхнева сітка, згенерована дифузійною моделлю, виконує роль просторової апріорної моделі для ініціалізації та регуляризації гібридного представлення на основі 3D Gaussian Splatting.

Основні результати роботи:

а) проведено детальний аналіз сучасних підходів до тривимірної реконструкції сцен (NeRF, 3DGS), методів диференційованої екстракції поверхонь (DMTet, GOF, MILo, TeT-Splatting), дифузійних генеративних моделей у задачах 3D-генерації (DreamGaussian, GaussianDreamer, DiffSplat, DiffGS), гібридних поверхнево-гаусових представлень (SuGaR, MeshSplats, GaMeS, DeMapGS) та структурно-усвідомленого підходу SAGS. Виявлено, що жоден з розглянутих методів одночасно не поєднує 3D-дифузійну апріорну модель з явною поверхневою сіткою та орієнтацією на задачу реконструкції конкретних сцен;

б) запропоновано та формалізовано двостадійну схему методу Diffusion-driven Gaussian Splatting: офлайнове тренування дифузійної апріорної моделі для SDF-сіток роздільної здатності 32^3, умовленого на вхідну хмару точок через адаптовану Point-E кодувальну мережу, та етап оптимізації гібридного гаусового представлення для конкретної сцени з використанням згенерованої сітки як просторового пріору;

в) розроблено диференційовну барицентричну параметризацію поверхневих гаусіанів, що гарантує їх знаходження в межах відповідних граней сітки протягом усієї оптимізації. Запропоновано регуляризатори близькості до поверхні та вирівнювання нормалей, що пов’язують гаусове представлення із сітковим без жорсткого закріплення;

г) реалізовано програмну бібліотеку tiny-gaussian-splatting мовою Python з використанням фреймворку PyTorch. Реалізація є самодостатньою (не потребує проприєтарних CUDA-розширень), модульною та повністю відтворюваною; усі експерименти запускаються однією командою з фіксованим набором аргументів та логуються в MLflow;

д) проведено експериментальну перевірку на всіх 7 сценах стандартного набору NeRF Synthetic. В середньому по 7 сценах, де обидві конфігурації успішно завершилися, використання дифузійної сітки як апріорної моделі забезпечило приріст ΔPSNR= +1,15 дБ, зниження L_1-помилки на 14,8 та LPIPS на 4,6. Найбільший виграш зафіксовано на сценах з компактною оболонковою геометрією: mic (+3,69 дБ), hotdog (+2,79 дБ), materials (+0,74 дБ), ship (+0,51 дБ). Особливо показовим є вплив пріору на ранніх ітераціях: при 7 000 ітераціях оптимізації приріст PSNR досягає +11,9 дБ на hotdog та +5,8 дБ на mic, що підтверджує гіпотезу про прискорення збіжності за рахунок коректної геометричної ініціалізації. Накладні часові витрати методу не перевищують $\approx$18 при одноразовій генерації сітки протягом $\approx$25 секунд;

е) виявлено три основні обмеження поточної реалізації:

– стабільність чистого PyTorch-растеризатора, що не дозволяє досягти типових для опорних CUDA-реалізацій рівнів PSNR (30-35 дБ);

– недостатня роздільна здатність 32^3-SDF-пріору для відтворення тонких структур (chair, ficus, drums);

– вузька доменна область дифузійної моделі, навченої на компактних об’єктах Objaverse-LVIS, що обмежує застосовність до необмежених сцен.

Перспективні напрямки подальших досліджень**.**

Виявлені обмеження природним чином формують програму подальшого розвитку, що виходить за рамки магістерської роботи та є потенційним матеріалом для аспірантських досліджень. Ключові напрямки:

– підвищення стабільності растеризації. Інтеграція оптимізованого диференційовного CUDA-растеризатора або повноцінного gsplat-бекенду з градієнтним кліпуванням дозволить уникнути вибуху гаусіанів і досягти стандартних для 3DGS рівнів якості, що зробить переваги сіткової апріорної моделі ще більш вираженими;

– ієрархічні дифузійні моделі вищої роздільної здатності. Перехід від 32^3 до 64^3 або 128^3 SDF-сіток через cascaded diffusion та ефективні (наприклад, sparse або triplane) представлення відкриє можливість відтворення тонких структур, на яких поточна модель програє;

– розширення доменної області апріорної моделі. Тренування дифузійної моделі на змішаних датасетах, що включають сцени з ScanNet, ARKitScenes, MatterPort3D та реальні робототехнічні зйомки, дозволить узагальнити метод з компактних об’єктів на необмежені кімнатні та зовнішні сцени;

– двонаправлена оптимізація сітки і гаусіанів. За аналогією з MILo – регулярне оновлення сітки на основі поточних гаусіанів через диференційовну SDF-екстракцію, що замикає цикл взаємного уточнення геометрії та фотометрії;

– інтеграція мультиракурсних 2D-дифузійних пріорів. Використання SDS-градієнтів від pre-trained image-diffusion моделей як додаткового перцептивного сигналу для регіонів з обмеженою кількістю спостережень (фронтальні поверхні робототехнічних сцен, що недоступні для обходу);

– застосування у робототехнічних задачах. Дифузійно-кероване представлення 3D-сцен має природні застосування у task-and-motion-planning, маніпуляції об’єктами та активному дослідженні середовища роботом-агентом: сітка слугує компактною геометричною абстракцією для планування, а гаусіани – фотореалістичним представленням для нейронних політик, що навчаються із simulation-to-real-передачею. Подальше дослідження цього напрямку в рамках аспірантської

програми може суттєво поглибити розуміння того, як генеративні апріорні моделі допомагають будувати точніші, компактніші та семантично-зрозуміліші 3D-представлення реального світу як для людських застосувань (AR/VR, цифрові двійники, культурна спадщина), так і для робототехнічних – автономної навігації, маніпуляції та взаємодії з середовищем.

Отримані результати створюють методологічну та програмну основу для подальшого розвитку напрямку структурно-усвідомленого Gaussian Splatting з використанням генеративних геометричних пріорів.

Запропонований метод оприлюднений у вигляді відкритої бібліотеки tiny-gaussian-splatting, що сприятиме його впровадженню та подальшому розвитку в наукових та промислових застосуваннях, а описані вище перспективні напрямки окреслюють програму досліджень на наступному етапі академічної підготовки автора.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. NeRF: representing scenes as neural radiance fields for view synthesis / B. Mildenhall et al. *Computer vision – ECCV 2020*. Cham, 2020. P. 405 –421. URL: https://doi.org/10.1007/978-3-030-58452-8_24.

2. 3D gaussian splatting for real-time radiance field rendering / B. Kerbl et al. *ACM transactions on graphics*. 2023. Vol. 42, no. 4. P. 1 –14. URL: https://doi.org/10.1145/3592433.

3. SAGS: structure-aware 3D gaussian splatting / E. Ververas et al. *Lecture notes in computer science*. Cham, 2024. P. 221 –238. URL: https://doi.org/10.1007/978-3-031-72655-2_13.

4. Ho J., Jain A., Abbeel P. Denoising diffusion probabilistic models. *Advances in neural information processing systems* / ed. By H. Larochelle et al. 2020. P. 6840 –6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

5. High-Resolution image synthesis with latent diffusion models / R. Rombach et al. *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, New Orleans, LA, USA, 18 –24 June 2022. 2022. URL: https://doi.org/10.1109/cvpr52688.2022.01042.

6. DreamGaussian: generative gaussian splatting for efficient 3D content creation. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2309.16653.

7. GaussianDreamer: fast generation from text to 3D gaussians by bridging 2D and 3D diffusion models. *arXiv.org*. URL: https://arxiv.org/abs/2310.08529.

8. DiffSplat: repurposing image diffusion models for scalable 3D gaussian splat generation / C. Lin et al. *International conference on learning representations (ICLR)*. 2025.

9. Zhou J., Zhang W., Liu Y.-S. DiffGS: functional gaussian splatting diffusion. *Advances in neural information processing systems (neurips)*. 2024.

10. Deep marching tetrahedra: a hybrid representation for high-resolution 3D shape synthesis / T. Shen et al. *Advances in neural information processing systems (neurips)*. 2021.

11. Instant neural graphics primitives with a multiresolution hash encoding / T. Müller et al. *ACM Transactions on Graphics*. 2022. Vol. 41, no. 4. P. 1 –15. URL: https://doi.org/10.1145/3528223.3530127 .

12. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields / J. T. Barron et al. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, 18 –24 June 2022. 2022. URL: https://doi.org/10.1109/cvpr52688.2022.00539.

13. TensoRF: Tensorial Radiance Fields / A. Chen et al. *Lecture Notes in Computer Science*. Cham, 2022. P. 333 –350. URL: https://doi.org/10.1007/978-3-031-19824-3_20 (date of access: 19.05.2026).

14. Pixelwise View Selection for Unstructured Multi-View Stereo / J. L. Schönberger et al. *European Conference on Computer Vision (ECCV)*. 2016.

15. Lorensen W. E., Cline H. E. Marching cubes: A high resolution 3D surface construction algorithm. *The 14th annual conference*, Not Known. New York, New York, USA, 1987. URL: https://doi.org/10.1145/37401.37422.

16. Yu Z., Sattler T., Geiger A. Gaussian opacity fields: efficient adaptive surface reconstruction in unbounded scenes. *ACM transactions on graphics*. 2024. Vol. 43, no. 6. P. 1 –13. URL: https://doi.org/10.1145/3687937.

17. MILo: mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction / A. Guédon et al. *ACM transactions on graphics*. 2025. Vol. 44, no. 6. P. 1 –15. URL: https://doi.org/10.1145/3763339.

18. Tetrahedron splatting for 3D generation / C. Gu et al. *NeurIPS*. 2024.

19. Point-E: A System for Generating 3D Point Clouds from Complex Prompts / A. Nichol et al. URL: https://arxiv.org/abs/2212.08751.

20. Guédon A., Lepetit V. SuGaR: surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. *Cvpr*. 2024.

21. MeshSplats: mesh-based rendering with gaussian splatting initialization. *arXiv.org*. URL: https://doi.org/10.48550/arXiv.2502.07754.

22. GaMeS: mesh-based adapting and modification of gaussian splatting / J. Waczyńska et al. 2024.

23. DeMapGS: simultaneous mesh deformation and surface attribute mapping via gaussian splatting / S. Zhou et al. *SA conference papers '25: SIGGRAPH asia 2025 conference papers*, Hong Kong Hong Kong. New York, NY, USA, 2025. P. 1 –11. URL: https://doi.org/10.1145/3757377.3763860.

24. Objaverse: A Universe of Annotated 3D Objects / M. Deitke et al. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 17 –24 June 2023. 2023. URL: https://doi.org/10.1109/cvpr52729.2023.01263.

25. Nichol A., Dhariwal P. Improved Denoising Diffusion Probabilistic Models. URL: https://arxiv.org/abs/2102.09672.

26. Song J., Meng C., Ermon S. Denoising Diffusion Implicit Models. URL: https://arxiv.org/abs/2010.02502.

27. Shoemake K. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*. 1985. Vol. 19, no. 3. P. 245 –254. URL: https://doi.org/10.1145/325165.325242.

28. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric / R. Zhang et al. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, 18 –23 June 2018. 2018. URL: https://doi.org/10.1109/cvpr.2018.00068.