Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет **СУ и Р**

Образовательная программа Робототехника и искусственный интеллект

О Т Ч Е Т

о производственной практике, научная-исследовательская

Тема задания: **Геометрически правдоподобный семантический Gaussian Splatting**

Обучающийся Муравья Никита Романович, гр. R3338

Руководитель практики от университета: **Бойцев Антон Александрович, доцент «Высшей школы цифровой культуры»**

Санкт-Петербург
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы — исследование и применение 3D-гауссовского сплаттинга для создания качественных 3D-реконструкций с использованием методов машинного обучения. Для этого необходимо:

— Изучить статью по 3D-сплаттингу и текущее состояние исследований;

— Рассмотреть математическую модель: гауссианы и рендеринг;

— Освоить инструменты Yandex DataSphere для запуска на GPU;

— Реализовать реконструкцию с использованием оригинального репозитория `gaussian-splatting`.

Полученные в ходе работы навыки имеют непосредственное применение в задачах робототехники:

— **Навигации** — построение карт по видеопотоку;

— **Распознавания** — высокая детализация сцены;

— **Симуляций** — быстрый рендеринг для обучения агентов;

— **Обработки LiDAR** — представление облаков точек через гауссианы.

Yandex DataSphere обеспечивает доступ к ускоренным вычислениям и упрощает запуск ресурсоёмких задач, что особенно важно для работы в реальном времени.

1 Математическая основа гауссовского сплаттинга

В этом разделе излагаются ключевые математические идеи, лежащие в основе метода 3D-гауссовского сплаттинга. Подход строится на представлении сцены как параметризованного семейства гауссиан в трёхмерном пространстве, что позволяет заменить традиционные методы 3D-реконструкции (NeRF, полигональные модели) более гибким и дифференцируемым способом визуализации.

1.1 Параметризация гауссовых сплаттов

Каждый элемент сцены моделируется гауссианой с параметрами:

— $\mu \in \mathbb{R}^3$ — центр гауссианы в мировом пространстве;

— $\Sigma \in \mathbb{R}^{3\times3}$ — ковариационная матрица, определяющая форму и ориентацию распределения;

— $C \in \mathbb{R}^3$ — вектор цветовых компонент (RGB);

— $\alpha \in [0, 1]$ — непрозрачность.

Плотность в пространстве выражается через функцию:

$$f(\mathbf{x}) = C \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Эта формула задаёт цветовое распределение, которое проецируется на изображение при рендеринге.

1.2 Преобразования и проекция

1.2.1 Мировое и видовое пространство

В контексте компьютерной графики принято различать два основных пространства координат:

— **Мировое пространство** (*world space*) — глобальная система координат сцены, в которой задаются положения всех объектов, включая камеры и источники света. Каждая точка имеет координаты $\mathbf{x} \in \mathbb{R}^3$.

— **Видовое пространство** (*view space* или *camera space*) — система координат, связанная с конкретной виртуальной камерой. Камера помещается в начало координат, ось z направлена в сторону взгляда, а оси x и y — по горизонтали и вертикали изображения.

Переход из мирового пространства в видовое осуществляется матрицей $W \in \mathbb{R}^{4\times4}$:

$$\mathbf{x}' = W \cdot \mathbf{x}$$

Ковариационная матрица преобразуется аналогично:

$$\Sigma' = U\Sigma U^T$$

где U — верхний левый блок якобиана проекции, отражающий локальное искажение гауссианы при проецировании на экран.

1.2.2 Проецирование на изображение

Для получения двумерной проекции используется матрица камеры:

$$\mathbf{p} = \Pi \cdot \mathbf{x}'$$

где Π — матрица перспективного (или ортографического) проецирования.

1.3 Градиенты для оптимизации

1.3.1 Частные производные и правило цепочки

Оптимизация параметров гауссиан осуществляется методом градиентного спуска. При этом ключевую роль играют частные производные по параметрам масштабирования и поворота. В частности, производные преобразованной ковариационной матрицы Σ' вычисляются согласно правилу цепочки:

$$\frac{d\Sigma'}{ds} = \frac{d\Sigma'}{d\Sigma} \cdot \frac{d\Sigma}{ds}, \qquad \frac{d\Sigma'}{dq} = \frac{d\Sigma'}{d\Sigma} \cdot \frac{d\Sigma}{dq}$$

Частная производная по элементу Σ_{ij} имеет следующий аналитический вид:

$$\frac{\partial \Sigma'}{\partial \Sigma_{ij}} = \begin{pmatrix} U_{1i}U_{1j} & U_{1i}U_{2j} \\ U_{2i}U_{1j} & U_{2i}U_{2j} \end{pmatrix}$$

где U — матрица проекции в экранное пространство.

1.3.2 Градиенты по кватерниону поворота

Матрица поворота $R(q)$, соответствующая кватерниону $q = (q_r, q_i, q_j, q_k)$, определяется следующим выражением:

$$R(q) = 2 \begin{pmatrix} \frac{1}{2} - (q_j^2 + q_k^2) & q_i q_j - q_r q_k & q_i q_k + q_r q_j \\ q_i q_j + q_r q_k & \frac{1}{2} - (q_i^2 + q_k^2) & q_j q_k - q_r q_i \\ q_i q_k - q_r q_j & q_j q_k + q_r q_i & \frac{1}{2} - (q_i^2 + q_j^2) \end{pmatrix}$$

Эта форма обеспечивает непрерывное и дифференцируемое задание поворота в трёхмерном пространстве.

1.3.3 Градиенты по масштабу

Дифференцирование по компоненте масштабирования s_k даёт:

$$\frac{\partial M_{ij}}{\partial s_k} = \begin{cases} R_{ik}, & j = k \\ 0, & \text{иначе} \end{cases}$$

где $M = R \cdot S$ — результирующая матрица линейного преобразования, представляющая собой произведение матрицы поворота R и диагональной матрицы масштабирования S.

1.4 Алгоритм оптимизации

Ниже приведён псевдокод алгоритма оптимизации гауссиан, который включает инициализацию параметров, итеративный рендеринг и вычисление потерь, а также динамическое управление плотностью и качеством гауссиан. Основные шаги:

— Инициализация позиций, ковариаций, цветов и прозрачностей на основе исходных данных SfM.

— Итеративное обновление параметров с помощью градиентного спуска (Adam), минимизирующего функцию потерь между текущим и целевым изображениями.

— Периодическая проверка и корректировка гауссиан: удаление слабых или слишком больших, а также разбиение или клонирование для улучшения детализации.

Листинг 1.1 — Gaussian Splatting Optimization Algorithm

```
procedure OptimizeGaussians()
    Initialize M ← initial positions from SfM
    Initialize S, C, A ← initial covariances, colors, opacities
    while not converged do
        V, I_target ← SampleTrainingView()
        I ← Render(M, S, C, A, V)
        L ← ComputeLoss(I, I_target)
        (M, S, C, A) ← AdamStep(grad L)
        if IsRefinementIteration() then
            for each Gaussian (mu, Sigma, C, alpha) do
                if alpha < eps or IsTooLarge(Sigma) then
                    RemoveGaussian()
                else if Norm(grad L) > tau_p then
                    if ScaleNorm(Sigma) > tau_S then
                        SplitGaussian(mu, Sigma, C, alpha)
                    else
                        CloneGaussian(mu, Sigma, C, alpha)
```

2 Использование гауссовского сплаттинга для текстово-управляемого рендеринга

В данном разделе рассматривается реализация текстово-управляемого рендеринга на основе гауссовского сплаттинга. Описываются ключевые компоненты системы, этапы интеграции CLIP-модели и процесс оптимизации параметров для генерации 3D-сцен по текстовым запросам.

2.1 Архитектура реализации

Гауссовский сплаттинг реализован как модульная система с чётким разделением ответственности между компонентами:

— **Модель гауссиан.** Представляет сцену как набор 3D-гауссиан с параметрами:

— положение $\mathbf{p} \in \mathbb{R}^3$;

— ковариационная матрица $\Sigma \in \mathbb{R}^{3\times3}$;

— цвет $\mathbf{c} \in \mathbb{R}^3$;

— прозрачность $\alpha \in [0,1]$.

— **Пайплайн рендеринга.** Выполняет проекцию 3D-гауссиан на 2D-изображение с учётом:

— матрицы камеры W (world-to-camera transform);

— аффинного приближения при проекции на плоскость изображения;

— глубины и прозрачности для смешивания.

— **CLIP-интеграция.** Связывает текстовое описание и изображение через общее эмбеддинг-пространство:

— эмбеддинги текста и изображения проецируются в общее пространство;

— используется косинусное расстояние в качестве функции потерь;

— применяется аугментация через случайные кропы.

— **Оптимизатор.** Обновляет параметры гауссиан с помощью:

— градиентного спуска с использованием Adam;

— динамической адаптации плотности точек;

— периодического сброса прозрачности.

2.2 Выбор библиотек и инструментов

Для реализации были выбраны следующие библиотеки:

— **PyTorch** — основа вычислений:

```
import torch
from torch.cuda.amp import autocast
```

— **OpenCLIP** — предобученные модели:

```
import open_clip
model, _, _ = open_clip.create_model_and_transforms('ViT-B-32',
    pretrained='laion2b_s34b_b79k')
```

— **TorchVision** — преобразования изображений:

```
from torchvision.transforms import Normalize
clippp = Normalize(mean=model.visual.image_mean,
    std=model.visual.image_std)
```

— **NumPy** — для численных операций:

```
import numpy as np
```

— **Matplotlib + Celluloid** — визуализация прогресса:

```
from celluloid import Camera
```

2.3 Создание и инициализация гауссиан

2.3.1 Генерация начальной сцены

— **Генерация координат.** Случайные точки $\mathbf{xyz} \in \mathbb{R}^{N\times 3}$ на единичной сфере:

$$\mathbf{xyz}_{\text{norm}} = \frac{\mathbf{xyz}}{\|\mathbf{xyz}\|_2}, \quad N = 100,$$

с масштабированием на коэффициент 1.3.

— **Цвета через сферические гармоники (SH).** Используются SH-коэффициенты $\mathbf{shs} \in \mathbb{R}^{N\times 3}$:

$$\text{SH2RGB}(\mathbf{x}) = \frac{1}{1 + e^{-k\cdot\mathbf{x}}}.$$

— **Создание объекта `BasicPointCloud`.**

```
pcd = BasicPointCloud(points=xyz, colors=SH2RGB(shs),
    normals=np.zeros((num_pts, 3)))
```

2.4 Интеграция CLIP

2.4.1 Обработка эмбеддингов

— Загрузка модели:

```
model = torch.jit.script(model).requires_grad_(False).cuda().half()
```

— Нормализация изображений:

$$\mu = [0.481, 0.457, 0.407], \quad \sigma = [0.268, 0.261, 0.275].$$

— Аугментация кропами:

$$\mathbf{T} = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{bmatrix}, \quad s \in [0.7, 0.9],\ t_x, t_y \in [0, 1-s].$$

— Функция потерь:

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\mathbf{v}_i^\top \mathbf{t}}{\|\mathbf{v}_i\| \cdot \|\mathbf{t}\|}.$$

```
sim = img_vec @ text_vec.T
clip_loss = -sim.mean()
```

2.5 Пайплайн рендеринга

Листинг 2.1 — Rendering Pipeline for 3D Gaussians

```
procedure Render(w, h, M, S, C, A, V)
    (M_screen, S_screen) ← ScreenspaceGaussians(M, S, V)
    T ← TileGrid(w, h)
    (L, K) ← DuplicateWithKeys(M_screen, T)
    (L, K) ← SortByKeys(L, K)
    R ← ComputeTileRanges(T, K)
    I ← InitializeCanvas(w, h)
    for each tile t in T do
        for each pixel i in t do
            r ← GetTileRange(R, t)
            I[i] ← CompositeInOrder(i, L, r, K, M_screen,
                S_screen, C, A)
    return I
```

2.6 Оптимизация и обучение

— Настройка гиперпараметров:

```
args.iterations = 2500
args.position_lr_init = 1e-2
args.position_lr_final = 1e-5
```

— Обновление параметров:

```
gaussians.optimizer.step()
gaussians.optimizer.zero_grad(set_to_none=True)
```

— Динамическая адаптация плотности:

```
gaussians.densify_and_prune(opt.densify_grad_threshold, 0.005,
    camera_extent, size_threshold)
if iteration % opt.opacity_reset_interval == 0:
    gaussians.reset_opacity()
```

— Визуализация прогресса:

```
fig = plt.figure()
camera = PltCamera(fig)
camera.snap()
animation = camera.animate(blit=False, interval=50)
```

2.7 Пример использования

— Инициализация сцены: SH-цвета и случайные координаты.

— Рендеринг: с камеры, вращающейся вокруг объекта.

— Оптимизация: 2500 итераций градиентного спуска.

— Результат: визуализация в HTML5:

```
HTML(animation.to_html5_video())
```

2.8 Применение текстово-управляемого сплаттинга

В данной части представлена практическая применение генерации 3D-сцены на основе текстового описания, используя гауссовский сплаттинг и интеграцию с моделью CLIP. В качестве примера рассматривается создание модели подсолнухов Ван Гога.

2.8.1 Настройка и запуск обучения

Текстовый запрос задаёт сцену:

```
prompt = "a 3d model of Van Gogh's Sunflowers, 3d asset, high quality, not
    noisy, beautiful, black background"
```

Параметры модели, пайплайна и оптимизации инициализируются через объекты `ModelParams`, `OptimizationParams` и `PipelineParams`. Задаются основные гиперпараметры:

```
args.iterations = 2500
args.position_lr_init = 1e-2
args.position_lr_final = 1e-5
torch.manual_seed(2023)
```

Обучение запускается функцией `training()`, в которой происходит:

— сопоставление изображения и текста через CLIP;

— оптимизация параметров гауссиан с помощью Adam;

— динамическая адаптация плотности облака точек;

— визуализация процесса обучения.

После завершения итераций результат сохраняется в HTML5-видео:

```
1 HTML(animation.to_html5_video())
```

2.8.2 Результат и анализ модели

В результате реализованная система генерирует 3D-модель подсолнухов в виде облака гауссиан, соответствующего текстовому описанию. На рис. 2.1 представлен итог визуализации:

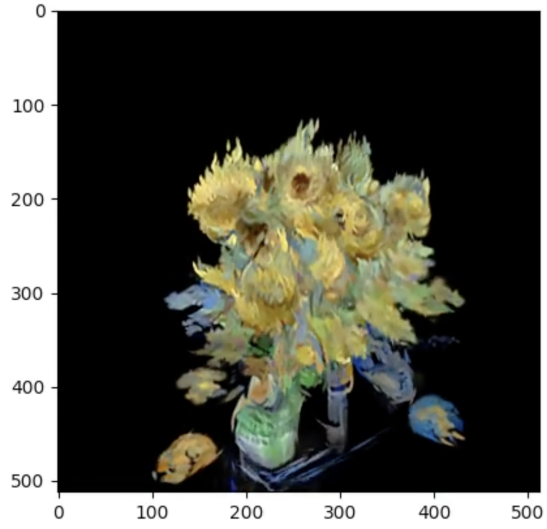

Рисунок 2.1 — 3D-модель подсолнухов Ван Гога, созданная с помощью текстово-управляемого гауссовского сплаттинга

Полученная модель:

— передаёт художественные особенности оригинальной сцены;

— поддерживает рендеринг с различных ракурсов;

— демонстрирует низкий уровень шума благодаря адаптивной плотности.

Таким образом, описанный подход подтверждает применимость гауссовского сплаттинга к задачам текстово-управляемой 3D-реконструкции, сочетая визуальную точность и эффективность дифференцируемого рендеринга.

ЗАКЛЮЧЕНИЕ

В ходе работы:

— Изучены основы гауссовского сплаттинга (см. раздел 1).

— Реализована текстово-управляемая генерация 3D-сцен с использованием CLIP (см. раздел 2).

— Построена 3D-модель подсолнухов Ван Гога (см. рис. 2.1), подтверждающая корректность реализации.

Применение в робототехнике

Разработанный метод может быть использован в следующих задачах:

— **Навигация и SLAM** — построение детализированных 3D-карт и оптимизация маршрутов.

— **Обнаружение объектов** — текстово-управляемое распознавание и локализация в сцене.

— **Симуляция и обучение** — генерация реалистичных сцен для обучения агентов.

— **Обработка LiDAR-данных** — сглаживание, реконструкция и представление облаков точек в виде гауссиан.

Преимущества и ограничения

Преимущества: высокая скорость рендеринга, поддержка динамических сцен, интеграция с языковыми моделями.
Ограничения: высокая вычислительная сложность, потребность в тонкой настройке гиперпараметров и производительном GPU.

Выводы

Разработана и реализована система текстово-управляемой 3D-реконструкции на основе гауссовского сплаттинга. В перспективе возможны улучшения, направленные на:

— оптимизацию под мобильные и встроенные устройства;

— интеграцию с физическими симуляторами;

— реализацию потоковой обработки в реальном времени.

Репозиторий с исходными материалами

Все исходные материалы, включая код, модели и примеры, находятся в открытом доступе на GitHub по адресу:

`gaussian_splatting_semantic`

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. 3D Gaussian Splatting for Real-Time Radiance Field Rendering / Kerbl B., Kopanas G., Leimkühler T., and Drettakis G. — arXiv : arXiv:2308.04079, 2023.

2. Kerbl B. et al. graphdeco-inria/gaussian-splatting: Official repository. — GitHub : GitHub, 2023.

3. Zwicker M. et al. EWA Volume Splatting. — VIS Conference : IEEE, 2001.

4. Project Nerfstudio. gsplat: Differentiable Gaussian Splatting library. — GitHub : GitHub, 2024.

5. Project Nerfstudio. gsplat Documentation. — gsplat.studio : gsplat.studio, 2024.

6. Face Hugging. Gaussian Splatting: A New Era in Real-Time 3D Rendering. — Hugging Face Blog : Hugging Face, 2024.

7. Хабр. 3D Gaussian Splatting: Реализация и применение. — Хабр : Хабр, 2023.

8. Ye V. A Python Engineer's Introduction to 3D Gaussian Splatting (Part 1). — Towards Data Science : Medium, 2023.

9. Ye V. A Python Engineer's Introduction to 3D Gaussian Splatting (Part 2). — Towards Data Science : Medium, 2023.

10. Ye V. A Python Engineer's Introduction to 3D Gaussian Splatting (Part 3). — Towards Data Science : Medium, 2024.