

# 3D Gaussian Splatting для рендеринга радиусных полей в реальном времени

БЕРНХАРД КЕРБЛ\*, Inria, Université Côte d’Azur, Франция

ГЕОРГИОС КОПАНАС\*, Inria, Université Côte d’Azur, Франция

ТОМАС ЛЕЙМКЮЛЕР, Max-Planck-Institut für Informatik, Германия

ДЖОРДЖ ДРЕТТАКИС, Inria, Université Côte d’Azur, Франция



Fig. 1. Наш метод достигает рендеринга радиусных полей в реальном времени с качеством, равным предыдущему методу с наилучшим качеством [Barron et al. 2022], требуя при этом времени оптимизации только конкурентного с самыми быстрыми предыдущими методами [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022]. Ключом к этой производительности является новое представление сцены с помощью 3D Гауссиан в сочетании с дифференцируемым рендерером в реальном времени, который обеспечивает значительное ускорение как оптимизации сцены, так и синтеза новых видов. Обратите внимание, что для сравнимого времени обучения с InstantNGP [Müller et al. 2022] мы достигаем качество, аналогичное их; хотя это максимальное качество, которое они достигают, обучаясь 51 минуту, мы достигаем передового качества, даже немного лучше, чем Mip-NeRF360 [Barron et al. 2022].

Методы радиусных полей недавно произвели революцию в синтезе новых видов сцен, захваченных с помощью множества фотографий или видео. Однако достижение высокого визуального качества по-прежнему требует нейронных сетей, которые дороги в обучении и рендеринге, а недавние более быстрые методы неизбежно жертвуют скоростью ради качества. Для неограниченных и полных сцен (в отличие от изолированных объектов) и рендеринга с разрешением 1080р ни один из текущих методов не может достичь частоты вывода изображений в реальном времени. Мы представляем три ключевых элемента, которые позволяют нам достичь передового визуального качества, сохранив при этом конкурентные времена обучения и, что важно, обеспечивают высококачественный синтез новых видов в реальном времени ( $\geq 30$  fps) с разрешением 1080р. Во-первых, начиная со скучных точек, созданных во время калибровки камеры, мы представляем сцену с помощью 3D Гауссиан, сохранив желаемые свойства непрерывных объемных радиусных полей для оптимизации сцены, избегая

\*Оба автора внесли равный вклад в статью.

Authors' Contact Information: Бернхард Кербл, bernhard.kerbl@inria.fr, Inria, Université Côte d’Azur, Франция; Георгиос Копанас, georgios.kopanas@inria.fr, Inria, Université Côte d’Azur, Франция; Томас Леймкюлер, thomas.leimkuehler@mpi-inf.mpg.de, Max-Planck-Institut für Informatik, Германия; Джордж Дреттакис, george.drettakis@inria.fr, Inria, Université Côte d’Azur, Франция.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Request permissions from owner/author(s).

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM 1557-7368/2023/8-ART1  
https://doi.org/10.1145/3592433

при этом ненужных вычислений в пустом пространстве; Во-вторых, мы выполняем чередующуюся оптимизацию/контроль плотности 3D Гауссиан, особенно оптимизируя анизотропную ковариацию для достижения точного представления сцены; В-третьих, мы разрабатываем быстрый алгоритм рендеринга с учетом видимости, который поддерживает анизотропное сплэттинг и ускоряет как обучение, так и позволяет рендеринг в реальном времени. Мы демонстрируем передовое визуальное качество и рендеринг в реальном времени на нескольких установленных наборах данных.

CCS Concepts: • Вычислительные методологии → Рендеринг; Модели на основе точек; РаSTERизация; Подходы машинного обучения.

Additional Key Words and Phrases: синтез новых видов, радиусные поля, 3D гауссианы, рендеринг в реальном времени

ACM Reference Format:

Бернхард Кербл, Георгиос Копанас, Томас Леймкюлер, and Джордж Дреттакис. 2023. 3D Gaussian Splatting для рендеринга радиусных полей в реальном времени. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 15 pages. <https://doi.org/10.1145/3592433>

## 1 Введение

Сетки и точки являются наиболее распространенными представлениями 3D-сцен, поскольку они явные и хорошо подходят для быстрой раSTERизации на базе GPU/CUDA. В отличие от них, современные методы Нейронных Радиусных Полей (NeRF) основываются на непрерывных представлениях сцен, обычно оптимизируя многослойный перцепtron (MLP) с использованием объемного трассирования лучей для синтеза

новых видов захваченных сцен. Аналогично, самые эффективные на данный момент решения радиусных полей строятся на непрерывных представлениях, интерполируя значения, хранящиеся, например, в воксельных [Fridovich-Keil and Yu et al. 2022] или хеш-сетках [Müller et al. 2022], или точках [Xu et al. 2022]. Хотя непрерывная природа этих методов способствует оптимизации, стохастическая выборка, необходимая для рендеринга, является дорогостоящей и может привести к шуму. Мы представляем новый подход, который сочетает в себе лучшее из обоих миров: наши 3D Гауссианы позволяют проводить оптимизацию с качеством визуализации на уровне современного уровня (SOTA) и конкурентным временем обучения, в то время как наше решение на основе тайлового сплайнинга обеспечивает рендеринг в реальном времени с SOTA качеством для разрешения 1080р на нескольких ранее опубликованных наборах данных [Barron et al. 2022; Hedman et al. 2018; Knapisch et al. 2017] (см. Рис. 1).

Наша цель — позволить рендеринг в реальном времени для сцен, захваченных с помощью множества фотографий, и создать представления с временем оптимизации, таким же быстрым, как у самых эффективных предыдущих методов для типичных реальных сцен. Недавние методы достигают быстрого обучения [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022], но испытывают трудности в достижении качества визуализации, получаемого текущими SOTA NeRF методами, т. е., Mip-NeRF360 [Barron et al. 2022], который требует до 48 часов времени обучения. Быстрые, но менее качественные методы радиусных полей могут достигать интерактивного времени рендеринга в зависимости от сцены (10–15 кадров в секунду), но не достигают рендеринга в реальном времени при высоком разрешении.

Наше решение строится на трех основных компонентах. Мы сначала вводим 3D Гауссианы как гибкое и выразительное представление сцены. Мы начинаем с того же входного сигнала, что и предыдущие методы, похожие на NeRF, т. е., камеры, откалиброванные с помощью Structure-from-Motion (SfM) [Snavely et al. 2006], и инициализируем набор 3D Гауссиан с использованием разреженного облака точек, которое создается бесплатно как часть процесса SfM. В отличие от большинства решений на основе точек, которые требуют данных Многовидового Стерео (MVS) [Aliev et al. 2020; Kopanas et al. 2021; Rückert et al. 2022], мы достигаем высококачественных результатов только с использованием точек SfM на входе. Отметим, что для NeRF-синтетического набора данных наш метод достигает высокого качества даже при случайной инициализации. Мы показываем, что 3D Гауссианы являются отличным выбором, поскольку это дифференцируемое объемное представление, но их также можно очень эффективно растеризовать, проецируя их в 2D и применяя стандартное  $\alpha$ -смешивание, используя ту же модель формирования изображения, что и NeRF. Второй компонент нашего метода заключается в оптимизации свойств 3D Гауссиан — 3D позиции, непрозрачности  $\alpha$ , анизотропной ковариации и коэффициентов сферических гармоник (SH) — чередующихся с шагами адаптивного контроля плотности, где мы добавляем и иногда удаляем 3D Гауссианы во время оптимизации. Процедура оптимизации создает достаточно

компактное, неструктурированное и точное представление сцены (1–5 миллионов Гауссиан для всех протестированных сцен). Третий и последний элемент нашего метода — это наше решение для рендеринга в реальном времени, которое использует быстрые алгоритмы сортировки GPU и вдохновлено тайловой растеризацией, следуя недавним работам [Lassner and Zollhofer 2021]. Однако благодаря нашему представлению 3D Гауссиан мы можем выполнять анизотропное сплайнинге, учитывая порядок видимости — благодаря сортировке и  $\alpha$ -смешиванию — и обеспечивать быстрый и точный обратный проход, отслеживая прохождение стольких отсортированных сплотов, сколько необходимо.

Подводя итог, мы предоставляем следующие вклады:

- Введение анизотропных 3D Гауссиан как высококачественного, неструктурированного представления радиусных полей.
- Метод оптимизации свойств 3D Гауссиан, чередующийся с адаптивным контролем плотности, создающий высококачественные представления для захваченных сцен.

- Быстрый, дифференцируемый подход к рендерингу для GPU, учитывающий видимость, позволяющий анизотропное сплайнинге и быстрое обратное распространение для достижения высококачественного синтеза новых видов.  
Наши результаты на ранее опубликованных наборах данных показывают, что мы можем оптимизировать наши 3D Гауссианы из многовидовых захватов и достичь качества, равного или лучшего, чем у лучших предыдущих неявных методов радиусных полей. Мы также можем достичь скоростей обучения и качества, аналогичных самым быстрым методам, и, что важно, представить первый *рендеринг в реальном времени* с высоким качеством для синтеза новых видов.

## 2 Связанные работы

Мы сначала кратко обозреваем традиционную реконструкцию, затем обсуждаем методы рендеринга на основе точек и радиусных полей, рассматривая их сходство; радиусные поля — это обширная область, поэтому мы фокусируемся только на непосредственно связанных работах. Для полного охвата области, пожалуйста, см. отличные недавние обзоры [Tewari et al. 2022; Xie et al. 2022].

### 2.1 Традиционная реконструкция сцен и рендеринг

Первые подходы к синтезу новых видов были основаны на световых полях, сначала плотно семплированных [Gortler et al. 1996; Levoy and Hanrahan 1996], а затем позволяющих неструктурированный захват [Buehler et al. 2001]. Появление метода Structure-from-Motion (SfM) [Snavely et al. 2006] открыло новую область, где коллекция фотографий могла использоваться для синтеза новых видов. SfM оценивает разреженное облако точек во время калибровки камеры, которое изначально использовалось для простой визуализации 3D пространства. Последующие многовидовые стерео методы (MVS) привели к появлению впечатляющих алгоритмов полной 3D реконструкции за прошедшие годы [Goesele et al. 2007], что позволило разработать несколько алгоритмов синтеза видов [Chaurasia et al. 2013; Eisemann et al. 2008; Hedman et al. 2018; Kopanas

et al. 2021]. Все эти методы *репроектируют и смешивают* входные изображения в камеру нового ракурса, используя геометрию для управления этой репроекцией. Эти методы давали отличные результаты во многих случаях, но обычно не могут полностью восстановиться после нереконструированных областей или из-за «чрезмерной реконструкции», когда MVS создает несуществующую геометрию. Недавние алгоритмы нейронного рендеринга [Tewari et al. 2022] значительно уменьшают такие артефакты и избегают огромной стоимости хранения всех входных изображений на GPU, превосходя эти методы по большинству параметров.

## 2.2 Нейронный рендеринг и радиусные поля

Глубокие методы обучения были внедрены на ранних этапах для синтеза новых видов [Flynn et al. 2016; Zhou et al. 2016]; сверточные нейронные сети (CNN) использовались для оценки весов смешивания [Hedman et al. 2018], или для решений в текстурном пространстве [Riegler and Koltun 2020; Thies et al. 2019]. Использование геометрии, основанной на MVS, является основным недостатком большинства этих методов; кроме того, использование CNN для окончательного рендеринга часто приводит к временному мерцанию. Объемные представления для синтеза новых видов были инициированы Soft3D [Penner and Zhang 2017]; глубокие методы обучения, связанные с объемным трассированием лучей, были предложены позднее [Henzler et al. 2019; Sitzmann et al. 2019], основываясь на непрерывном дифференцируемом поле плотности для представления геометрии. Рендеринг с использованием объемного трассирования лучей имеет значительную стоимость из-за большого количества выборок, необходимых для запроса объема. Нейронные радиусные поля (NeRFs) [Mildenhall et al. 2020] ввели важную выборку и позиционное кодирование для повышения качества, но использовали большой многослойный перцептрон, что негативно влияло на скорость. Успех NeRF привел к взрыву последующих методов, которые решают проблемы качества и скорости, часто вводя стратегии регуляризации; текущее состояние техники в качестве изображения для синтеза новых видов — это Mip-NeRF360 [Barron et al. 2022]. Хотя качество рендеринга выдающееся, время обучения и рендеринга остается чрезвычайно высоким; мы можем соответствовать или в некоторых случаях превзойти это качество, обеспечивая быстрое обучение и рендеринг в реальном времени. Наиболее современные методы сосредоточились на более быстром обучении и/или рендеринге, главным образом за счет трех проектных решений: использования пространственных структур данных для хранения (нейронных) признаков, которые затем интерполируются во время объемного трассирования лучей, различных кодировок и емкости MLP. Такие методы включают различные варианты дискретизации пространства [Chen et al. 2022b,a; Fridovich-Keil and Yu et al. 2022; Garbin et al. 2021; Hedman et al. 2021; Reiser et al. 2021; Takikawa et al. 2021; Wu et al. 2022; Yu et al. 2021], кодовые книги [Takikawa et al. 2022], и кодировки, такие как хеш-таблицы [Müller et al. 2022], что позволяет использовать меньший MLP или полностью отказаться от нейронных сетей

[Fridovich-Keil and Yu et al. 2022; Sun et al. 2022]. Наиболее примечательными из этих методов являются InstantNGP [Müller et al. 2022], который использует хеш-сетку и сетку занятости для ускорения вычислений и меньший MLP для представления плотности и внешнего вида; и Plenoxels [Fridovich-Keil and Yu et al. 2022], которые используют разреженную воксельную сетку для интерполяции непрерывного поля плотности и могут полностью обойтись без нейронных сетей. Оба они полагаются на сферические гармоники: первый для прямого представления направленных эффектов, второй для кодирования своих входных данных в цветовую сеть. Хотя оба предоставляют выдающиеся результаты, эти методы все еще могут испытывать трудности с эффективным представлением пустого пространства, отчасти зависящие от типа сцены/захвата. Кроме того, качество изображения в значительной степени ограничено выбором структурированных сеток, используемых для ускорения, а скорость рендеринга ограничивается необходимостью запрашивать множество выборок для данного шага трассировки лучей. Неупорядоченные, явные, дружественные к GPU 3D Гауссианы, которые мы используем, обеспечивают более быструю скорость рендеринга и лучшее качество *без* нейронных компонентов.

## 2.3 Рендеринг на основе точек и радиусные поля

Методы на основе точек эффективно рендерят отсоединенные и неструктурные геометрические образцы (например, облака точек) [Gross and Pfister 2011]. В самой простой форме рендеринг точек выборки [Grossman and Dally 1998] растраивает неструктурный набор точек с фиксированным размером, для которого может быть использована нативная поддержка типов точек графических API [Sainz and Pajarola 2004] или параллельная программная растеризация на GPU [Laine and Karras 2011; Schütz et al. 2022]. Будучи верным исходным данным, рендеринг точек выборки страдает от дыр, вызывает алиасинг и является строго разрывным. Семинальная работа по качественному рендерингу на основе точек решает эти проблемы путем "сплэттинга" точечных примитивов с протяженностью больше пикселя, например, круглых или эллиптических дисков, эллипсоидов или сурфелей [Botsch et al. 2005; Pfister et al. 2000; Ren et al. 2002; Zwicker et al. 2001b]. Появился недавний интерес к *дифференцируемым* методам рендеринга на основе точек [Wiles et al. 2020; Yifan et al. 2019]. Точки были дополнены нейронными признаками и отрендерены с использованием CNN [Aliev et al. 2020; Rückert et al. 2022], что привело к быстрому или даже реальному времени синтезу видов; однако они все еще зависят от MVS для начальной геометрии, и таким образом наследуют его артефакты, особенно пере- или недо-реконструкцию в сложных случаях, таких как бесособые/блестящие области или тонкие структуры. Альфа-смешивание на основе точек и стильный рендеринг NeRF имеют, по сути, одну и ту же модель формирования изображения. В частности, цвет  $C$  задается объемным рендерингом вдоль луча:

$$C = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) c_i \quad \text{with} \quad T_i = \exp \left( - \sum_{j=1}^{i-1} \sigma_j \delta_j \right), \quad (1)$$

где выборки плотности  $\sigma$ , пропускания  $T$  и цвета с берутся вдоль луча с интервалами  $\delta_i$ . Это можно переписать как

$$C = \sum_{i=1}^N T_i \alpha_i \mathbf{c}_i, \quad (2)$$

с

$$\alpha_i = (1 - \exp(-\sigma_i \delta_i)) \text{ and } T_i = \prod_{j=1}^{i-1} (1 - \alpha_j).$$

Типичный нейронный подход на основе точек (например, [Koranas et al. 2022, 2021]) вычисляет цвет  $C$  пикселя путем смешивания  $N$  упорядоченных точек, перекрывающих пиксель:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (3)$$

где  $\mathbf{c}_i$  — это цвет каждой точки и  $\alpha_i$  задается путем оценки 2D Гауссиана с ковариацией  $\Sigma$  [Yifan et al. 2019], умноженного на обучаемую по-точечную непрозрачность. Из уравнений 2 и 3 мы можем ясно видеть, что модель формирования изображения одна и та же. Однако алгоритм рендеринга очень различен. NeRF представляет собой непрерывное представление, неявно представляющее пустое/занятое пространство; требуется дорогая случайная выборка для нахождения выборок в уравнении 2 с последующим шумом и вычислительными затратами. Напротив, точки представляют собой неструктурированное, дискретное представление, достаточно гибкое для создания, уничтожения и перемещения геометрии, аналогично NeRF. Это достигается путем оптимизации непрозрачности и положений, как показано в предыдущих работах [Koranas et al. 2021], избегая недостатков полноценного объемного представления. Pulsar [Lassner and Zollhofer 2021] достигает быстрой сферической растеризации, которая вдохновила нашу тайловую систему и алгоритм сортировки рендеринга. Однако, учитывая анализ выше, мы хотим сохранять (приближенное) обычное  $\alpha$ -смешивание на отсортированных сплатах для получения преимуществ объемных представлений: наша растеризация учитывает порядок видимости в отличие от их метода, не зависящего от порядка. Кроме того, мы обратно распространяем градиенты на все сплаты в пикселе и растеризуем анизотропные сплаты. Все эти элементы способствуют высокому визуальному качеству наших результатов (см. раздел 7.3). Кроме того, упомянутые выше предыдущие методы также используют CNN для рендеринга, что приводит к временной нестабильности. Тем не менее, скорость рендеринга Pulsar [Lassner and Zollhofer 2021] и ADOP [Rückert et al. 2022] послужила мотивацией для разработки нашего быстрого решения рендеринга. Хотя фокусируясь на зеркальных эффектах, диффузный трек рендеринга на основе точек Neural Point Catacaustics [Koranas et al. 2022] преодолевает эту временную нестабильность с помощью MLP, но все же требует геометрию MVS на входе. Самый последний метод [Zhang et al. 2022] в этой категории не требует MVS и также использует SH для направлений; однако он может обрабатывать только сцены одного объекта и требует маски для инициализации. Хотя он быстр для малых разрешений и небольшого количества точек, неясно, как он сможет масштабироваться до сцен типичных

наборов данных [Barron et al. 2022; Hedman et al. 2018; Knapitsch et al. 2017]. Мы используем 3D Гауссианы для более гибкого представления сцены, избегая необходимости в геометрии MVS и достигая рендеринга в реальном времени благодаря нашему тайловому алгоритму рендеринга для спроектированных Гауссиан. Недавний подход [Xu et al. 2022] использует точки для представления радиусного поля с помощью подхода функции радиального базиса. Они применяют методы прореживания и уплотнения точек во время оптимизации, но используют объемное трассирование лучей и не могут достичь скоростей реального времени. В области захвата человеческих действий 3D Гауссианы использовались для представления захваченных человеческих тел [Rhodin et al. 2015; Stoll et al. 2011]; совсем недавно они использовались с объемным трассированием лучей для задач зрения [Wang et al. 2023]. Нейронные объемные примитивы были предложены в подобном контексте [Lombardi et al. 2021]. Хотя эти методы вдохновили выбор 3D Гауссиан в качестве нашего представления сцены, они сосредотачиваются на конкретном случае реконструкции и рендеринга одного изолированного объекта (тела человека или лица), что приводит к сценам с малой глубиной сложности. В противоположность этому, наша оптимизация анизотропной ковариации, чередующаяся оптимизация/контроль плотности и эффективная сортировка по глубине для рендеринга позволяют нам обрабатывать полные, сложные сцены, включая фон, как внутри помещений, так и на открытом воздухе, и с большой глубиной сложности.

### 3 Обзор

Входными данными для нашего метода являются изображения статической сцены вместе с соответствующими камерами, откалиброванными с помощью SfM [Schönberger and Frahm 2016], который в качестве побочного эффекта создает разреженное облако точек. Из этих точек мы создаем набор 3D Гауссиан (Раздел 4), определяемых позицией (средним значением), матрицей ковариации и непрозрачностью  $\alpha$ , что позволяет использовать очень гибкий режим оптимизации. В результате получается достаточно компактное представление 3D сцены, отчасти потому, что высокоанизотропные объемные сплаты могут компактно представлять мелкие структуры. Направленная составляющая внешнего вида (цвет) радиусного поля представляется с помощью сферических гармоник (SH), что является стандартной практикой [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022]. Наш алгоритм переходит к созданию представления радиусного поля (Раздел 5) через последовательность шагов оптимизации параметров 3D Гауссиан, то есть положения, ковариации,  $\alpha$  и коэффициентов SH, чередующихся с операциями для адаптивного контроля плотности Гауссиан. Ключом к эффективности нашего метода является наш растеризатор на основе тайлов (Раздел 6), который позволяет выполнять  $\alpha$ -смешивание анизотропных сплатов, учитывая порядок видимости благодаря быстрой сортировке. Наш быстрый растеризатор также включает быстрый обратный проход, отслеживая накопленные значения  $\alpha$ , без

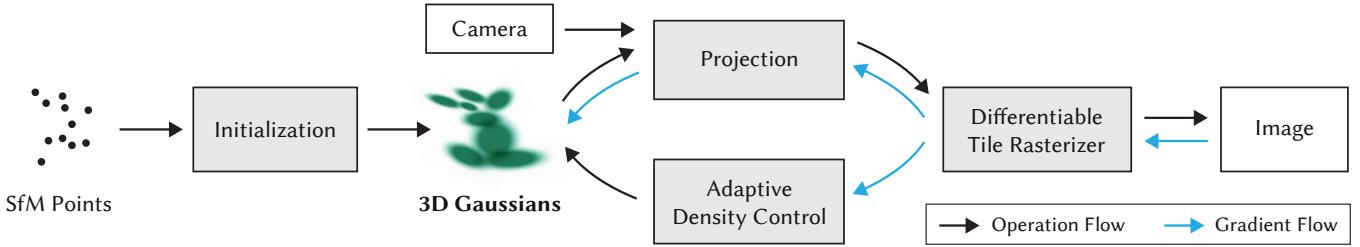


Fig. 2. Оптимизация начинается с разреженного облака точек SfM и создает набор 3D Гауссиан. Затем мы оптимизируем и адаптивно контролируем плотность этого набора Гауссиан. Во время оптимизации мы используем наш быстрый растеризатор на основе тайлов, что позволяет достичь конкурентного времени обучения по сравнению с самыми быстрыми современными методами радиусных полей. После обучения наш рендерер обеспечивает навигацию в реальном времени для широкого спектра сцен.

ограничения на количество Гауссиан, которые могут получать градиенты. Обзор нашего метода представлен на Рис. 2.

#### 4 Дифференцируемое 3D-сплэттинг Гауссиан

Наша цель – оптимизировать представление сцены, которое позволяет синтез высококачественных новых видов, начиная с разреженного набора (SfM) точек без нормалей. Для этого нам нужен примитив, который наследует свойства дифференцируемых объемных представлений, при этом оставаясь неструктурированным и явным для обеспечения очень быстрого рендеринга. Мы выбираем 3D Гауссианы, которые являются дифференцируемыми и могут быть легко спроектированы в 2D-сплаты, что позволяет быстро выполнять  $\alpha$ -смешивание для рендеринга.

Наше представление имеет сходства с предыдущими методами, использующими 2D-точки [Коранас et al. 2021; Yifan et al. 2019], и предполагает, что каждая точка является маленьким плоским кругом с нормалью. Учитывая крайнюю разреженность точек SfM, оценка нормалей крайне затруднена. Аналогично, оптимизация очень зашумленных нормалей из такой оценки также была бы очень сложной задачей. Вместо этого мы моделируем геометрию как набор 3D Гауссиан, которые не требуют нормалей. Наши Гауссианы определяются полной 3D матрицей ковариации  $\Sigma$ , заданной в мировом пространстве [Zwicker et al. 2001a] с центром в точке (среднее значение)  $\mu$ :

$$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)} \quad (4)$$

. Этот Гауссиан умножается на  $\alpha$  в нашем процессе смешивания.

Однако нам нужно спроектировать наши 3D Гауссианы в 2D для рендеринга. Цвикер и др. [2001a] демонстрируют, как выполнить эту проекцию в пространство изображения. Учитывая преобразование просмотра  $W$ , матрица ковариации  $\Sigma'$  в координатах камеры выглядит следующим образом:

$$\Sigma' = JW \Sigma W^T J^T \quad (5)$$

где  $J$  – это якобиан аффинного приближения проективного преобразования. Цвикер и др. [2001a] также показывают, что если мы пропустим третью строку и столбец  $\Sigma'$ , мы получим матрицу дисперсии размером  $2 \times 2$  с той же структурой и

свойствами, как если бы мы начинали с плоских точек с нормалью, как в предыдущих работах [Коранас et al. 2021].

Очевидный подход заключался бы в непосредственной оптимизации матрицы ковариации  $\Sigma$  для получения 3D Гауссиан, представляющих радиусное поле. Однако матрицы ковариации имеют физический смысл только тогда, когда они являются положительно полуопределенными. Для нашей оптимизации всех параметров мы используем градиентный спуск, который сложно ограничить производством таких допустимых матриц, и шаги обновления и градиенты могут очень легко создавать недопустимые матрицы ковариации.

В результате мы выбрали более интуитивное, но эквивалентное выражение для оптимизации. Матрица ковариации 3D Гауссиана аналогична описанию конфигурации эллипсоида. Учитывая матрицу масштабирования  $S$  и матрицу поворота  $R$ , мы можем найти соответствующую  $\Sigma$ :

$$\Sigma = RSS^T R^T \quad (6)$$

Чтобы позволить независимую оптимизацию обоих факторов, мы храним их отдельно: 3D вектор  $s$  для масштабирования и кватернион  $q$  для представления вращения. Их можно легко преобразовать в соответствующие матрицы и объединить, убедившись, что  $q$  нормализован для получения единичного кватерниона.

Чтобы избежать значительных накладных расходов из-за автоматического дифференцирования во время обучения, мы явно выводим градиенты для всех параметров. Подробности точных вычислений производных находятся в приложении А.

Это представление анизотропной ковариации – подходящее для оптимизации – позволяет нам оптимизировать 3D Гауссианы для адаптации к геометрии различных форм в захваченных сценах, что приводит к довольно компактному представлению. Рис. 3 иллюстрирует такие случаи.

#### 5 Оптимизация с адаптивным контролем плотности 3D Гауссиан

Основой нашего подхода является шаг оптимизации, который создает плотный набор 3D Гауссиан, точно представляющих сцену для синтеза произвольных видов. В дополнение к позициям  $p$ ,  $\alpha$  и ковариации  $\Sigma$ , мы также оптимизируем коэффициенты SH, представляющие цвет с каждой Гауссианы,



Fig. 3. Мы визуализируем 3D Гауссианы после оптимизации, сжимая их на 60% (справа). Это четко показывает анизотропные формы 3D Гауссиан, которые компактно представляют сложную геометрию после оптимизации. Слева — фактическое отрендеренное изображение.

чтобы правильно захватывать зависимый от вида внешний вид сцены. Оптимизация этих параметров чередуется с шагами, контролирующими плотность Гауссиан для лучшего представления сцены.

### 5.1 Оптимизация

Оптимизация основана на последовательных итерациях рендеринга и сравнения полученного изображения с обучающими видами из захваченного набора данных. Неизбежно, геометрия может быть неправильно размещена из-за неоднозначностей проекции 3D в 2D. Таким образом, наша оптимизация должна уметь создавать геометрию, а также уничтожать или перемещать геометрию, если она была неправильно расположена. Качество параметров ковариаций 3D Гауссиан критично для компактности представления, поскольку большие однородные области могут быть захвачены небольшим числом крупных анизотропных Гауссиан.

Мы используем методы стохастического градиентного спуска для оптимизации, полностью используя стандартные GPU-ускоренные фреймворки, и возможность добавлять пользовательские CUDA-ядра для некоторых операций, следуя современным рекомендациям [Fridovich-Keil and Yu et al. 2022; Sun et al. 2022]. В частности, наш быстрый растеризатор (см. Раздел 6) критичен для эффективности нашей оптимизации, поскольку это основное вычислительное узкое место оптимизации.

Мы используем сигмоидальную функцию активации для  $\alpha$ , чтобы ограничить её диапазоном  $[0 - 1]$  и получить гладкие градиенты, и экспоненциальную функцию активации для масштаба ковариации по аналогичным причинам.

Мы оцениваем начальную ковариационную матрицу как изотропную Гауссиану с осьми, равными среднему расстоянию до ближайших трех точек. Мы используем стандартную технику экспоненциального затухания, аналогичную Plenoxels [Fridovich-Keil and Yu et al. 2022], но только для позиций. Функция потерь представляет собой комбинацию  $\mathcal{L}_1$  и терма D-SSIM:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} \quad (7)$$

Мы используем  $\lambda = 0.2$  во всех наших тестах. Подробнее о расписании обучения и других элементах представлено в Разделе 7.1.

### 5.2 Адаптивный контроль Гауссиан

Мы начинаем с начального набора разреженных точек из SfM, а затем применяем наш метод для адаптивного контроля числа Гауссиан и их плотности на единицу объема<sup>1</sup>, что позволяет нам перейти от начального разреженного набора Гауссиан к более плотному набору, который лучше представляет сцену и имеет правильные параметры. После прогрева оптимизации (см. Раздел 7.1), мы уплотняем каждые 100 итераций и удаляем любые Гауссианы, которые практически прозрачны, то есть с  $\alpha$  меньше порога  $\epsilon_\alpha$ .

Наш адаптивный контроль Гауссиан должен заполнять пустые области. Он фокусируется на областях с отсутствующими геометрическими особенностями («недостаточная реконструкция»), но также и в областях, где Гауссианы покрывают большие области сцены (что часто соответствует «чрезмерной реконструкции»). Мы наблюдаем, что у обоих случаев есть *большие* градиенты положения в пространстве видов. Интуитивно это объясняется тем, что они соответствуют областям, которые еще не хорошо реконструированы, и оптимизация пытается переместить Гауссианы для исправления этого.

Поскольку оба случая являются хорошими кандидатами для уплотнения, мы уплотняем Гауссианы со средней величиной градиента положения в пространстве видов выше порога  $\tau_{\text{pos}}$ , который мы установили равным 0.0002 в наших тестах.

Далее мы представляем подробности этого процесса, проиллюстрированные на Рис. 4.

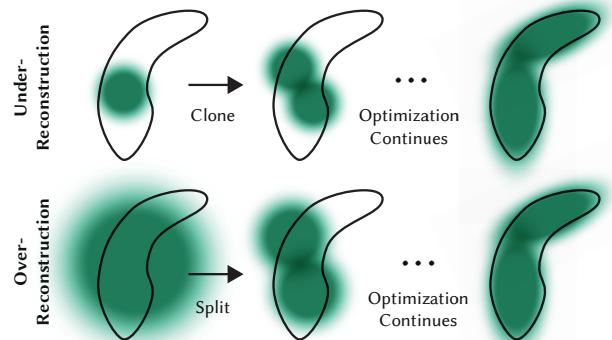


Fig. 4. Наша адаптивная схема уплотнения Гауссиан. Верхний ряд (недостаточная реконструкция): Когда мелкомасштабная геометрия (черный контур) недостаточно покрыта, мы клонируем соответствующую Гауссиану. Нижний ряд (чрезмерная реконструкция): Если мелкомасштабная геометрия представлена одним большим сплатом, мы разделяем его на два.

Для маленьких Гауссиан в недостаточно реконструированных регионах нам нужно покрыть новую геометрию, которая

<sup>1</sup>Плотность Гауссиан не следует путать, конечно же, с плотностью  $\sigma$  в литературе NeRF.

должна быть создана. Для этого предпочтительно клонировать Гауссианы, просто создавая копию того же размера и перемещая её в направлении градиента положения.

С другой стороны, крупные Гауссианы в регионах с высокой дисперсией должны быть разделены на меньшие Гауссианы. Мы заменяем такие Гауссианы двумя новыми, и делим их масштаб на множитель  $\phi = 1.6$ , который мы определили экспериментально. Мы также инициализируем их положение, используя исходную 3D Гауссиану как PDF для выборки.

В первом случае мы обнаруживаем и лечим необходимость увеличения как общего объема системы, так и количества Гауссиан, тогда как во втором случае мы сохраняем общий объем, но увеличиваем количество Гауссиан. Подобно другим объемным представлениям, наша оптимизация может застрять с "плавающими" объектами близко к входным камерам; в нашем случае это может привести к необоснованному увеличению плотности Гауссиан. Эффективный способ умерить увеличение числа Гауссиан – установить значение  $\alpha$  близкое к нулю каждые  $N = 3000$  итераций. Затем оптимизация увеличивает  $\alpha$  для Гауссиан, где это необходимо, позволяя нашему подходу удаления исключать Гауссианы с  $\alpha$  меньше  $\epsilon_\alpha$ , как описано выше. Гауссианы могут сжиматься или расширяться и значительно пересекаться с другими, но мы периодически удаляем Гауссианы, которые являются очень большими в мировом пространстве и те, что имеют большой след в пространстве видов. Эта стратегия обеспечивает хорошее общее управление общим числом Гауссиан. Гауссианы в нашей модели остаются примитивами в евклидовом пространстве в любое время; в отличие от других методов [Bargon et al. 2022; Fridovich-Keil and Yu et al. 2022], нам не требуются стратегии уплотнения пространства, деформации или проекции для дальних или больших Гауссиан.

## 6 Быстрый дифференцируемый растеризатор для Гауссиан

Наши цели заключаются в обеспечении быстрого общего рендеринга и быстрой сортировки, чтобы позволить приближенное  $\alpha$ -смешивание – включая для анизотропных сплатов – и избежать жестких ограничений на количество сплатов, которые могут получать градиенты, как это существует в предыдущих работах [Lassner and Zollhofer 2021].

Для достижения этих целей мы разработали тайловый растеризатор для Гауссиан, вдохновленный недавними подходами программного растеризатора [Lassner and Zollhofer 2021], который позволяет предварительно сортировать примитивы для всего изображения сразу, избегая затрат на сортировку для каждого пикселя, что замедляло предыдущие решения с  $\alpha$ -смешиванием [Koranas et al. 2022, 2021]. Наш быстрый растеризатор позволяет эффективно выполнять обратное распространение ошибки по произвольному числу смешанных Гауссиан с низким дополнительным потреблением памяти, требуя только постоянные накладные расходы на пиксель. Наш конвейер растеризации полностью дифференцируем, и после проекции в 2D (Раздел 4) он может растеризовать

анизотропные сплаты, аналогично предыдущим методам 2D сплэттинга [Koranas et al. 2021].

Наш метод начинается с разделения экрана на тайлы размером  $16 \times 16$ , а затем переходит к отсечению 3D Гауссиан относительно пирамиды видимости и каждого тайла. В частности, мы сохраняем только те Гауссианы, у которых интервал достоверности 99% пересекает пирамиду видимости. Кроме того, мы используем защитную полосу для тривиального отбраковывания Гауссиан в крайних положениях (т.е., тех, чьи средние значения близки к ближней плоскости и сильно выходят за пределы пирамиды зрения), так как вычисление их проекции в 2D ковариацию было бы нестабильным. Затем мы создаем экземпляры каждой Гауссианы в соответствии с количеством перекрывающихся тайлов и присваиваем каждому экземпляру ключ, объединяющий глубину в пространстве просмотра и ID тайла. После этого мы сортируем Гауссианы на основе этих ключей, используя быструю GPU сортировку Radix [Merrill and Grimshaw 2010]. Обратите внимание, что здесь нет дополнительной сортировки точек на каждый пиксель, и смешивание выполняется на основе этой первоначальной сортировки. В результате, наше  $\alpha$ -смешивание может быть приближенным в некоторых конфигурациях. Однако эти приближения становятся незначительными, когда сплаты достигают размера отдельных пикселей. Мы обнаружили, что этот выбор значительно повышает производительность обучения и рендеринга без создания заметных артефактов в сценах после сходимости.

После сортировки Гауссиан мы формируем список для каждого тайла, определяя первый и последний элемент, отсортированный по глубине, который влияет на данный тайл. Для растеризации мы запускаем один блок потоков для каждого тайла. Каждый блок сначала совместно загружает пакеты Гауссиан в разделяемую память, а затем, для данного пикселя, накапливает значения цвета и  $\alpha$ , проходя списки спереди назад, тем самым максимально увеличивая параллелизм как для загрузки/разделения данных, так и для их обработки. Когда достигается целевое значение насыщения  $\alpha$  в пикселе, соответствующий поток останавливается. С регулярными интервалами потоки в тайле проверяются, и обработка всего тайла завершается, когда все пиксели насыщаются (т.е.,  $\alpha$  становится равным 1). Детали сортировки и высокоуровневый обзор всего процесса растеризации представлены в Приложении C.

Во время растеризации насыщение  $\alpha$  является единственным критерием остановки. В отличие от предыдущих работ, мы не ограничиваем количество примитивов, участвующих в смешивании, которые получают обновления градиента. Мы применяем это свойство, чтобы позволить нашему подходу обрабатывать сцены с произвольной и изменяющейся глубиной сложности и точно обучать их, не прибегая к настройке гиперпараметров, специфичных для сцены. Во время обратного прохода нам необходимо восстановить полную последовательность смешанных точек для каждого пикселя в прямом проходе. Одним из решений могло бы быть хранение произвольно длинных списков смешанных точек для каждого пикселя в глобальной памяти [Koranas et al. 2021]. Чтобы избежать накладных расходов на динамическое управление памятью, мы выбираем повторное прохождение списков для

каждого тайла; мы можем повторно использовать отсортированный массив Гауссиан и диапазоны тайлов из прямого прохода. Чтобы облегчить вычисление градиентов, мы теперь проходим их сзади наперёд.

Прохождение начинается с последней точки, которая оказалась влияние на любой пиксель в тайле, и загрузка точек в разделяемую память снова происходит совместно. Кроме того, каждый пиксель будет проводить (дорогостоящее) тестирование наложения и обработку точек только если их глубина меньше или равна глубине последней точки, которая внесла вклад в его цвет во время прямого прохода. Вычисление градиентов, описанных в Разделе 4, требует накопленных значений непрозрачности на каждом шаге во время исходного процесса смешивания. Вместо обхода явного списка постепенно уменьшающихся значений непрозрачности в обратном проходе, мы можем восстановить эти промежуточные значения непрозрачности, сохраняя только общую накопленную непрозрачность в конце прямого прохода. В частности, каждая точка сохраняет окончательное накопленное значение непрозрачности  $\alpha$  в прямом процессе; мы делим это на  $\alpha$  каждой точки в нашем обратном проходе, чтобы получить необходимые коэффициенты для вычисления градиентов.

## 7 Обсуждение результатов и оценка

Мы обсудим некоторые детали реализации, представим результаты и проведем оценку нашего алгоритма в сравнении с предыдущими работами, а также проведем исследование по удалению компонентов.

### 7.1 Реализация

Мы реализовали наш метод на Python с использованием фреймворка PyTorch и написали пользовательские CUDA-ядра для растеризации, которые являются расширенными версиями предыдущих методов [Kopanas et al. 2021], и используем библиотеку NVIDIA CUB для быстрой сортировки Radix sort [Merrill and Grimshaw 2010]. Также мы создали интерактивный просмотрщик, используя открытый исходный код SIBR [Bonomo et al. 2020], который используется для интерактивного просмотра. Мы использовали эту реализацию для измерения достигнутой частоты кадров. Исходный код и все наши данные доступны по адресу: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>

*Детали оптимизации.* Для стабильности мы «разогреваем» вычисления в более низком разрешении. В частности, мы начинаем оптимизацию, используя изображения с разрешением в 4 раза ниже и увеличиваем их дважды после 250 и 500 итераций. Оптимизация коэффициентов SH чувствительна к недостатку угловой информации. Для типичных «NeRF-like» захватов, где центральный объект наблюдается с помощью фотографий, сделанных по всему полушарию вокруг него, оптимизация работает хорошо. Однако, если захват имеет пропущенные угловые области (например, при захвате угла сцены или выполнении «внутрь-наружу» [Hedman et al. 2016] захвата), полностью некорректные значения для нулевого компонента SH (т.е., базового или диффузного цвета) могут быть

получены в результате оптимизации. Чтобы преодолеть эту проблему, мы начинаем с оптимизации только нулевого компонента, а затем вводим одну полосу SH после каждой из 1000 итераций, пока все 4 полосы SH не будут представлены.

### 7.2 Результаты и оценка

*Результаты.* Мы протестирували наш алгоритм на общем количестве 13 реальных сцен, взятых из ранее опубликованных наборов данных и синтетического набора данных Blender [Mildenhall et al. 2020]. В частности, мы протестирували наш подход на полном наборе сцен, представленных в Mip-Nerf360 [Barron et al. 2022], который является текущим уровнем качества рендеринга NeRF, двух сценах из набора данных Tanks&Temples [2017] и двух сценах, предоставленных Hedman et al. [Hedman et al. 2018]. Сцены, которые мы выбрали, имеют очень разные стили захвата, и охватывают как ограниченные внутренние сцены, так и большие неограниченные внешние среды. Мы используем ту же конфигурацию гиперпараметров для всех экспериментов в нашей оценке. Все результаты указаны для работы на GPU A6000, за исключением метода Mip-NeRF360 (см. ниже). В дополнительных материалах мы показываем видео пути рендеринга для выбора сцен, содержащих виды, далекие от входных фотографий.

*Реальные сцены.* В плане качества текущий уровень развития – это Mip-Nerf360 [Barron et al. 2021]. Мы сравниваемся с этим методом как с качественным эталоном. Мы также сравниваемся с двумя самыми последними быстрыми методами NeRF: InstantNGP [Müller et al. 2022] и PlenoXels [Fridovich-Keil and Yu et al. 2022]. Мы используем разделение train/test для наборов данных, используя методологию, предложенную Mip-NeRF360, брать каждую 8-ю фотографию для теста, для последовательных и значимых сравнений для генерации метрик ошибок, используя стандартные метрики PSNR, L-PIPS и SSIM, наиболее часто используемые в литературе; пожалуйста, см. Таблица ???. Все числа в таблице взяты из наших собственных запусков кода авторов для всех предыдущих методов, за исключением тех, что относятся к Mip-NeRF360 на их наборе данных, где мы скопировали числа из оригинальной публикации, чтобы избежать путаницы о текущем уровне развития. Для изображений в наших фигурах мы использовали наш собственный запуск Mip-NeRF360: числа для этих запусков находятся в Приложении D. Мы также показываем среднее время обучения, скорость рендеринга и объем памяти, используемый для хранения оптимизированных параметров. Мы сообщаем результаты для базовой конфигурации InstantNGP (Base), которая работает в течение 35K итераций, а также немного большей сети, предложенной авторами (Big), и двух конфигураций, 7K и 30K итераций для нашего метода. Мы показываем разницу в визуальном качестве для наших двух конфигураций на Рис. ???. Во многих случаях качество после 7K итераций уже довольно хорошее. Время обучения различается в зависимости от наборов данных, и мы сообщаем их отдельно. Обратите внимание, что разрешение изображений также различается в зависимости от наборов данных. В проектном сайте мы предоставляем все

Таблица 1. Оценка PSNR для тестовых прогонов. Для этого эксперимента мы вручную понизили разрешение высококачественных версий входных изображений каждой сцены до установленного разрешения рендеринга наших других экспериментов. Это снижает случайные артефакты (например, из-за сжатия JPEG в предварительно уменьшенных входных данных Mip-NeRF360).

	Truck-5K	Garden-5K	Bicycle-5K	Truck-30K	Garden-30K	Bicycle-30K	Average-5K	Average-30K
Limited-BW	14.66	22.07	20.77	13.84	22.88	20.87	19.16	19.19
Random Init	16.75	20.90	19.86	18.02	22.19	21.05	19.17	20.42
No-Split	18.31	23.98	22.21	20.59	26.11	25.02	21.50	23.90
No-SH	22.36	25.22	22.88	24.39	26.59	25.08	23.48	25.35
No-Clone	22.29	25.61	22.15	24.82	27.47	25.46	23.35	25.91
Isotropic	22.40	25.49	22.81	23.89	27.00	24.81	23.56	25.23
Full	22.71	25.82	23.18	24.81	27.70	25.65	23.90	26.05

рендеры тестовых видов, которые мы использовали для вычисления статистики для всех методов (наших и предыдущих работ) на все сцены. Обратите внимание, что мы сохранили исходное входное разрешение для всех рендеров. Таблица показывает, что наша полностью сходившаяся модель достигает качества наравне и иногда немного лучше, чем у лидирующего метода Mip-NeRF360; обратите внимание, что на том же оборудовании их среднее время обучения составило 48 часов<sup>2</sup>, по сравнению с нашими 35–45 минутами, а их время рендеринга составляет 10 секунд/кадр. Мы достигаем сравнимого качества с InstantNPG и Plenoxtels после 5–10 минут обучения, но дополнительное время обучения позволяет нам достичь уровня современного развития, чего нельзя сказать о других быстрых методах. Для Tanks & Temples мы достигаем аналогичного качества как базовый InstantNPG при аналогичном времени обучения (~7 минут в нашем случае). Мы также показываем визуальные результаты этого сравнения для тестового вида для нашего метода и предыдущих методов рендеринга, выбранных для сравнения, на Рис. ??; результаты нашего метода после 30К итераций обучения. Мы видим, что в некоторых случаях даже Mip-NeRF360 имеет оставшиеся артефакты, которых избегает наш метод (например, размытость в растительности – в Bicycle, Stump – или на стенах в Room). В дополнительном видео и на веб-странице мы предоставляем сравнения путей с расстояния. Наш метод сохраняет визуальные детали хорошо покрытых областей даже с дальнего расстояния, чего не всегда можно добиться другими методами.

*Синтетические ограниченные сцены.* Помимо реалистичных сцен, мы также оцениваем наш подход на синтетическом наборе данных *Blender* [Mildenhall et al. 2020]. Сцены в вопросе предоставляют исчерпывающий набор видов, ограничены в размере и предоставляют точные параметры камеры. В таких сценариях мы можем достичь результатов уровня современного состояния даже с произвольной инициализацией: мы начинаем обучение с 100К равномерно случайных Гауссиан внутри объема, охватывающего границы сцены. Наш подход быстро и автоматически сокращает их до примерно 6–10К значимых Гауссиан. Конечный размер обученной модели после 30К итераций достигает около 200–500К Гауссиан на сцену. Мы

<sup>2</sup>Мы обучали Mip-NeRF360 на узле с 4 GPU A100 в течение 12 часов, что эквивалентно 48 часам на одном GPU. Обратите внимание, что A100 быстрее, чем GPU A6000.

сообщаем и сравниваем наши достигнутые значения PSNR с предыдущими методами в Таблица ?? с использованием белого фона для совместимости. Примеры можно увидеть на Рис. 8 (второе изображение слева) и в дополнительных материалах. Обученные синтетические сцены рендерятся со скоростью 180–300 FPS.

*Компактность.* В сравнении с предыдущими явными представлениями сцен, анизотропные Гауссианы, используемые в нашей оптимизации, способны моделировать сложные формы с меньшим количеством параметров. Мы демонстрируем это, оценивая наш подход против высоко компактных, основанных на точках моделей, полученных [Zhang et al. 2022]. Мы начинаем с их начального облака точек, которое получено путем пространственного вырезания с масками переднего плана и оптимизируем, пока не достигнем равенства с их заявленными значениями PSNR. Обычно это происходит в течение 2–4 минут. Мы превышаем их заявленные метрики, используя примерно четверть их количества точек, что приводит к среднему размеру модели 3.8 МБ, по сравнению с их 9 МБ. Мы отмечаем, что для этого эксперимента мы использовали только две степени наших сферических гармоник, аналогично их подходу.

### 7.3 Исследования по удалению компонентов

Мы выделили различные вклады и алгоритмические решения, которые мы приняли, и создали серию экспериментов для измерения их влияния. В частности, мы тестируем следующие аспекты нашего алгоритма: инициализация из SfM, наши стратегии уплотнения, анизотропная ковариация, тот факт, что мы позволяем неограниченному количеству сплатов иметь градиенты и использование сферических гармоник. Количественное влияние каждого выбора суммировано в Таблица 1.

*Инициализация из SfM.* Мы также оцениваем важность инициализации 3D Гауссиан из облака точек SfM. Для этого исследования мы равномерно выбираем куб с размером, равным трем размерам ограничивающего прямоугольника входных камер. Мы наблюдаем, что наш метод работает относительно хорошо, избегая полного провала даже без точек SfM. Вместо этого он ухудшается в основном на заднем плане, см. Рис. 5. Также в областях, плохо покрытых обучающими видами, метод случайной инициализации кажется более подверженным



Fig. 5. Инициализация с точками SfM помогает. Выше: инициализация с случайным облаком точек. Ниже: инициализация с использованием точек SfM.

наличию "плавающих" точек, которые не могут быть удалены оптимизацией. С другой стороны, синтетический набор данных NeRF не проявляет такого поведения, потому что у него нет заднего плана и он хорошо ограничен входными камерами (см. обсуждение выше).

**Уплотнение.** Мы далее оцениваем наши два метода уплотнения, а именно стратегию клонирования и разделения, описанные в Разделе 5. Мы отключаем каждый метод по отдельности и оптимизируем, используя остальную часть метода без изменений. Результаты показывают, что разделение больших Гауссиан важно для обеспечения хорошей реконструкции заднего плана, как видно на Рис. 6, тогда как клонирование маленьких Гауссиан вместо их разделения позволяет достичь лучшей и более быстрой сходимости, особенно когда в сцене появляются тонкие структуры.

**Неограниченная глубина сложности сплотов с градиентами.** Мы оцениваем, даст ли пропуск вычисления градиентов после  $N$  передних точек нам скорость без ущерба для качества, как предложено в Pulsar [Lassner and Zollhofer 2021]. В этом тесте мы выбираем  $N=10$ , что в два раза больше значения по умолчанию в Pulsar, но это привело к нестабильной оптимизации из-за серьезного приближения в вычислении градиентов. Для сцены Truck качество ухудшилось на 11 dB в PSNR (см. Таблица 1, Limited-BW), и визуальный результат показан на Рис. 7 для Garden.

**Анизотропная ковариация.** Важным алгоритмическим выбором в нашем методе является оптимизация полной матрицы ковариации для 3D Гауссиан. Чтобы продемонстрировать влияние этого выбора, мы проводим исследование, где удаляем анизотропию, оптимизируя одно скалярное значение, которое



Fig. 6. Исследование стратегии уплотнения для двух случаев "клонирование" и "разделение" (Раздел 5).



Fig. 7. Если мы ограничим количество точек, получающих градиенты, эффект на визуальное качество будет значительным. Слева: лимит в 10 Гауссиан, получающих градиенты. Справа: наш полный метод.

управляет радиусом 3D Гауссиана по всем трем осям. Результаты этой оптимизации представлены визуально на Рис. 8. Мы наблюдаем, что анизотропия значительно улучшает способность 3D Гауссиан выравниваться с поверхностями, что, в свою очередь, позволяет достичь гораздо более высокого качества рендеринга, сохраняя то же количество точек.

**Сферические гармоники.** Наконец, использование сферических гармоник улучшает наши общие показатели PSNR, поскольку они компенсируют зависимые от вида эффекты (Таблица 1).

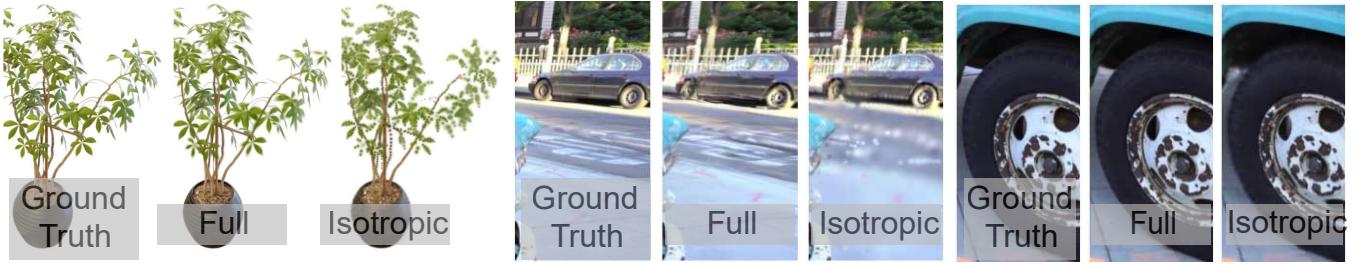


Fig. 8. Мы обучаем сцены с отключенной и включенной анизотропией Гауссиан. Использование анизотропных объемных сплотов позволяет моделировать мелкие структуры и оказывает значительное влияние на визуальное качество. Обратите внимание, что в иллюстративных целях мы ограничили Ficus использованием не более 5К Гауссиан в обеих конфигурациях.

#### 7.4 Ограничения

Наши метод не без ограничений. В регионах, где сцена плохо наблюдается, у нас есть артефакты; в таких регионах другие методы также сталкиваются с трудностями (например, Mip-NeRF360 на Рис. 9). Хотя анизотропные Гауссианы имеют много преимуществ, как описано выше, наш метод может создавать вытянутые артефакты или «пятнистые» Гауссианы (см. Рис. 10); снова предыдущие методы также испытывают трудности в этих случаях. У нас также иногда возникают скачкообразные артефакты, когда наша оптимизация создает большие Гауссианы; это обычно происходит в областях с зависимым от вида внешним видом. Одной из причин этих скачкообразных артефактов является тривиальное отклонение Гауссиан через защитную полосу в растеризаторе. Более принципиальный подход к отбраковке мог бы уменьшить эти артефакты. Другим фактором является наш простой алгоритм видимости, который может приводить к тому, что Гауссианы внезапно меняют порядок глубины/смешивания. Этого можно было бы избежать с помощью сглаживания, что мы оставляем для будущей работы. Кроме того, в настоящее время мы не применяем никакой регуляризации к нашей оптимизации; это помогло бы как с невидимыми областями, так и с скачкообразными артефактами. Хотя мы использовали одни и те же гиперпараметры для всей оценки, ранние эксперименты показывают, что уменьшение скорости обучения положения может быть необходимо для сходимости в очень больших сценах (например, городских наборах данных). Хотя мы весьма компактны по сравнению с предыдущими точечными подходами, потребление нашей памяти значительно выше, чем у решений на основе NeRF. Во время обучения больших сцен пиковое потребление GPU-памяти может превышать 20 ГБ в нашей неоптимизированной прототипе. Однако эта цифра могла бы быть значительно снижена за счет тщательной низкоуровневой реализации логики оптимизации (аналогично InstantNGP). Рендеринг обученной сцены требует достаточного объема GPU-памяти для хранения полной модели (несколько сотен мегабайт для крупных сцен) и дополнительных 30–500 МБ для растеризатора, в зависимости от размера сцены и разрешения изображения. Мы отмечаем, что есть множество возможностей для дальнейшего снижения потребления памяти нашим методом. Техники сжатия для облаков точек — это хорошо изученная область [De Queiroz and

Chou 2016]; было бы интересно увидеть, как такие подходы могут быть адаптированы к нашему представлению.



Fig. 9. Сравнение артефактов сбоя: Mip-NeRF360 имеет «плаывающие объекты» и зернистую текстуру (слева, передний план), тогда как наш метод производит грубые, анизотропные Гауссианы, приводящие к низкой детализации (справа, задний план). Сцена Train.



Fig. 10. В видах, которые мало пересекаются с теми, что были видны во время обучения, наш метод может производить артефакты (справа). Опять же, Mip-NeRF360 также имеет артефакты в этих случаях (слева). Сцена DrJohnson.

## 8 Обсуждение и выводы

Мы представили первый подход, который действительно позволяет рендеринг радиусных полей в реальном времени с высоким качеством в широком диапазоне сцен и стилей захвата, требуя при этом времени обучения, конкурентного с самыми быстрыми предыдущими методами.

Наш выбор примитива 3D Гауссиан сохраняет свойства объемного рендеринга для оптимизации, одновременно позволяя непосредственно выполнять быстрое растеризованное сплайнирование. Наша работа демонстрирует, что — вопреки широко распространенному мнению — непрерывное представление

не является строго необходимым для обеспечения быстрого и высококачественного обучения радиусных полей.

Большая часть (~80%) нашего времени обучения тратится на код Python, поскольку мы создали наше решение на PyTorch, чтобы позволить другим легко использовать наш метод. Только растеризация реализована как оптимизированные CUDA ядра. Мы ожидаем, что перенос всей оптимизации целиком на CUDA, как это сделано, например, в InstantNGP [Müller et al. 2022], может обеспечить значительное дальнейшее ускорение для приложений, где производительность имеет ключевое значение.

Мы также продемонстрировали важность построения на принципах рендеринга в реальном времени, используя мощь GPU и скорость архитектуры программного конвейера растеризации. Эти проектные решения являются ключом к производительности как для обучения, так и для рендеринга в реальном времени, предоставляя конкурентное преимущество в производительности над предыдущим объемным трассированием лучей.

Было бы интересно выяснить, можно ли использовать наши Гауссианы для выполнения реконструкции сетки захваченной сцены. Помимо практических последствий, учитывая широкое использование сеток, это позволило бы нам лучше понять, где именно находится наш метод в континууме между объемными и поверхностными представлениями.

Подводя итог, мы представили первое решение для рендеринга радиусных полей в реальном времени с качеством рендеринга, соответствующим лучшим дорогим предыдущим методам, и временем обучения, конкурентным с самыми быстрыми существующими решениями.

## Acknowledgments

Это исследование было профинансировано грантом ERC Advanced FUNGRAPH № 788065 <http://fungraph.inria.fr>. Авторы благодарны Adobe за щедрые пожертвования, инфраструктуру OPAL от Université Côte d’Azur и за ресурсы HPC от GENCI-IDRIS (Грант 2022-AD011013409). Авторы благодарят анонимных рецензентов за ценные отзывы, П. Хедмана и А. Тевари за проверку ранних черновиков, а также Т. Мюллера, А. Ю и С. Фридович-Кейла за помощь в сравнениях.

## Список литературы

- Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural Point-Based Graphics. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII*. 696–712.
- Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5855–5864.
- Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR* (2022).
- Sebastien Bonopera, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. 2020. sibr: A System for Image Based Rendering. [https://gitlab.inria.fr/sibr/sibr\\_core](https://gitlab.inria.fr/sibr/sibr_core)
- Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. 2005. High-Quality Surface Splatting on Today’s GPUs. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics* (New York, USA) (*SPBG’05*). Eurographics Association, Goslar, DEU, 17–24.
- Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proc. SIGGRAPH*.
- Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 1–12.
- Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022b. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.
- Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2022a. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. *arXiv preprint arXiv:2208.00277* (2022).
- Ricardo L De Queiroz and Philip A Chou. 2016. Compression of 3D point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing* 25, 8 (2016), 3947–3956.
- Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating textures. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 409–418.
- John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world’s imagery. In *CVPR*.
- Fridivit-Keil and Yu, Matthew Tancik, Qinzhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*.
- Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 14346–14355.
- Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. 2007. Multi-view stereo for community photo collections. In *ICCV*.
- Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 43–54.
- Markus Gross and Hanspeter Pfister. 2011. *Point-based graphics*. Elsevier.
- Jeff P. Grossman and William J. Dally. 1998. Point Sample Rendering. In *Rendering Techniques*.
- Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. on Graphics (TOG)* 37, 6 (2018).
- Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-Out Image-Based Rendering. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)* 35, 6 (December 2016). <http://www-sop.inria.fr/revues/Basilic/2016/HRDB16>
- Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).
- Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. 2019. Escaping plato’s cave: 3d shape from adversarial rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9984–9993.
- Arno Knapsitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1–13.
- Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. 2022. Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)* 41, 6 (2022), 201. <http://www-sop.inria.fr/revues/Basilic/2022/KLRJD22>
- Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. *Computer Graphics Forum* 40, 4 (2021), 29–43. <https://doi.org/10.1111/cgf.14339>
- Samuli Laine and Tero Karras. 2011. High-performance software rasterization on GPUs. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*. 79–88.
- Christoph Lassner and Michael Zollhofer. 2021. Pulsar: Efficient Sphere-Based Neural Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1440–1449.
- Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 31–42.
- Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.
- Duane G Merrill and Andrew S Grimshaw. 2010. Revisiting sorting for GPGPU stream architectures. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*. 545–546.
- Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. <https://doi.org/10.1145/3528223.3550127>
- Eric Penner and Li Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–11.
- Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. 2000. Surfels: Surface Elements as Rendering Primitives. In *Proceedings of the 27th Annual*

- Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 335–342. <https://doi.org/10.1145/344779.344936>
- Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In *International Conference on Computer Vision (ICCV)*.
- Liu Ren, Hanspeter Pfister, and Matthias Zwicker. 2002. Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering. *Computer Graphics Forum* 21 (2002).
- Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 765–773.
- Gernot Riegler and Vladlen Koltun. 2020. Free view synthesis. In *European Conference on Computer Vision*. Springer, 623–640.
- Darius Rückert, Linus Franke, and Marc Stamminger. 2022. ADOP: Approximate Differentiable One-Pixel Point Rendering. *ACM Trans. Graph.* 41, 4, Article 99 (jul 2022), 14 pages. <https://doi.org/10.1145/3528223.3530122>
- Miguel Sainz and Renato Pajarola. 2004. Point-based rendering techniques. *Computers and Graphics* 28, 6 (2004), 869–879. <https://doi.org/10.1016/j.cag.2004.08.014>
- Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Markus Schütz, Bernhard Kerl, and Michael Wimmer. 2022. Software Rasterization of 2 Billion Points in Real Time. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 24 (jul 2022), 17 pages. <https://doi.org/10.1145/3543863>
- Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. 2019. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2437–2446.
- Noah Snavely, Steven M Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. In *Proc. SIGGRAPH*.
- Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. 2011. Fast articulated motion tracking using a sums of gaussians body model. In *2011 International Conference on Computer Vision*. IEEE, 951–958.
- Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *CVPR*.
- Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.
- Towaki Takikawa, Joey Litani, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. (2021).
- Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Treitschke, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. 2022. Advances in neural rendering. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 703–735.
- Justus Thies, Michael Zollhofer, and Matthias Nießner. 2019. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.
- Angtian Wang, Peng Wang, Jian Sun, Adam Kortylewski, and Alan Yuille. 2023. VoGE: A Differentiable Volume Renderer using Gaussian Ellipsoids for Analysis-by-Synthesis. In *The Eleventh International Conference on Learning Representations*. [https://openreview.net/forum?id=AdPjb9cud\\_Y](https://openreview.net/forum?id=AdPjb9cud_Y)
- Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. 2020. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7467–7477.
- Xiuchao Wu, Jiamin Xu, Zihan Zhu, Hujun Bao, Qixing Huang, James Tompkin, and Wei Xu. 2022. Scalable Neural Indoor Scene Rendering. *ACM Transactions on Graphics (TOG)* (2022).
- Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 641–676.
- Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5438–5448.
- Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztieli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.
- Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *ICCV*.
- Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. 2022. Differentiable Point-Based Radiance Fields for Efficient View Synthesis. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) (SA '22). Association for Computing Machinery, New York, NY, USA, Article 7, 12 pages. <https://doi.org/10.1145/3550469.3555413>

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. 2016. View synthesis by appearance flow. In *European conference on computer vision*. Springer, 286–301.

Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. 2001a. EWA volume splatting. In *Proceedings Visualization, 2001. VIS'01*. IEEE, 29–538.

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001b. Surface Splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 371–378. <https://doi.org/10.1145/383259.383300>

## A Подробности вычисления градиентов

Напомним, что  $\Sigma/\Sigma'$  — это матрицы ковариации Гауссия в мировом/видовом пространстве,  $q$  — это вращение, а  $s$  — масштабирование,  $W$  — это преобразование просмотра, а  $J$  — Якобиан аффинного приближения проективного преобразования. Мы можем применить цепное правило для нахождения производных по масштабированию и вращению:

$$\frac{d\Sigma'}{ds} = \frac{d\Sigma}{d\Sigma} \frac{d\Sigma}{ds} \quad (8)$$

и

$$\frac{d\Sigma'}{dq} = \frac{d\Sigma}{d\Sigma} \frac{d\Sigma}{dq} \quad (9)$$

Упростив уравнение 5, используя  $U = JW$  и  $\Sigma'$  как (симметричную) верхнюю левую  $2 \times 2$  матрицу  $U\Sigma U^T$ , обозначая элементы матрицы индексами, мы можем найти частные производные  $\frac{\partial\Sigma'}{\partial\Sigma_{ij}} = \begin{pmatrix} U_{1,i}U_{1,j} & U_{1,i}U_{2,j} \\ U_{1,j}U_{2,i} & U_{2,i}U_{2,j} \end{pmatrix}$ . Далее ищем производные  $\frac{d\Sigma}{ds}$  и  $\frac{d\Sigma}{dq}$ . Поскольку  $\Sigma = RSS^TR^T$ , мы можем вычислить  $M = RS$  и переписать  $\Sigma = MM^T$ . Таким образом, мы можем записать  $\frac{d\Sigma}{ds} = \frac{dS}{dM} \frac{dM}{ds}$  и  $\frac{d\Sigma}{dq} = \frac{dS}{dM} \frac{dM}{dq}$ . Так как матрица ковариации  $\Sigma$  (и её градиент) симметрична, общая первая часть компактно находится как  $\frac{d\Sigma}{ds} = 2M^T$ . Для масштабирования также имеем  $\frac{\partial M_{i,j}}{\partial s_k} = \begin{cases} R_{i,k} & \text{если } j = k \\ 0 & \text{иначе} \end{cases}$ . Чтобы вывести градиенты для вращения, вспомним преобразование единичного кватерниона  $q$  с действительной частью  $q_r$  и мнимыми частями  $q_i, q_j, q_k$  в матрицу вращения  $R$ :

$$R(q) = 2 \begin{pmatrix} \frac{1}{2} - (q_i^2 + q_k^2) & (q_i q_j - q_r q_k) & (q_i q_k + q_r q_j) \\ (q_i q_j + q_r q_k) & \frac{1}{2} - (q_i^2 + q_k^2) & (q_j q_k - q_r q_i) \\ (q_i q_k - q_r q_j) & (q_j q_k + q_r q_i) & \frac{1}{2} - (q_i^2 + q_j^2) \end{pmatrix} \quad (10)$$

В результате получаем следующие градиенты для компонентов  $q$ :

$$\begin{aligned} \frac{\partial M}{\partial q_r} &= 2 \begin{pmatrix} 0 & -s_y q_k & s_z q_j \\ s_x q_k & 0 & -s_z q_i \\ -s_x q_j & s_y q_i & 0 \end{pmatrix}, & \frac{\partial M}{\partial q_i} &= 2 \begin{pmatrix} 0 & s_y q_j & s_z q_k \\ s_x q_j & -2s_y q_i & -s_z q_r \\ s_x q_k & s_y q_r & -2s_z q_i \end{pmatrix} \\ \frac{\partial M}{\partial q_j} &= 2 \begin{pmatrix} -2s_x q_j & s_y q_i & s_z q_r \\ s_x q_i & 0 & s_z q_k \\ -s_x q_r & s_y q_k & -2s_z q_j \end{pmatrix}, & \frac{\partial M}{\partial q_k} &= 2 \begin{pmatrix} -2s_x q_k & -s_y q_r & s_z q_i \\ s_x q_r & -2s_y q_k & s_z q_j \\ s_x q_i & s_y q_j & 0 \end{pmatrix} \end{aligned} \quad (11)$$

Выход градиентов для нормализации кватерниона тривиален.

## B Алгоритм оптимизации и увеличения плотности

Наши алгоритмы оптимизации и увеличения плотности суммированы в Алгоритме 1.

## C Подробности растеризатора

**Сортировка.** Наш дизайн основан на предположении о большом количестве маленьких сплотов, и мы оптимизируем это

---

**Algorithm 1** Оптимизация и увеличение плотности  
 $w, h$ : ширина и высота обучающих изображений

```

 $M \leftarrow \text{SfM Точки}$                                 ▷ Позиции
 $S, C, A \leftarrow \text{InitAttributes()}$                 ▷ Ковариации, Цвета,
Непрозрачности
 $i \leftarrow 0$                                          ▷ Счетчик итераций
while не сходится do
     $V, \hat{I} \leftarrow \text{SampleTrainingView()}$  ▷ Камера  $V$  и Изображение
     $I \leftarrow \text{Растеризовать}(M, S, C, A, V)$            ▷ Алг. 2
     $L \leftarrow \text{Loss}(I, \hat{I})$                            ▷ Потери
     $M, S, C, A \leftarrow \text{Adam}(ablaL)$  ▷ Обратное распространение
ошибки и шаг
    if IsRefinementIteration( $i$ ) then
        for all Гауссианы  $(\mu, \Sigma, c, \alpha)$  в  $(M, S, C, A)$  do
            if  $\alpha < \epsilon$  или IsTooLarge( $\mu, \Sigma$ ) then      ▷ Обрезка
                УдалитьГауссиану()
            end if
            if  $ablaL > \tau_p$  then          ▷ Увеличение плотности
                if  $\|S\| > \tau_S$  then          ▷ Пересоздание
                    РазделитьГауссиану( $\mu, \Sigma, c, \alpha$ )
                else                      ▷ Недостаточная реконструкция
                    КлонироватьГауссиану( $\mu, \Sigma, c, \alpha$ )
                end if
            end if
        end for
    end if
     $i \leftarrow i + 1$ 
end while

```

---

путем однократной сортировки сплатов для каждого кадра с использованием поразрядной сортировки в начале. Мы разделяем экран на тайлы размером  $16 \times 16$  пикселей (или бины). Мы создаем список сплатов для каждого тайла, создавая экземпляр каждого сплата в каждом перекрывающем его тайле  $16 \times 16$ . Это приводит к умеренному увеличению числа Гауссиан для обработки, что, однако, амортизируется более простым потоком управления и высокой параллельностью оптимизированной поразрядной сортировки GPU [Merrill and Grimshaw 2010]. Мы назначаем ключ для каждого экземпляра сплата длиной до 64 бит, где младшие 32 бита кодируют его проекционную глубину, а старшие биты кодируют индекс перекрываемого тайла. Точная длина индекса зависит от того, сколько тайлов помещается в текущее разрешение. Сортировка по глубине таким образом решается напрямую для всех сплатов параллельно с помощью одной поразрядной сортировки. После сортировки мы можем эффективно создавать списки Гауссиан для обработки для каждого тайла, определяя начало и конец диапазонов в отсортированном массиве с одинаковым ID тайла. Это делается параллельно, запуская один поток для каждого 64-битного элемента массива для сравнения его старших 32 бит с двумя соседями. По сравнению с [Lassner and Zollhofer 2021], наша растеризация полностью исключает последовательные этапы обработки примитивов и создает более компактные списки для обхода во время прямого прохода для каждого тайла. Мы

показываем высокоуровневый обзор подхода к растеризации в Алгоритме 2.

---

**Algorithm 2** GPU программный растеризатор 3D Гауссиан

$w, h$ : ширина и высота изображения для растеризации  
 $M, S$ : средние значения и ковариации Гауссиан в мировом пространстве  
 $C, A$ : цвета и непрозрачности Гауссиан  
 $V$ : конфигурация камеры текущего вида

---

```

function Растеризовать( $w, h, M, S, C, A, V$ )
    ОбрезатьГауссианы( $p, V$ )           ▷ Отсечение пирамиды
    видимости
     $M', S' \leftarrow \text{ScreenspaceGaussians}(M, S, V)$  ▷ Преобразование
     $T \leftarrow \text{СоздатьТайлы}(w, h)$ 
     $L, K \leftarrow \text{ДублироватьСКлючами}(M', T)$       ▷ Индексы и
    ключи
    СортироватьПоКлючам( $K, L$ )          ▷ Глобальная сортировка
     $R \leftarrow \text{ОпределитьДиапазоныТайлов}(T, K)$ 
     $I \leftarrow 0$                           ▷ Инициализация холста
    for all Тайлы  $t$  в  $I$  do
        for all Пиксели  $i$  в  $t$  do
             $r \leftarrow \text{ПолучитьДиапазонТайла}(R, t)$ 
             $I[i] \leftarrow \text{СмешатьПоПорядку}(i, L, r, K, M', S', C, A)$ 
        end for
    end for
    return  $I$ 
end function

```

---

*Числовая стабильность.* Во время обратного прохода мы восстанавливаем промежуточные значения непрозрачности, необходимые для вычисления градиентов, многократно деля накопленную непрозрачность из прямого прохода на  $\alpha$  каждой Гауссианы. Если реализовано наивно, этот процесс подвержен числовым нестабильностям (например, деление на 0). Чтобы решить эту проблему, как в прямом, так и в обратном проходе, мы пропускаем любые обновления смешивания с  $\alpha < \epsilon$  (мы выбираем  $\epsilon$  равным  $\frac{1}{255}$ ) и также ограничиваем  $\alpha$  сверху значением 0.99. Наконец, перед тем как Гауссиана будет включена в прямой проход растеризации, мы вычисляем накопленную непрозрачность, если бы она была включена, и прекращаем фронтальное смешивание **до** того, как оно сможет превысить 0.9999.

## D Метрики ошибок для каждой сцены

Таблицы 2–7 содержат различные собранные метрики ошибок для нашей оценки всех рассматриваемых техник и реальных сцен. Мы приводим как скопированные числа Mip-NeRF360, так и те, которые были получены нами для создания изображений в статье; средние значения по полному набору данных Mip-NeRF360 составляют PSNR 27.58, SSIM 0.790 и LPIPS 0.240.

Таблица 2. SSIM метрики для сцен Mip-NeRF360. † скопировано из оригинальной статьи.

	велосипед	цветы	сад	пень	холм с деревьями	комната	стол	кухня	бонсай
Plenoxels	0.496	0.431	0.6063	0.523	0.509	0.8417	0.759	0.648	0.814
INGP-Base	0.491	0.450	0.649	0.574	0.518	0.855	0.798	0.818	0.890
INGP-Big	0.512	0.486	0.701	0.594	0.542	0.871	0.817	0.858	0.906
Mip-NeRF360 <sup>†</sup>	0.685	0.583	0.813	0.744	0.632	0.913	0.894	0.920	<b>0.941</b>
Mip-NeRF360	0.685	0.584	0.809	0.745	0.631	0.910	0.892	0.917	0.938
Ours-7k	0.675	0.525	0.836	0.728	0.598	0.884	0.873	0.900	0.910
Ours-30k	<b>0.771</b>	<b>0.605</b>	<b>0.868</b>	<b>0.775</b>	<b>0.638</b>	<b>0.914</b>	<b>0.905</b>	<b>0.922</b>	0.938

Таблица 7. LPIPS метрики для сцен Tanks&amp;Temples и Deep Blending.

	Грузовик	Поезд	Доктор Джонсон	Игровая комната
Plenoxels	0.335	0.422	0.521	0.499
INGP-Base	0.274	0.386	0.381	0.465
INGP-Big	0.249	0.360	0.352	0.428
Mip-NeRF360	0.159	0.354	<b>0.237</b>	0.252
Ours-7k	0.209	0.350	0.343	0.291
Ours-30k	<b>0.148</b>	<b>0.218</b>	0.244	<b>0.241</b>

Таблица 3. PSNR метрики для сцен Mip-NeRF360. † скопировано из оригинальной статьи.

	велосипед	цветы	сад	пень	холм с деревьями	комната	стол	кухня	бонсай
Plenoxels	21.912	20.097	23.4947	20.661	22.248	27.594	23.624	23.420	24.669
INGP-Base	22.193	20.348	24.599	23.626	22.364	29.269	26.439	28.548	30.637
INGP-Big	22.171	20.652	25.069	23.466	22.373	29.690	26.691	29.479	30.685
Mip-NeRF360 <sup>†</sup>	24.37	<b>21.73</b>	26.98	26.40	22.87	<b>31.63</b>	<b>29.55</b>	<b>32.23</b>	<b>33.46</b>
Mip-NeRF360	24.305	21.649	26.875	26.175	<b>22.929</b>	31.467	29.447	31.989	33.397
Ours-7k	23.604	20.515	26.245	25.709	22.085	28.139	26.705	28.546	28.850
Ours-30k	<b>25.246</b>	21.520	<b>27.410</b>	<b>26.550</b>	22.490	30.632	28.700	30.317	31.980

Таблица 4. LPIPS метрики для сцен Mip-NeRF360. † скопировано из оригинальной статьи.

	велосипед	цветы	сад	пень	холм с деревьями	комната	стол	кухня	бонсай
Plenoxels	0.506	0.521	0.3864	0.503	0.540	0.4186	0.441	0.447	0.398
INGP-Base	0.487	0.481	0.312	0.450	0.489	0.301	0.342	0.254	0.227
INGP-Big	0.446	0.441	0.257	0.421	0.450	0.261	0.306	0.195	0.205
Mip-NeRF360 <sup>†</sup>	0.301	0.344	0.170	0.261	0.339	<b>0.211</b>	<b>0.204</b>	<b>0.127</b>	<b>0.176</b>
Mip-NeRF360	0.305	0.346	0.171	0.265	0.347	0.213	0.207	0.128	0.179
Ours-7k	0.318	0.417	0.153	0.287	0.404	0.272	0.254	0.161	0.244
Ours-30k	<b>0.205</b>	<b>0.336</b>	<b>0.103</b>	<b>0.210</b>	<b>0.317</b>	0.220	<b>0.204</b>	0.129	0.205

Таблица 5. SSIM метрики для сцен Tanks&amp;Temples и Deep Blending.

	Грузовик	Поезд	Доктор Джонсон	Игровая комната
Plenoxels	0.774	0.663	0.787	0.802
INGP-Base	0.779	0.666	0.839	0.754
INGP-Big	0.800	0.689	0.854	0.779
Mip-NeRF360	0.857	0.660	<b>0.901</b>	0.900
Ours-7k	0.840	0.694	0.853	0.896
Ours-30k	<b>0.879</b>	<b>0.802</b>	0.899	<b>0.906</b>

Таблица 6. PSNR метрики для сцен Tanks&amp;Temples и Deep Blending.

	Грузовик	Поезд	Доктор Джонсон	Игровая комната
Plenoxels	23.221	18.927	23.142	22.980
INGP-Base	23.260	20.170	27.750	19.483
INGP-Big	23.383	20.456	28.257	21.665
Mip-NeRF360	24.912	19.523	<b>29.140</b>	29.657
Ours-7k	23.506	18.892	26.306	29.245
Ours-30k	<b>25.187</b>	<b>21.097</b>	28.766	<b>30.044</b>