Репараметризационный трюк для дифференцируемого рендеринга

Кирилл Струминский

26 июня 2023 г.

1 Введение

За последние годы дифференцируемый подход к отрисовке трехмерных сцен привел к существенному прогрессу в задаче многовидовой реконструкции. Залогом успеха многих работ являются обучаемые представления сцены в виде полей плотности и светимости. Эти представления принимают различные формы. Например, в одной из первых работ в этой области, предложившей алгоритм NeRF [4], поля параметризуются многослойным перцептроном (MLP). В алгоритме DVGO, приведенном в работе [7], поля плотности и светимости параметризованные напрямую с помощью дискретной объемной сетки. Несмотря на архитектурное разнообразие представлений, подавляющее большинство работ использует алгоритм рендеринга, предложенный несколько десятков лет назад в статье [3]. В докладе пойдет речь об алгоритме рендеринга, который может послужить заменой алгоритму, использованному в статье [4] и последующих работах.

Альтернатива востребована, поскольку стандартный алгоритм отрисовки сцен оказывается неэффективен с вычислительной точки зрения. Модель NeRF работает следующим образом: при генерации одной точки на изображении через нее запускается луч, также проходящий через камеру, а цвет точки вычисляется как взвешенная сумма значений светимости вдоль луча с весами, пропорциональными значениям плотности в точках. Подсчет каждого слагаемого в сумме требует вычислительно трудоемкого запуска нейронной сети, что порождает компромисс между вычислительной эффективностью и качеством рендеринга (в зависимости от числа используемых слагаемых).

Целью нашей работы является разработка альтернативного NeRF подхода к отрисовке на основе Монте-Карло оценок, оптимизирующего описанную выше процедуру, и проведение оценки качества его работы с точки зрения затрачиваемого времени и метрик реконструкции трехмерных сцен. Дальше мы опишем идею предложенного подхода и приведем примеры работы в рамках двух приложений.

2 Репараметризационный трюк

Мы предлагаем алгоритм рендеринга на основе репараметризационного трюка и семплирования через обратную функцию распределения, работающий в два этапа. На первом

этапе алгоритм проходит по запущенному лучу чтобы оценить плотность сцены вдоль луча. На втором — строит Монте-Карло приближение цвета, используя для генерации точек распределение вероятностей, порожденное выучиваемым полем плотности. Рисунок 1 иллюстрирует оценки с разным количеством точек. Как видно по динамике в изображениях, качество рендеринга постепенно улучшается с ростом числа точек, при этом 8 точек уже дают результат, близкий к оптимальному. Полученная оценка полностью дифференцируема и не требует обучения дополнительных моделей. Помимо этого, модели нужно лишь небольшое количество точек для того, чтобы построить достаточно точное приближение цвета. Интуитивно, значения светимости есть смысл считать только на тех точках луча, где он попадает на твердую поверхность. Как результат, наш алгоритм оказывается особенно удобным для современных архитектур, например архитектур из работ [5], [1], [7], которые используют отдельные модели для параметризации светимости и плотности. Так, на первом этапе алгоритма считается только плотность, а на втором — только светимость. В сравнении с классическим алгоритмом рендеринга, второй этап нашего алгоритма избегает вычисления светимости в избыточном числе точек и, тем самым, снижает необходимое количество времени и памяти ценой появления (небольшой) дисперсии при использовании стохастической оценки.

Ниже, мы приводим более формальное описание разработанного метода. Ожидаемый цвет вдоль луча вычисляется по формуле

$$C(\boldsymbol{r}) = \int_0^{+\infty} c_{\boldsymbol{r}}(t)\sigma_{\boldsymbol{r}}(t) \exp\left(-\int_0^t \sigma_{\boldsymbol{r}}(s)\mathrm{d}s\right)\mathrm{d}t, \tag{1}$$

где $c_{\boldsymbol{r}}(\cdot)$ задает поле светимости, а $\sigma_{\boldsymbol{r}}(\cdot)$ задает поле плотности.

В Уравнении 1 можно выделить множитель $p_{\boldsymbol{r}}(\cdot) = \sigma_{\boldsymbol{r}}(t)\exp(-\int_0^t \sigma_{\boldsymbol{r}}(s)\mathrm{d}s)$, который представляет из себя не нормированную плотность. Обозначим также за α нормировочную константу

$$\alpha = \int_0^{+\infty} \sigma_{\boldsymbol{r}}(t) \exp\left(-\int_0^t \sigma_{\boldsymbol{r}}(s)\mathrm{d}s\right)\mathrm{d}t. \tag{2}$$

Наше решение вдохновлено репараметризационным трюком [2, 6]. Мы делаем замену переменной в Уравнении 1. Введя $F_{\boldsymbol{r}}(t) := 1 - \exp\left(-\int_0^t \sigma_{\boldsymbol{r}}(s)\mathrm{d}s\right)$ и $y := F_{\boldsymbol{r}}(t)$, мы переписываем

$$C(\boldsymbol{r}) = \int_0^{+\infty} c_{\boldsymbol{r}}(t)p_{\boldsymbol{r}}(t)\mathrm{d}t = \int_0^{\alpha} c_{\boldsymbol{r}}(F_{\boldsymbol{r}}^{-1}(y))\mathrm{d}y = \int_0^1 \alpha c_{\boldsymbol{r}}(F_{\boldsymbol{r}}^{-1}(\alpha u))\mathrm{d}u. \tag{3}$$

Пределы интегрирования равны $F_{\boldsymbol{r}}(0) = 0$ и $\alpha = \lim_{t\to+\infty} F_{\boldsymbol{r}}(t)$. Функция $F_{\boldsymbol{r}}(t)$ представляет из себя не нормированную функцию распределения случайной величины t. В рендеринге объемных сцен функция $F_{\boldsymbol{r}}(t)$ носит название функции непрозрачности, а её правый предел α равен непрозрачности точки изображения, через которую проходит луч $\boldsymbol{r}$. После первого преобразования в Уравнении 3 пределы интегрирования зависят от прозрачности $F_{\boldsymbol{r}}$ и, как следствие, от плотности на луче $\sigma_{\boldsymbol{r}}$. Вторым преобразованием мы избавляемся от этой зависимости, переходя к пределам интегрирования в интервале $[0, 1]$.

2.1 Оценки ожидаемого цвета по методу Монте-Карло

Учитывая приведенный выше вывод, мы строим *репараметризованную оценку Монте-Карло* для интеграла правой части в Уравнении 3:

$$\hat{C}(\boldsymbol{r}) := \frac{\alpha}{k} \sum_{i=1}^{k} c_{\boldsymbol{r}}(F_{\boldsymbol{r}}^{-1}(\alpha u_i)), \tag{4}$$

с k независимыми одинаково определенными точками $u_1, \ldots, u_k$ из распределения $U[0, 1]$. Легко показать, что оценка в Уравнении 4 является несмещенной оценкой ожидаемого цвета в Уравнении 1, а его градиент является несмещенной оценкой градиента ожидаемого цвета $\mathrm{C}(\boldsymbol{r})$. Кроме того, мы предлагаем заменить случайные величины $u_1, \ldots, u_k$ независимыми случайными величинами в пределах ячеек регулярной сетки $v_i \sim U[\frac{i-1}{k+1}, \frac{i}{k+1}], i = 1, \ldots, k$. Последняя замена дает стратифицированный вариант оценки из Уравнения 1 и в большинстве случаев приводят к более низким оценкам дисперсии. Важно отметить, что в приведенной выше оценке случайные величины $u_1, \ldots, u_k$ не зависят от плотности $\sigma_{\boldsymbol{r}}$ или цвета $c_{\boldsymbol{r}}$. По сути, для репараметризованной оценки Монте-Карло мы генерируем точки из $p_{\boldsymbol{r}}(t)$, используя обратную функцию распределения $F_{\boldsymbol{r}}^{-1}(\alpha u)$.

2.2 Иерархическая генерация точек на луче

Наконец, мы предлагаем применить наш алгоритм к иерархической схеме выбора точек на луче, также предложенной в NeRF. Для этого мы также введем вспомогательную плотность, но изменим алгоритм её обучения. Во-первых, мы заменим алгоритм генерации точек из вспомогательной плотности на наш алгоритм генерации точек из $p_r(t)$, использующий обратную функцию распределения. Во-вторых, мы не будем использовать вспомогательную функцию потерь для обучения плотности, а вместо этого мы будет пробрасывать градиенты через дифференцируемый алгоритм генерации точек. Последнее позволяет нам отказаться от вспомогательной функции потерь, обучая вспомогательную плотность напрямую решать интересующую нас задачу: выбрать лучшие точки для оценки накопленного цвета. Мы также называем вспомогательную плотность предложеной, поскольку такое название лучше отражает ее назначение.

3 Эксперименты

Рендеринг В этом разделе мы исследуем предлагаемый алгоритм рендеринга, который использует оценку цвета из Уравнения 4. Учитывая точную аппроксимацию поля плотности, наша оценка цвета является несмещенной для любого заданного количества выборок k. Для экспериментов мы выбрали параметризацию сцены из работы DVGO [7], поскольку она позволяет быстро оценивать плотность, в то время как на вычислений светимости уходит значительная доля вычислительных ресурсов. В наших экспериментах мы брали параметры модели по умолчанию и лишь заменяли только алгоритм рендеринга.

Мы оценили влияние дополнительной дисперсии оценки на точность рендеринга, результаты приведены слева на Рисунке 2. Наш алгоритм достигает того же среднего зна-

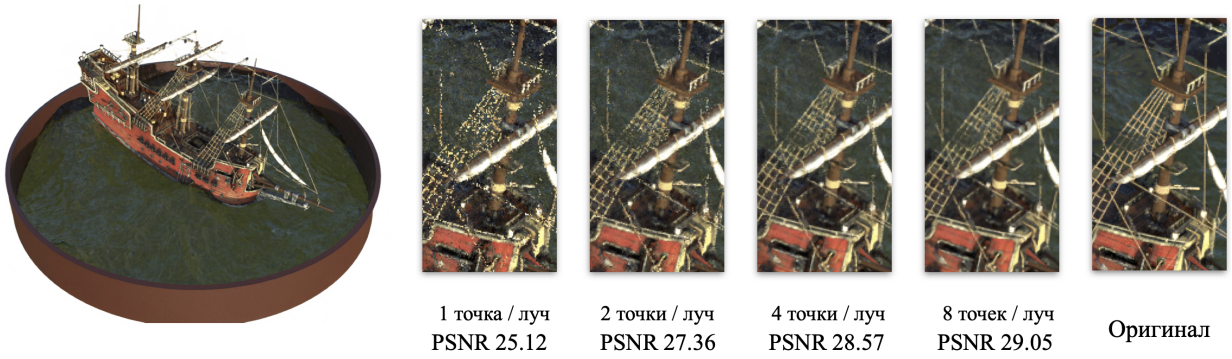

Рис. 1: Новые виды корабля, созданные с помощью предложенных оценок яркости Монте-Карло. Для каждого луча мы оцениваем плотность, а затем вычисляем яркость в нескольких точках луча, сгенерированных с использованием плотности луча. Как видно из приведенных выше изображений, качество рендеринга постепенно улучшается с количеством выборок лучей без видимых артефактов при восьми точках на луче.

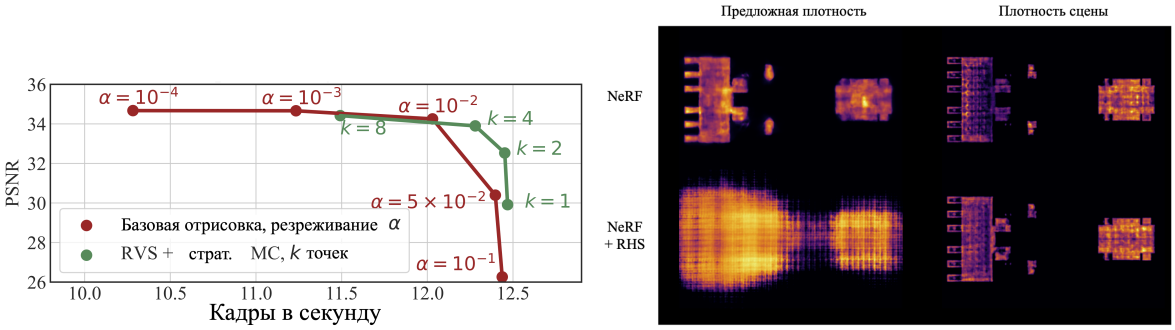

Рис. 2: **Слева:** Качество рендеринга во время тестирования в зависимости от скорости рендеринга. Используя представление сцены, предварительно обученное стандартным алгоритмом рендеринга, мы оцениваем алгоритмы рендеринга в различных конфигурациях. **Справа:** Визуализация двухмерного фрагмента предложной плотности и итоговой плотности сцены Lego.

чения PSNR с выборками из k и даже превосходит стандартный алгоритм рендеринга с более высоким порогом разреженности α.

Однако, как показали наши дальнейшие эксперименты, обучение с нашей аппроксимацией цвета дает худшее качество реконструкции сцены по сравнению с базовой аппроксимацией. Более низкое качество реконструкции моделей, обученных с помощью оценок Монте-Карло, требует дальнейшего изучения.

Иерархическое сэмплирование В этом разделе мы оцениваем предлагаемый подход к иерархической выборке точек на луче для модели NeRF. Окончательная цветовая аппроксимация вычисляется, как в Уравнении 1, а в сравнении мы изменяем лишь алгоритм выбора узловых точек аппроксимации интеграла. Каждый запуск описывается конфигурацией, состоящей из пары чисел (N_p, N_f). Значение N_p определяет количество вычислений предложной плотности, а N_f определяет общее число точек на луче, выбранных для при-

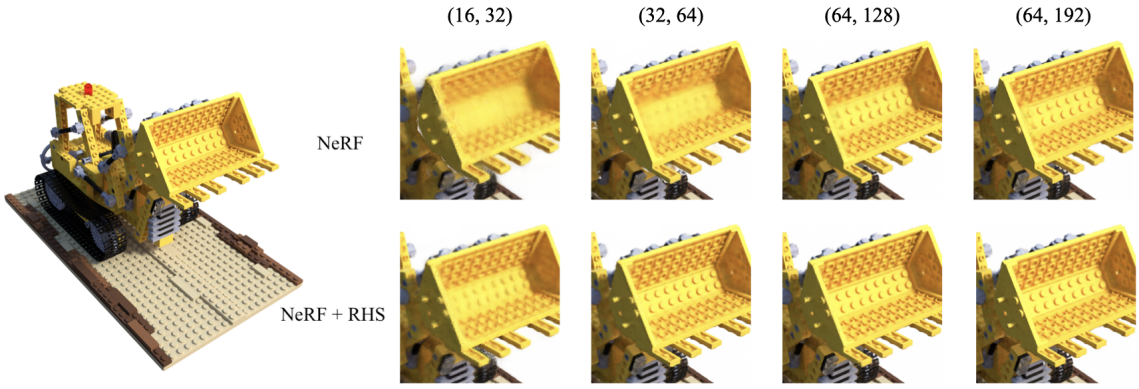

Рис. 3: Сравнение рендеринга вида тестового набора на сцене Lego (Blender). Строки соответствуют разным конфигурациям (N_p, N_f). Базовая линия NeRF имеет трудности с восстановлением мелких деталей для конфигураций $(16, 32)$ и $(32, 64)$, а некоторые части остаются размытыми даже в конфигурациях $(64, 128)$ и $(64, 192)$, в то время как наш метод уже дает реалистичную реконструкцию для конфигурации $(32, 64)$.

ближения итогового цвета. Наш алгоритм обозначен аббревиатурой RHS *(Reparameterized Hierarchical Sampling)*.

Мы начинаем сравнение на сцене Lego набора данных Blender [4] для различных конфигураций (N_p, N_f), которые соответствуют количеству предложений и точным оценкам сети. Наш метод превзошел базовый уровень во всех конфигурациях и по всем показателям, за исключением LPIPS в конфигурации $(64, 192)$, где он показал аналогичную производительность. Мы видим, что улучшение более существенно для меньших (N_p, N_f). Наш метод также имеет незначительное ускорение по сравнению с базовым из-за того, что первый не использует компонент излучения сети предложений. На Рисунке 3 приведена визуализация моделей, обученных двумя методами.

Мы также визуализируем предложную и финальную плотности, полученные двумя алгоритмами на Рисунке 2. Рисунки строятся для некоторого фиксированного значения координаты z, изображая разрез плотности на (x, y)-плоскости. В то время как визуализация с итоговой плотностью почти не отличается, плотность предложной оказывается очень разной. Это происходит из-за того, что исходный алгоритм обучает предложную плотность, пытаясь реконструировать сцену (но используя меньшее количество точек для оценки цвета), в то время как наш алгоритм обучает плотность именно выбирать точки аппроксимации, что приведет к лучшему общему результату реконструкции. Мы считаем, что это также может объяснить улучшенное качество рендеринга, которое показывает наш метод.

Список литературы

[1] Sara Fridovich-Keil и др. “Plenoxels: Radiance Fields Without Neural Networks”. В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, с. 5501—5510.

[2] Diederik P Kingma и Jimmy Ba. “Adam: A Method for Stochastic Optimization”. В: *ICLR (Poster)*. 2015.

[3] Nelson Max. “Optical models for direct volume rendering”. В: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), с. 99—108.

[4] Ben Mildenhall и др. “Nerf: Representing scenes as neural radiance fields for view synthesis”. В: *European conference on computer vision*. Springer. 2020, с. 405—421.

[5] Thomas Müller и др. “Instant Neural Radiance Fields”. В: *ACM SIGGRAPH 2022 Real-Time Live!* 2022, с. 1—2.

[6] Danilo Jimenez Rezende, Shakir Mohamed и Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. В: *International conference on machine learning*. PMLR. 2014, с. 1278—1286.

[7] Cheng Sun, Min Sun и Hwann-Tzong Chen. “Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction”. В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, с. 5459—5469.