

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО  
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
GRADUATION THESIS**

**Разработка метода рендеринга облаков в реальном времени на основе моделей  
машинного обучения**

**Обучающийся / Student** Федотов Богдан Сергеевич

**Факультет/институт/клластер/ Faculty/Institute/Cluster** школа разработки видеоигр  
**Группа/Group** J4221

**Направление подготовки/ Subject area** 09.04.03 Прикладная информатика

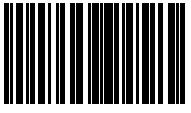
**Образовательная программа / Educational program** Технологии разработки  
компьютерных игр 2022

**Язык реализации ОП / Language of the educational program** Русский

**Квалификация/ Degree level** Магистр

**Руководитель ВКР/ Thesis supervisor** Карсаков Андрей Сергеевич, кандидат технических  
наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная  
категория "ординарный доцент")

Обучающийся/Student

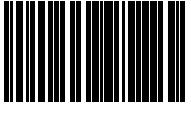
Документ подписан	
Федотов Богдан Сергеевич	
31.05.2024	

(эл. подпись/ signature)

Федотов Богдан  
Сергеевич

(Фамилия И.О./ name  
and surname)

Руководитель ВКР/  
Thesis supervisor

Документ подписан	
Карсаков Андрей Сергеевич	
30.05.2024	

(эл. подпись/ signature)

Карсаков  
Андрей  
Сергеевич

(Фамилия И.О./ name  
and surname)

**Министерство науки и высшего образования Российской Федерации**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**  
**ITMO University**

**АННОТАЦИЯ**  
**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**  
**SUMMARY OF A GRADUATION THESIS**

**Обучающийся / Student** Федотов Богдан Сергеевич

**Факультет/институт/клластер/ Faculty/Institute/Cluster** школа разработки видеоигр

**Группа/Group** J4221

**Направление подготовки/ Subject area** 09.04.03 Прикладная информатика

**Образовательная программа / Educational program** Технологии разработки  
компьютерных игр 2022

**Язык реализации ОП / Language of the educational program** Русский

**Квалификация/ Degree level** Магистр

**Тема ВКР/ Thesis topic** Разработка метода рендеринга облаков в реальном времени на  
основе моделей машинного обучения

**Руководитель ВКР/ Thesis supervisor** Карсаков Андрей Сергеевич, кандидат технических  
наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная  
категория "ординарный доцент")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**  
**DESCRIPTION OF THE GRADUATION THESIS**

**Цель исследования / Research goal**

Повышение производительности рендеринга облаков при помощи нейросетей, при этом не  
жертвуя визуальным качеством и не увеличивая требования по памяти.

**Задачи, решаемые в ВКР / Research tasks**

Изучение существующих методов рендеринга реалистичных облаков в реальном времени,  
выявление передовых методов и их слабых мест, разработка теоретической модели  
улучшенной версии метода рендеринга облаков, практическая имплементация  
улучшенного метода рендеринга облаков, проведение экспериментов над улучшенным  
методом.

**Краткая характеристика полученных результатов / Short summary of results/findings**

Было разработано две новые модификации алгоритма Neural Radiance Caching (NRC).

Первой модификацией является использование адаптивной фильтрации тренировочных  
пакетов (batch) с применением многоуровневой прямоугольной сетки. Благодаря этой  
модификации, пустые тренировочные пакеты пропускаются, уменьшая время тренировки в  
случаях, когда облако занимает не весь экран целиком. Вторая модификация добавляет  
нормализованную плотность облака во входные данные нейросети NRC, что сократило  
время тренировки нейросети примерно в шесть раз. Дополнительно, в данной работе были  
проведены эксперименты над новыми методами вычисления коэффициента пропускания из  
статьи An unbiased ray-marching transmittance estimator. Результаты экспериментов  
подтвердили возможность их совместного применения с алгоритмом NRC. Эти методы не

только уменьшают дисперсию рендеринга трассировки путей, но и уменьшают значение функции потерь нейросети NRC, увеличивая качество тренировки нейросети.

Обучающийся/Student

Документ подписан	
Федотов Богдан Сергеевич	
31.05.2024	

(эл. подпись/ signature)

Федотов Богдан  
Сергеевич

(Фамилия И.О./ name  
and surname)

Руководитель ВКР/  
Thesis supervisor

Документ подписан	
Карсаков Андрей Сергеевич	
30.05.2024	

(эл. подпись/ signature)

Карсаков  
Андрей  
Сергеевич

(Фамилия И.О./ name  
and surname)

## СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ .....	6
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ .....	7
ВВЕДЕНИЕ .....	8
1 КОНТЕКСТ ИССЛЕДОВАНИЯ.....	11
1.1 Уравнение переноса излучения .....	11
1.2 Трассировка путей .....	12
2 ОБЗОР МЕТОДОВ РЕНДЕРИНГА ОБЛАКОВ И ВЫЧИСЛЕНИЯ РАССЕЯНИЯ .....	15
2.1 Рендеринг облаков, заданных явными поверхностями .....	15
2.1.1 Полигональные меши .....	15
2.1.2 Системы частиц .....	16
2.1.3 Облака точек .....	16
2.2 Объёмы и трассировка путей .....	18
2.3 Рендеринг с применением нейросетей .....	20
2.3.1 Нейронный рендеринг .....	21
2.3.2 Объёмный рендеринг с применением нейронных сетей.....	26
2.4 Neural Radiance Caching.....	27
2.4.1 Входные данные нейросети NRC и их сбор .....	28
2.4.2 Архитектура нейронной сети .....	29
3 МЕТОДЫ ПРОВЕДЕНИЯ ИССЛЕДОВАНИЯ .....	31
3.1 Базовая имплементация алгоритма NRC.....	31
3.2 Датасет облаков .....	31
3.3 Экспериментальные метрики .....	31
3.4 Системные характеристики.....	32
4 УЛУЧШЕНИЯ АЛГОРИТМА NRC ДЛЯ РЕНДЕРИНГА ОБЛАКОВ	33
4.1 Дополнительные входные данные об облаке .....	33

4.2 Фильтрация тренировочных пакетов с помощью многоуровневой прямоугольной сетки .....	35
5 НОВЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ КОЭФФИЦИЕНТА ПРОПУСКАНИЯ .....	41
6 РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ .....	43
6.1 Дополнительные входные данные об облаке .....	43
6.2 Фильтрация тренировочных пакетов .....	46
6.3 Новые методы вычисления коэффициента пропускания .....	47
ПЛАНЫ НА БУДУЩЕЕ.....	51
ЗАКЛЮЧЕНИЕ .....	52
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	53
ПРИЛОЖЕНИЕ А АЛГОРИТМ ПОИСКА ТРЕНИРОВОЧНЫХ ПАКЕТОВ В ДЕРЕВЕ.....	60

## **СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

SPP — Samples per pixel

FPS — Frames per second

MSE — Mean squared error

NeRF — Neural Radiance Fields

NRC — Neural Radiance Caching

ReSTIR — Reservoir-based Spatio-Temporal Importance Resampling

GI — Global illumination

MLP — Multi Layer Perceptron

MB — Megabyte

GPU — Graphics processing unit

## ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

*Смещение (bias)* — разница между ожидаемым и реальным значением вычисляемого параметра.

*Дисперсия (variance)* — разброс значений измеряемого параметра.

*Функция потерь (loss)* — функция, переводящая предсказание нейросети в численное измерение качества этого измерения по сравнению с эталонным значением

*Пакет (batch)* — подмножество тренировочной или инференсной выборки нейросети фиксированного размера. При пакетной тренировке, на каждой итерации выборка просматривается целиком, и только после этого изменяются веса модели.

*Рендеринг* — процесс создания изображения по компьютерной модели

Нейронный рендеринг — в глобальном смысле, рендеринг с применением нейронных сетей. В современном понимании, алгоритм обратного рендеринга, при котором параметры сцены представляются в виде нейросети и оптимизируются с помощью обратного распространения ошибки из фиксированного дифференцируемого алгоритма рендеринга. Такая нейросеть носит название «нейронное представление сцены».

*Фазовая функция (phase function)* — угловое распределение интенсивности света, рассеянного частицей, при заданной длине волны. Определяется физическими свойствами среды.

*Коэффициент пропускания (transparency)* — физическая характеристика среды, обозначающая порцию света, проходящую сквозь среду без рассеяния.

## ВВЕДЕНИЕ

Рендеринг реалистичных облаков в реальном времени является важной частью научной визуализации, в том числе, метеорологической. Реалистичная визуализация позволяет быстро верифицировать правильность метеорологических симуляций и ускоряет поиск ошибок, которые в них могут возникать.

Также отрисовка облаков в реальном времени является важной частью игровой индустрии. Графика в играх уже достаточно давно достигла фотореализма, однако отрисовка неоднородных сред, в том числе облаков, всё ещё представляет серьёзные трудности. Игры являются комплексными системами, в которых постоянно производится множество различных симуляций, все из которых должны происходить в реальном времени ( $\geq 60$  FPS). В результате, на весь процесс рендеринга одного кадра остаётся крайне небольшой бюджет по времени и вычислительным мощностям.

В то же время, физически корректный рендеринг облаков является ресурсозатратным из-за оптических свойств облаков. Облака практически не поглощают свет в видимом спектре [1], а только рассеивают его. Рассеяние света является наиболее дорогой частью вычисления освещения в физически корректных методах рендеринга, так как требует интегрирования по всем возможным направлениям, в которых свет рассеивается. Более того, облака являются неоднородной средой, а рассеяние света в них подчиняется рассеянию Мie, которое требует большого количества вычислений, и его приходится упрощать, жертвуя качеством рендеринга.

Раньше для убедительной имитации облаков было достаточно сделать многослойный анимированный скайбокс, который хорошо смотрится только издалека. Развитие вычислительных мощностей и технологий теперь позволяет добиваться отрисовки объёмных облачных пейзажей, не отличимых от реальности, но зачастую такие фотореалистичные результаты

либо не работают в реальном времени, либо затратны по памяти.

Целью данной работы является повышение производительности существующего метода рендеринга облаков, при этом не жертвуя качеством рендеринга и не увеличивая затраты по памяти.

В задачи работы входят: изучение существующих методов рендеринга реалистичных облаков в реальном времени, выявление передовых методов и их слабых мест, разработка улучшенной версии метода рендеринга облаков. Ключевыми метриками для оценки степени достигнутых улучшений будут служить FPS для оценки производительности, динамика функции потерь нейросети (loss) для оценки скорости тренировки нейросети и Mean squared error (MSE) для оценки качества отрисовки по сравнению с референсным изображением.

Научная новизна данной работы заключается в двух новых модификациях алгоритма Neural Radiance Caching (NRC), которые не были обнаружены в текущей научной литературе.

Первой модификацией является использование адаптивной фильтрации тренировочных пакетов (batch) с применением многоуровневой прямоугольной сетки. Благодаря этой модификации, «пустые» тренировочные пакеты пропускаются, уменьшая время тренировки в случаях, когда облако занимает не весь экран целиком.

Вторая модификация добавляет нормализованную плотность облака во входные данные нейросети NRC, что сократило время тренировки нейросети примерно в шесть раз.

Дополнительно, в данной работе были проведены эксперименты над новыми методами вычисления коэффициента пропускания из статьи [2]. Результаты экспериментов подтвердили возможность их совместного применения с алгоритмом NRC. Эти методы не только уменьшают дисперсию рендеринга трассировки путей, но и уменьшают значение функции потерь нейросети NRC, увеличивая

качество тренировки нейросети.

Данная диссертация состоит из следующих основных разделов:

- контекст исследования — объяснение важных для понимания основной темы диссертации терминов;
- методы проведения исследования — объяснение того, каким образом проводилось исследование и на каких данных проводились эксперименты;
- улучшения алгоритма NRC для рендеринга облаков — описание новых модификаций алгоритма NRC, предлагаемых в данной диссертации;
- новые методы вычисления коэффициента пропускания — описание методов вычисления коэффициента пропускания на основании маршировки лучей из статьи [2], которые экспериментально использовались совместно с NRC;
- результаты и обсуждение результатов экспериментов;
- планы на будущее — возможные пути развития данной работы и алгоритма NRC.

# 1 КОНТЕКСТ ИССЛЕДОВАНИЯ

## 1.1 Уравнение переноса излучения

Физически корректное вычисление освещения [3] основывается на понятиях радиометрии, а потому освещение вычисляется как энергетическая яркость (radiance). Для того чтобы получить цвет, необходимо домножить энергетическую яркость на цвет источника света.

При прохождении узкого пучка излучения малой площади сквозь среду или при его столкновении с поверхностями, энергетическая яркость излучения меняется за счёт следующих факторов: поглощения излучения средой или поверхностью, испускания излучения средой или поверхностью, а также рассеяния излучения средой или поверхностью. Далее для краткости будем называть такие узкие пучки излучения малой площади лучами.

В свою очередь, рассеяние можно разделить на два вида. Лучи могут рассеиваться из других направлений по направлению рассматриваемого луча и увеличивать его энергетическую яркость (*in-scattering*), а может рассеиваться рассматриваемый луч в различных направлениях, уменьшая энергетическую яркость рассматриваемого луча (*out-scattering*).

Дифференциальная форма уравнения переноса излучения имеет следующий вид:

$$\frac{\partial}{\partial t} L_o(p', \omega) = -\sigma_t(p', \omega)L_i(p', -\omega) + \sigma_t(p', \omega)L_s(p', \omega), \quad (1)$$

где  $p' = p + t\omega$  — точка в направлении  $\omega$  на расстоянии  $t$  от точки  $p$ ;  $L_i$  — входящая энергетическая яркость;  $L_s$  — термин, объединяющий энергетическую яркость испускаемого излучения и входящего рассеяния (*in-scattering*);  $\sigma_t = \sigma_a + \sigma_s$  — коэффициент угасания, соединяющий в себе коэффициенты поглощения  $\sigma_a$  и рассеяния  $\sigma_s$ .

Уравнение (1) состоит из двух слагаемых. Первое слагаемое отвечает за уменьшение энергетической яркости луча, проходящего сквозь среду, в

связи с поглощением энергии средой и рассеяния энергии луча в других направлениях. Второе слагаемое отвечает за увеличение энергетической яркости луча за счёт испускаемого средой излучения и рассеянния излучения из других направлений в направлении рассматриваемого луча.

Термин  $L_s$  из уравнения (1) вычисляется следующим образом:

$$L_s(p, \omega) = \frac{\sigma_a(p, \omega)}{\sigma_t(p, \omega)} L_e(p, \omega) + \frac{\sigma_s(p, \omega)}{\sigma_t(p, \omega)} \int_{S^2} f(p, \omega', \omega) L_i(p, \omega') \partial\omega', \quad (2)$$

где  $L_e$  — энергетическая яркость испускаемого излучения;  $f(p, \omega', \omega)$  — фазовая функция (phase function), которая аппроксимирует функцию плотности вероятности углов, в которых среда рассеивает излучение.

Если проинтегрировать уравнение (1), то получим:

$$L(p, \omega) = \int_0^\infty T_r(p' \rightarrow p) \sigma_t(p', \omega) L_s(p', -\omega) \partial t,$$

где  $T_r(p' \rightarrow p)$  — это коэффициент пропускания (transmittance) на отрезке между  $p$  и  $p'$ , равная:  $T_r(p' \rightarrow p) = \exp(-\int_0^t \sigma_t(p_v, \omega) \partial v)$ .

Облака являются средой, не излучающей свет. Также облака практически не поглощают излучение в видимом спектре [1]. Исходя из этих двух фактов, для вычисления освещения в облаках можно опустить вычисления поглощения и испускания света, и сосредоточиться исключительно на рассеянии света. Такое умозаключение используется во многих алгоритмах рендеринга облаков, обсуждаемых далее.

## 1.2 Трассировка путей

Трассировка путей [4] — это несмешённый (unbiased) алгоритм рендеринга, использующий численный метод Монте-Карло для вычисления уравнения (1).

Алгоритм пускает лучи из каждого пикселя камеры в сцену и строит путь, вершинами которого являются точки рассеяния, где направление

луча меняется. На рисунке 1.1 представлен пример пути, который после нескольких отражений от земли и листвы заканчивается на скайбоксе. В каждой точке пути дополнительно бросаются лучи к солнцу, чтобы посчитать от него прямое освещение. Таким образом, этот путь учитывает освещение от скайбокса и от солнца. Изображение взято из видео How to Build a Real-time Path Tracer от NVIDIA [5].

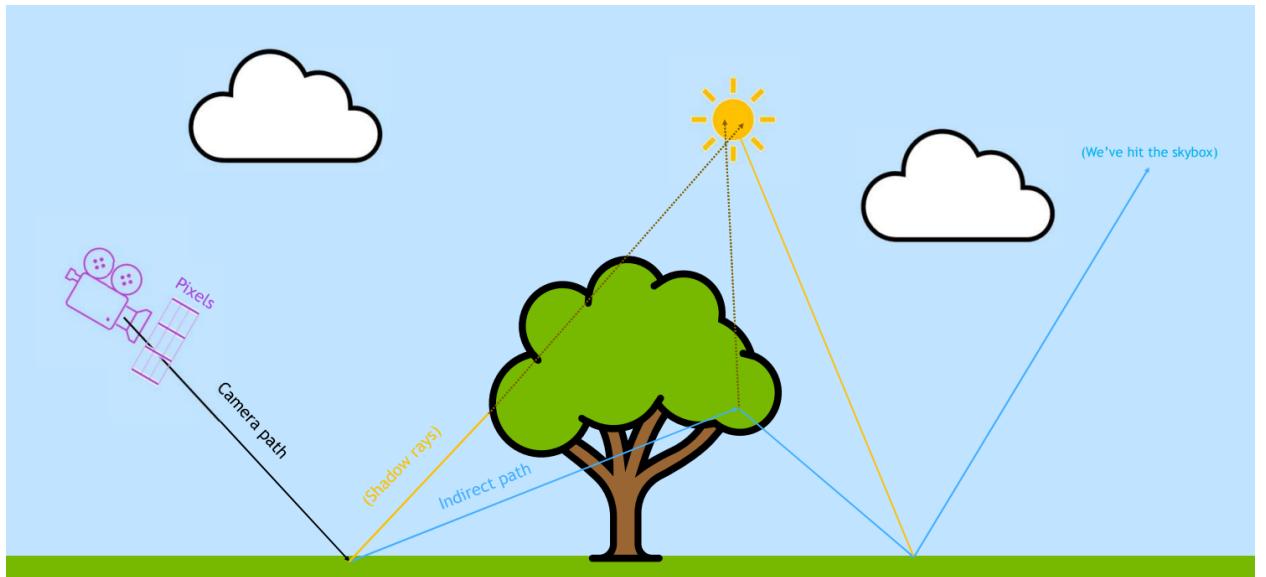


Рисунок 1.1 – Пример одного пути, начинающегося от камеры и заканчивающегося на скайбоксе

Луч рассеивается при столкновении с поверхностями или с частицами среды. Лучи рассеиваются в бесконечном множестве направлений по сфере вокруг точки рассеяния, что соответствует формуле (2). Но трассировка путей, полагаясь на метод Монте-Карло, за раз выбирает только одно направление, в котором продолжится путь луча. Выбор направления после рассеяния является случайным, но чтобы увеличить процент полезных путей, выбор направления пути можно осуществлять, используя *importance sampling* по какой-либо функции. *Importance sampling* — техника для уменьшения дисперсии в вычислениях Монте-Карло за счёт использования вместо случайного распределения распределение по известной функции, которая похожа на интегрируемую функцию. Для рендеринга сред *importance sampling* направлений путей применяют по фазовой функции среды.

Путь заканчивается, когда достигнет источник света или скайбокс, а также когда алгоритм русской рулетки случайно прервёт путь. В зависимости от имплементации могут использоваться и другие эвристики для прерывания путей.

Один такой путь называют пробой (sample), и для того чтобы получить качественное итоговое изображение, на каждый пиксель нужно взять много проб, а также использовать алгоритмы для убиивания шума. Для обозначения числа проб на каждый пиксель в изображении используют термин SPP (Samples per pixel). На рисунке 1.2 и рисунке 1.3 представлены примеры рендеринга при помощи алгоритма трассировки путей с одной пробой на пиксель и с 1024 пробами на пиксель, соответственно.

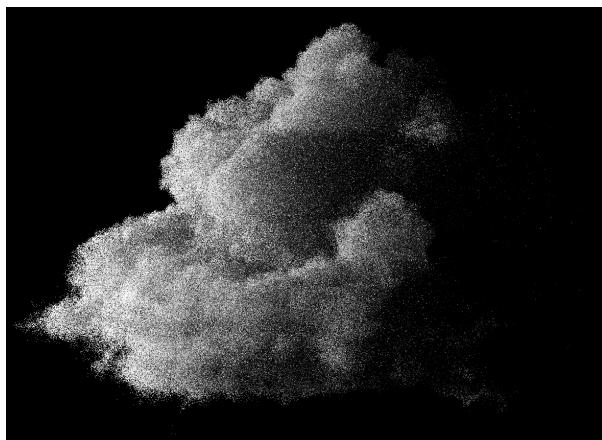


Рисунок 1.2 – 1 SPP



Рисунок 1.3 – 1024 SPP

Далее рассмотрим различные подходы к отрисовке облаков.

## 2 ОБЗОР МЕТОДОВ РЕНДЕРИНГА ОБЛАКОВ И ВЫЧИСЛЕНИЯ РАССЕЯНИЯ

Целью данного литературного обзора является выявление методов рендеринга с применением нейронных сетей, которые могут стать основой для создания алгоритма, способного отрисовывать статичные фотoreалистичные облака в реальном времени.

Алгоритмы рендеринга облаков можно грубо классифицировать на основании структур данных, в которых хранятся облака. Такими структурами данных выступают:

- явные поверхности: полигональные меши, системы частиц, облака точек;
- неявные поверхности, такие как Signed Distance Fields (SDF);
- объёмные сетки
- нейронные представления, где данные об облаке хранятся в виде нейросети.

Сперва рассмотрим подробнее способы отрисовки облаков, заданных явными поверхностями.

### 2.1 Рендеринг облаков, заданных явными поверхностями

#### 2.1.1 Полигональные меши

##### Отрисовка

облаков с помощью полигональных мешей ранее применялась [6], но сейчас крайне устарела, оказавшись непрактичной. Форма реалистичных облаков имеет большое количество мелких и гладких деталей, которые невозможно представить в виде цельных мешей. Это требует слишком большого числа треугольников. Более того, меши плохо поддаются анимации.

### 2.1.2 Системы частиц

Отрисовка облаков с помощью систем частиц в некоторой мере решает проблемы модели полигональных мешей и оно нашло широкое применение [7, Секция 3.3.2.2.] для моделирования и отрисовки облаков. Частицы, с точки зрения рендеринга, обычно представляют в виде billboard прямоугольника, состоящего из двух треугольников. С точки зрения физической симуляции, частицы — это точки в пространстве, с которыми связаны различные симулируемые параметры: позиция, скорость, цвет, и так далее.

С одной стороны, частицы различных размеров и различной плотности позволяют моделировать облака с высоким уровнем детализации, которые относительно просто анимировать с помощью физических симуляций. Эти симуляции могут быть упрощёнными и следовать дизайнерской задумке художника по визуальным эффектам, а могут и основываться на реалистичных физических процессах [8], с помощью которых моделируют формирование и перемещение атмосферных облаков.

С другой стороны, в чистом виде подход к моделированию облаков с помощью системы частиц плохо масштабируется, потому что системы с большим числом частиц крайне ресурсозатратны, что не позволяет использовать их в системах, требующих работу в реальном времени.

### 2.1.3 Облака точек

Есть отдельное большое семейство алгоритмов для 3Д реконструкции [9] объектов или сред по снимкам или измерениям различных сенсоров. Эти снимки и измерения могут быть созданы хоть синтетически по сценам, отрисованным на компьютере, хоть сняты вживую с помощью камер и специальной аппаратуры. Конечная цель 3Д реконструкции получить 3Д сцену, которую можно будет отрисовать как угодно и, желательно, иметь возможность редактировать её.

Один из возможных способов хранения данных о

такой реконструированной сцене является облако точек. Облако точек — это коллекция точек в декартовых координатах, где каждая точка может хранить дополнительные данные вроде цвета или нормали.

Облако точек является довольно гибким представлением сцены, а потому имеет множество различных способов [10], которыми его можно отрисовать.

Одним из самых простых способов отрисовки облака точек является *splatting* [11], где каждая точка представляется в виде примитива, чаще всего, круглого или эллиптического, площадь которого зависит от показателя плотности точки. Направление сплата задаётся нормалью точки.

Существуют различные методы для отрисовки облаков точек при помощи трассировки лучей [12; 13]. Многие из этих методов также опираются на *splatting*, либо вместо лучей используют конусы или цилиндры, так как сами по себе точки и лучи не имеют площади, а потому они не могут пересечься. Методы рендеринга облаков точек, основывающиеся на трассировке лучей, способны показывать фотoreалистичную картинку, но редко достигают скорости работы в реальном времени.

Одним из недостатков облаков точек является требование дополнительного времени на препроцессинг облака точек, чтобы подготовить их к процессу отрисовки. Подготовка может включать в себя такие этапы как генерацию нормалей поверхностей, а также генерацию сплатов различных радиусов.

Облака точек являются хорошим вариантом для хранения данных о поверхностях, однако плохо подходят для сред. Облако точек может дать понимание о границах поверхности, но облака явной поверхности не имеют. Для реалистичной отрисовки облаков важны не столько их границы, которые можно обрисовать облаком точек, сколько их внутреннее строение. Внешний вид облаков достигается за счёт множественного рассеяния, которое происходит внутри облаков, когда свет проходит сквозь них.

Далее рассмотрим методы объёмного рендеринга, решающие проблему рассеяния, обозначенную выше.

## 2.2 Объёмы и трассировка путей

Обычно облака и другие объёмные эффекты и среды представляют в виде объёма, разбитого на воксели. Каждый воксель содержит свойства среды вроде плотности в центре вокселя. Узнать свойства облака между вокселями можно с помощью интерполяции вокселей ближайших к рассматриваемой точке.

Трассировка путей симулирует физически корректное распространение света, с помощью просчитывания пути множества лучей, исходящих из камеры и далее распространяющимся согласно среде распространения и физическим свойствам объектов, с которыми они сталкиваются. Путь разбивается на сегменты, на концах каждого из которых вычисляется изменение в излучении пути на основании уравнения (1.1).

На конце пути получаем финальную энергетическую яркость пробы этого пути. Финальная энергетическая яркость, она же цвет пикселя в линейном пространстве, получается после смешивания энергетических яркостей всех проб в данном пикселе.

Недостатками этого подхода являются большое количество вычислений, требуемое для выполнения алгоритма, а также шумная итоговая картинка если использовать мало проб на пиксель. Путь луча в каждой точке рассеяния может направиться в любом направлении по сфере направлений, однако трассировка путей за один раз выбирает лишь одно направление и продолжает путь в этом направлении. Направление дальнейшего пути в точках рассеяния является случайной величиной, которая обсчитывается методом Монте-Карло. Из-за случайности выбора направлений рассеяния, дисперсия получаемых

вычислений крайне велика. Более того, при вычислении освещения в средах, их коэффициент пропускания также зачастую вычисляется приблизительно с некоторой дисперсией, что ещё больше увеличивает количество шума в отрисованном изображении. Чтобы отрисованное изображение сошлось с истинным значением уравнения (??) и шум в нём перестал быть заметен, требуется огромное количество проб на пиксель. Это могут быть тысячи проб на пиксель, что требует большого времени для рендеринга.

С другой стороны, финальный результат получается фотореалистичным, так как процесс вычисления повинуется достаточно честным законам радиометрии, пускай обычно и немного упрощённым.

Большой объём научных работ в сфере трассировки путей направлен на оптимизацию вычислений и снижение шума при рендеринге с малым числом проб на пиксель. Для этого используются различные оптимизации случайных распределений, из которых берутся пробы направлений рассеянных путей, чтобы концентрировать их в важных направлениях, определяемых физическими свойствами поверхностей и сред. Такая техника называется *importance sampling* [3; 14] и она заключается в том, что если в качестве распределения выборки использовать распределение похожее на истинное распределение, то результат сходится быстрее. Также *importance sampling* может выполняться при помощи нейросетей [15].

В облаках рассеяние происходит согласно фазовой функции Ми, однако её вычисления слишком сложны, поэтому их либо предрассчитывают и сохраняют в больших таблицах [16], либо аппроксимируют другими похожими фазовыми функциями. Самой популярной фазовой функцией для аппроксимации рассеяния Ми является фазовая функция Хенни-Гринштейна [17], так как у неё есть аналитическое решение, которое позволяет выполнять идеальный *importance sampling* по ней, а также она очень проста для вычисления, а потому хорошо подходит для вычислений в реальном времени. Однако фазовая функция Хенни-Гринштейна не точно соответствует фазовой

функции Ми. Есть и другие альтернативы аппроксимирующих фазовых функций, в число которых входят Cornette-Shanks [18], Draine [19], а также смесь методов Хенни-Гринштейна и Draine [20].

Для оптимизации трассировки путей используют методы, чтобы пропускать пустые пространства сцены в трассировке путей, а также мгновенно прекращать пути, которые уходят в пустые участки сцены и точно ни с чем не столкнутся. Это можно делать с помощью ограничивающих объёмов объектов сцены и специальных структур данных, называемых иерархиями ограничивающих объёмов. Другой похожей структурой данных для разделения трёхмерного пространства и поиска по дереву являются октодеревья [21]. Также для этой задачи могут использоваться SDF [22], которые показывают ближайшее расстояние до поверхности объектов.

Другое направление для оптимизаций является кэширование вычислений освещения в специальных структурах данных. Например, для этого используются SD-Trees [23]. Но за эту скорость приходится платить дополнительной памятью для поддержки структур данных.

Передовые решения для рендеринга сред также используют нейронные сети для оптимизации трассировки путей, что подводит нас к рендерингу облаков с применением нейросетей.

### 2.3 Рендеринг с применением нейросетей

Нейросети в рендеринге могут использоваться множеством различных способов, но из всех можно выделить две больших группы применений: нейросети применяются для оптимизации параметров сцены или для оптимизации рендеринга. Начнём со случая, когда нейросеть оптимизирует параметры сцены. Такой подход имеет особый термин: нейронный рендеринг.

### 2.3.1 Нейронный рендеринг

Праородителем нейронного рендеринга является обратный рендеринг [24—27], принадлежащий к методам 3Д реконструкции. Обратный рендеринг использует классическое представление параметров сцены в виде мешей, материалов, объёмов, и так далее. Часть параметров сцены или даже все параметры сцены в обратном рендеринге изначально неизвестны, но известно конечное изображение. Это изображение может быть отрисовано классическими методами рендеринга или может быть фотографией из реальной жизни. Обратный рендеринг по конечному изображению старается восстановить исходные параметры сцены, используя различные специализированные алгоритмы обратного рендеринга или нейросети. Поскольку неизвестных параметров сцены в классическом представлении может быть очень большое количество, то в точности восстановить их из имеющегося ограниченного набора наблюдений является сложной задачей.

Поскольку термин «нейронный рендеринг» является новым, точная его формулировка ещё не устоялась, и часто подвергается изменениям и уточнениям. Согласно обзорной статье Advances In Neural Rendering [28], сейчас под нейронным рендерингом следует понимать алгоритм рендеринга, при котором нейронная сеть учится представлять трёхмерную сцену, а изображение отрисовывается фиксированным дифференцируемым рендер движком. Они также называют это 3Д нейронным рендерингом, чтобы отличать его от более раннего определения нейронного рендеринга, который они

в обзоре называют 2Д нейронным рендерингом. 2Д нейронный рендеринг использует ряд двумерных изображений, полученных, например, с помощью традиционных методов рендеринга, и обучает нейросеть переводить этот ряд входных изображений в одно результирующее референсное изображение.

Если подытожить, в 2Д рендеринге нейросеть учится рендерингу, а в 3Д рендеринге нейросеть учится нейронному представлению сцены, в то время как рендеринг осуществляется фиксированным дифференцируемым способом. Сравнение 2Д и 3Д нейронного рендеринга из статьи Advances In Neural Rendering представлено на рисунках 2.1–2.2.

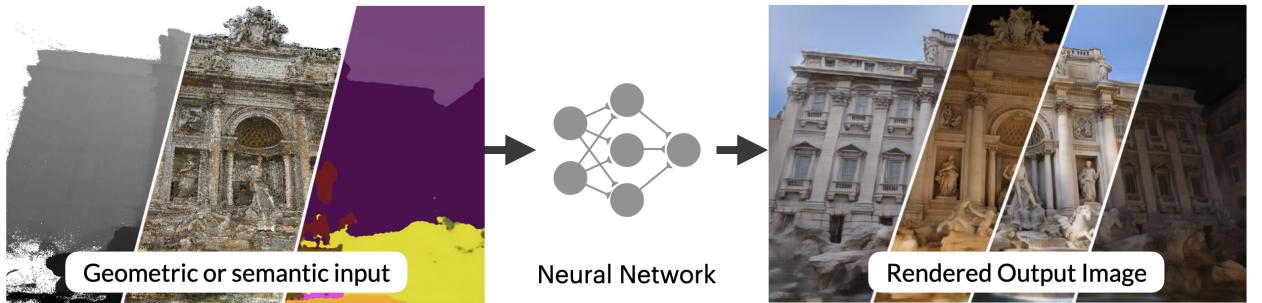


Рисунок 2.1 – 2D нейронный рендеринг. Изображение взято из статьи Advances In Neural Rendering [28]

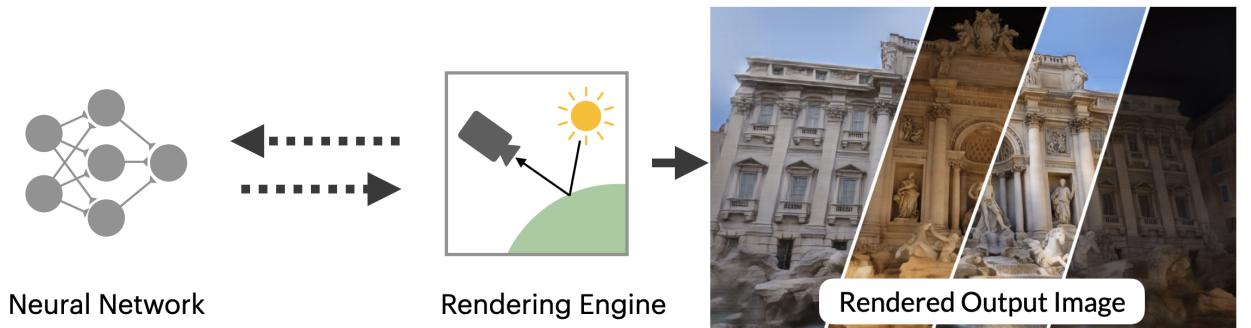


Рисунок 2.2 – 3D нейронный рендеринг. Изображение взято из статьи Advances In Neural Rendering [28]

Появление алгоритма Neural Radiance Fields (NeRF) [29] сильно толкнуло область нейронного рендеринга вперёд. Эта работа представила удобный способ нейронного представления сцены в виде нейросети, метод тренировки этой нейросети и способ отрисовки результатов.

На рисунке 2.3 представлена визуализация того, как работает алгоритм NeRF. Нейросеть принимает на вход позицию и направление луча, задаваемые ray marching алгоритмом, а на выход выводит плотность и цвет поверхности. Для нейросети собирают датасет фотографий, показывающих статичную

сцену со всех сторон. На этом датасете нейросеть выучивает и сохраняет в нейросети сцену, используя фиксированный дифференцируемый объёмный рендерер и обратное распространение ошибки между фотографиями из датасета полученными результатами тренировочной отрисовки. После обучения камеру можно свободно перемещать по сцене и получать рендеры сцены с новых ракурсов, которых не было в изначальном датасете.

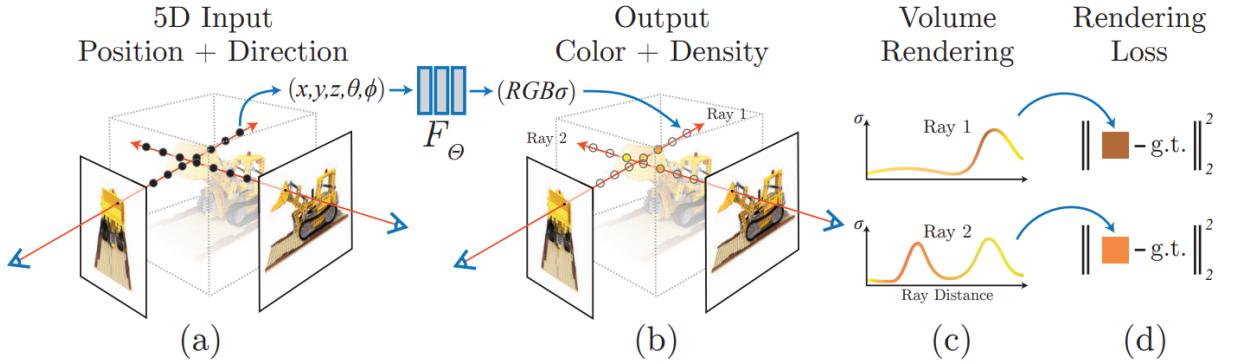


Рисунок 2.3 – Визуализация алгоритма работы NeRF. Изображение взято из оригинальной статьи по NeRF [29]

Во время маршировки лучей каждая проба на луче берётся из нейросети вместо использования явного объёмного представления сцены. Благодаря этому явное представление объёма сжимается до размеров нейронной сети (5МБ для NeRF). Оригинальная имплементация алгоритма NeRF добилась фотореалистичного рендеринга, но тренировка нейронной сети занимала два дня и рендеринг одного кадра занимал 30 секунд на видеокарте NVIDIA V100. Такая низкая производительность обусловлена достаточно наивной имплементацией: лишние лучи никак не отсекаются и могут уходить в пустые пространства сцены; каждая проба на луче получается за счёт обращения в нейронную сеть, при том что каждое обращение является очень дорогим.

У NeRF также есть недостаток, что он подходит только для статичных сцен, которые он детально запоминает. Ничего кроме положения камеры в изученных сценах изменить не удастся. Не получится передвинуть объекты или изменить освещение в сцене. Также этот алгоритм плохо подходит для

представления сред вроде облаков. NeRF может запомнить внешний вид облака с различных сторон, но он не может изучить внутреннее строение облака и его воздействие на проходящий через облако свет, а потому внешний вид облаков с ракурсов, которых не было в тренировочном датасете скорее всего будет не совсем корректен.

Большое число работ, основывающихся на NeRF, работают над улучшением различных его недостатков. В первую очередь, над производительностью.

В статье Automatic integration for fast neural volume rendering [30] представлен способ конструировать интегральные и производные нейронные сети, с помощью которых можно вычислить определённый интеграл всего луча, используя всего два обращения в нейронную сеть на концах отрезка, ограничивающего луч, в согласии с основной теоремой математического анализа.

DONeRF [31] работает существенно быстрее, чем NeRF. Этот алгоритм использует дополнительную нейронную сеть классификатор, предсказывающую вероятность глубины поверхностей вдоль рассматриваемого луча. Пробы вдоль луча распределяются логарифмически по полученной функции распределения классифицированной глубины. Чем выше вероятность, тем больше выделяется проб, при этом ближайшие объекты получают больше проб. Также авторы этой работы берут пробы на луче в радиально искажённом пространстве, чтобы сузить размер сцены и ослабить высокие частоты фона на больших сценах. Однако, пробирование за счёт глубины слабо справляется с прозрачными и отражающими поверхностями.

fV-SRN [32] для предсказания плотности объёма использует маленькую нейронную сеть, помещающуюся в разделяемую память GPU, в комбинации с сеткой небольшого разрешения, хранящей дополнительные латентные данные для нейронной сети. За счёт того что нейронная сеть маленькая, и

обращается только в разделяемую память GPU, она работает существенно быстрее, но ей не хватает ёмкости для точных вычислений. Сетка хранит промежуточные данные в пространстве и восполняет недостаток выразительной мощности нейронной сети. Нейронная сеть и латентная сетка тренируются вместе. Благодаря маленькому размеру нейронной сети и маленькому разрешению сетки с небольшим числом латентных данных, такое представление объёма остаётся компактным и требует памяти не больше, чем NeRF.

В работе Instant Neural Graphics Primitives With Multiresolution Hash Encoding [33] представили многоуровневую хэш воксельную сетку, которая по хэшу пространственных координат извлекает из сетки feature vector на каждом уровне хэш воксельной сетки и конкатенирует их вместе с остальными входными данными кодирующих нейронных сетей. Хэш сетка тренируется вместе с основной нейронной сетью. Сетка имеет многоуровневую структуру. Размер хэш сетки с каждым уровнем становится меньше, и если на крупной сетке хэш коллизий не возникает, то на мелкой они могут становиться частыми. Однако авторы работы заявляют, что сетка при коллизиях по размеру ошибки автоматически оптимизирует наиболее важные параметры, и коллизии не ухудшают качество результатов. При использовании других оптимизаций, включая Fully fused networks, описанных ранее, и оптимизаций, зависящих от задачи, авторы демонстрируют многократный прирост производительности в тренировке и рендеринге при применении к ряду алгоритмов, включающих NRC [34] и NeRF [29]. Например, тренировка NeRF ускоряется всего лишь до 5 минут, а NRC начинает тренироваться онлайн и рендерить в реальном времени в Full HD разрешении.

Нейросети в рендеринге используются не только для предсказания исходных параметров сцены или нейронного представления. Есть также алгоритмы прямого рендеринга, которые используют традиционные методы

хранения и отрисовки сцены, но дополнительно применяют нейронные сети для аппроксимации некоторых сложных вычислений вроде входящего рассеяния (in-scattering) света. Такие алгоритмы не относятся к нейронному рендерингу. Правильнее будет называть их традиционными методами рендеринга с применением нейросетей.

### 2.3.2 Объёмный рендеринг с применением нейронных сетей

Чтобы добиться работы в реальном времени, передовые решения [35; 36] для отрисовки объёмов используют мало проб на пиксель в трассировке путей и применяют нейронные сети для фильтрации шума [37; 38] в итоговой картинке.

Помимо убираания шума, нейросеть в статье Interactive Path Tracing and Reconstruction of Sparse Volumes [36] используется для адаптивного нейронного сэмплинга, распределяющего нагрузку по количеству проб для каждого пикселя изображения. Адаптивный сэмплинг заставляет трассировщик путей использовать больше проб на пиксель в тех пикселях, где они наиболее нужны. Этот алгоритм позволяет достичь существенного прироста в качестве рендеринга трассировки путей, однако за это приходится платить производительностью, потому что больше путей начинает проходить через самые сложные части среды, вычисления которых наиболее ресурсозатратны.

Алгоритм Neural Radiance Caching (NRC) [34] использует нейронную сеть вместе с обычным трассировщиком путей и обучает нейросеть предсказывать освещение на поверхности объекта по трёхмерным координатам и параметрам поверхности в этих координатах. Этот алгоритм был выбран в качестве базового алгоритма для рендеринга облаков, а потому следует рассмотреть его подробнее в следующем подразделе.

## 2.4 Neural Radiance Caching

Вычислять длинные пути в трассировщике путей дорого, а в случае с облаками, большинство путей в трассировщике путей являются длинными, так как облака практически не поглощают свет, а только рассеивают, изменяя траекторию пути. В результате, пути, проходящие через облако редко получают раннее прерывание. В основном, это происходит либо в результате того, что путь покидает облако и попадает в скайбокс, либо в результате случайного прерывания алгоритмом русской рулетки.

Алгоритм NRC использует нейронную сеть для кэширования вычислений трассировщика путей и сокращения длины путей во время инференса. Нейросеть обучаются онлайн на результатах работы трассировщика путей на удлинённых путях. Рендеринг выполняют на коротких путях, а на конце пути остаточную энергетическую яркость предсказывают натренированной нейросетью. Поскольку просчёты длинных путей является очень ресурсозатратным, такой подход снижает вычислительную нагрузку.

Для рендерера не важно, было ли освещение рассчитано коротким путём с использованием нейронного кэша или длинным путём стандартного трассировщика путей. Обучение нейросети выполняется онлайн. Это означает, что оно делается без предварительной тренировки, прямо во время рендеринга. Из-за этого NRC требуется небольшое время на «прогрев», во время которого NRC будет выдавать плохие предсказания энергетической яркости и тренироваться. У авторов статьи процесс обучения нейросети для рендеринга сложных сцен с различными поверхностями занимает всего около восьми кадров, что при отрисовке в 60 FPS (Frames per second) едва заметно.

#### 2.4.1 Входные данные нейросети NRC и их сбор

Тренировка нейросети NRC осуществляется на данных о лучах и о поверхностях, с которыми они сталкиваются. Выходным значением нейросети является цвет пути.

Сбор данных для тренировки NRC осуществляется следующим способом. Сначала трассировщик путей рассматривает короткие пути, и в конечной точке пути сохраняет позицию и направление луча. Далее используя эти данные отдельно подготавливаются данные для тренировки и для инференса NRC.

Для тренировки короткие пути продолжают с того места где остановились, удлиняя путь. В конце удлинённого пути, позицию и направление луча, а также данные о поверхности, с которой луч столкнулся в конце пути, сохраняют в буффер для тренировки нейросетью. Данные о поверхности включают: нормаль луча к поверхности, коэффициент шероховатости, диффузный коэффициент отражения и зеркальный коэффициент отражения.

Для инференса данные о поверхности аналогично получают из поверхности, с которой столкнулся короткий, неудлинённый луч.

Авторы статьи замечают, что нейросети лучше обучаются, когда корреляция между входными и выходными данными линейна. Для диффузного и зеркального коэффициента отражения это истинно по-умолчанию, поэтому с ними не требуется выполнять преобразений, в отличие от остальных четырёх входных параметров.

Для направления, нормали и коэффициента шероховатости они используют one-blob encoding [15]. Он хорошо работает, когда небольшое изменение во входных значениях слабо влияет на выходное значение. Однако, позиции этому требованию не соответствуют, а потому для них

используется частотное кодирование [39], которое впервые использовалось для обучения нейросети NeRF [29]. В NRC авторы работы использовали только синусную часть кодирования, отбросив косинусы, которые особо не влияли на результат.

Для рендеринга облаков данные о поверхностях не нужны, так как их нет в сцене, но если их убрать и оставить только данные о позициях и направлениях лучей, то скорость тренировки NRC сильно падает. Далее по тексту будет описано, как была решена эта проблема.

#### 2.4.2 Архитектура нейронной сети

Авторы статьи разработали низкоуровневую архитектуру нейронной сети, которую они назвали Fully fused MLP (Multi Layer Perceptron). Благодаря новой архитектуре им удалось добиться рендеринга в реальном времени при использовании одной пробы на пиксель в трассировщике путей.

Эта архитектура позволяет помещать нейросеть целиком в быструю память GPU, находящуюся на чипе (регистры и L1 кэш), и избегать лишних обращений в медленную глобальную память GPU во время тренировки и инференса. Матрица весов нейросети помещается в регистры GPU, а промежуточные активации слоёв в общую память (shared memory) GPU. Поскольку памяти на чипе видеокарты мало, то и размер нейросети не получится сделать большим. Авторы статьи используют 7 полносвязных (fully connected) слоёв, 5 из которых являются скрытыми (hidden) и состоят из 64 нейронов.

На рисунке 2.4 представлена архитектура Fully fused MLP. (а) В нейросеть подаются большие пакеты для инференса. (б) Большие пакеты дробятся на маленькие сегменты по 128 элементов. Каждый элемент обрабатывается отдельной группой потоков GPU. (в) Внутри группы потоков построчно перемножается матрица весов  $W_i$  с матрицей активаций  $H_i$ , чтобы вычислить новую строку в следующей матрице активаций  $H'_{i+1}$ .

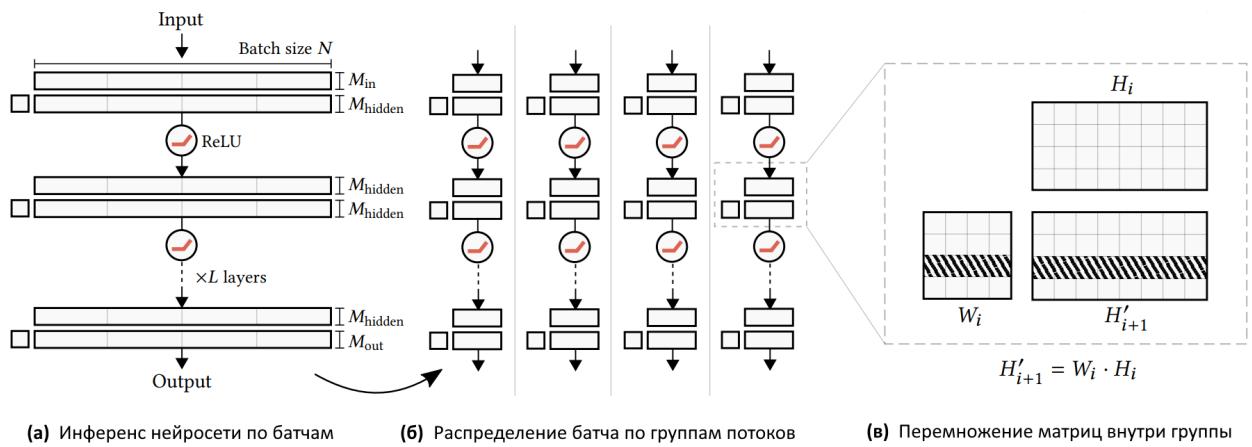


Рисунок 2.4 – Архитектура Fully fused MLP. Изображение взято из статьи об NRC [34]

По данным из статьи, Fully fused архитектура для тренировки и инференса NRC работает примерно в 5–10 раз быстрее, чем TensorFlow 2.5.0 с XLA (рисунок 2.5). На левом графике рисунка сравнивается производительность тренировки, а на правом графике — инференс NRC на пакетах различных размеров. Жирной линией отмечен результат для нейросети шириной в 64 нейрона, а штрихованной для нейросети шириной в 128 нейронов.

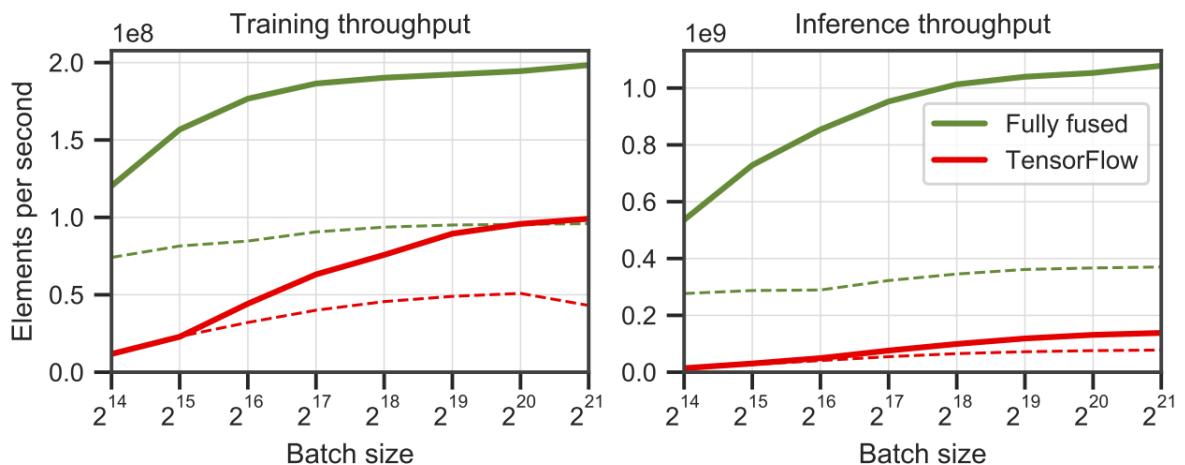


Рисунок 2.5 – Сравнение производительности Fully fused MLP и Tensorflow 2.5.0 с XLA. Изображение взято из статьи об NRC [34]

### **3 МЕТОДЫ ПРОВЕДЕНИЯ ИССЛЕДОВАНИЯ**

#### **3.1 Базовая имплементация алгоритма NRC**

В качестве базовой имплементации алгоритма NRC была взята работа Яна Спиндлера, доступная на GitHub [40] вместе с текстом его бакалаврского диплома. Далее по тексту имплементация Яна Спиндлера будет называться «базовой» для краткости.

Был сделан форк, который также доступен на GitHub [41], где содержится код всех описанных в данной работе модификаций над базовой имплементацией и код для анализа результатов экспериментов.

В базовой имплементации NRC уже реализованы некоторые улучшения, которые можно применить к алгоритму NRC. В число таких входит применение Multi Resolution Hash Encoding [33] для кодирования позиций лучей во входных данных для нейросети. Как указано в работе Яна Спиндлера, Multi Resolution Hash Encoding для кодирования позиций работает медленнее, чем частотное кодирование, так как выполняется с помощью отдельной кодирующей нейросети, с другой стороны, уменьшает количество шума в итоговом изображении.

#### **3.2 Датасет облаков**

В качестве тестового облака для всех экспериментов использовалось облако от компании Disney [42] в формате VDB в четвертичном разрешении.

#### **3.3 Экспериментальные метрики**

В зависимости от производимых модификаций, использовались различные метрики для измерения качества и производительности модификаций:

- для измерения скорости тренировки нейросети использовалось время

в миллисекундах;

- для измерения качества тренировки нейросети использовалась метрика относительной функции потерь (Relative L2 Loss);
- для измерения количества шума в изображении использовалась метрика относительной дисперсии (Relative Variance);
- для измерения величины смещённости результатов нейросети использовалась метрика относительного смещения (Relative Bias).

Все измерения метрик проводились с одной пробой на пиксель. Трассировка путей в таком случае выдаёт шумный результат из-за того, что многое в алгоритме трассировки путей зависит от генератора случайных чисел. Чтобы этот шум не помешал сравнению метрик при различных модификациях, все качественные метрики были усреднены между множеством повторных запусков экспериментов.

### 3.4 Системные характеристики

Эксперименты проводились на компьютере со следующими системными характеристиками:

- видеокарта RTX 3080 с 10 гигабайтами видео памяти;
- процессор Ryzen 7 2700;
- 32 гигабайта оперативной памяти с частотой 2400 МГц;
- операционная система Windows 11.

## 4 УЛУЧШЕНИЯ АЛГОРИТМА NRC ДЛЯ РЕНДЕРИНГА ОБЛАКОВ

В этой секции описаны модификации над базовой имплементацией алгоритма NRC. Результаты и обсуждение результатов экспериментов над этими модификациями находятся далее в секции «Результаты и обсуждение».

### 4.1 Дополнительные входные данные об облаке

Для того чтобы тренировка нейросети NRC быстро сходилась, ей требуются данные о сцене. Авторы оригинальной статьи делали NRC под рендеринг поверхностей, а потому в их статье на вход нейросети помимо данных о луче поступают данные о поверхностях, с которыми в конце пути сталкивается луч. Для рендеринга облаков эти входные данные бесполезны, и их нужно заменить данными об облаке. Авторы оригинальной статьи опробовали NRC на рендеринге объёмов, заменив ненужные входные данные о поверхностях константами, но никаких входных данных об объёме нейросети не предоставили. В реализации NRC Яна Спиндлера входные данные о поверхностях вовсе удалены, но данных об облаках он тоже в нейросеть никаких не предоставил.

Поскольку NRC тренируется онлайн прямо во время рендеринга, то пока NRC не натренируется достаточно, он сильно искажает цвета пикселей изображения во время рендеринга. В оригинальной статье написано, что нейронная сеть NRC уже на восьмом кадре показывала хорошие результаты, на которых особо не видно искажений цветов от недостатка тренировки. При попытке применить NRC напрямую к рендерингу облаков без предоставления дополнительных данных нейросети об облаке, сильные искажения цвета стали исчезать заметно дольше. Только на 64м кадре цветовые искажения становятся не заметны, как видно на рисунке 4.1. Рисунок содержит последовательность снимков затенённой части облака в

различные моменты времени. Цветовые искажения наиболее заметны в той части облака, которая находится в тени.

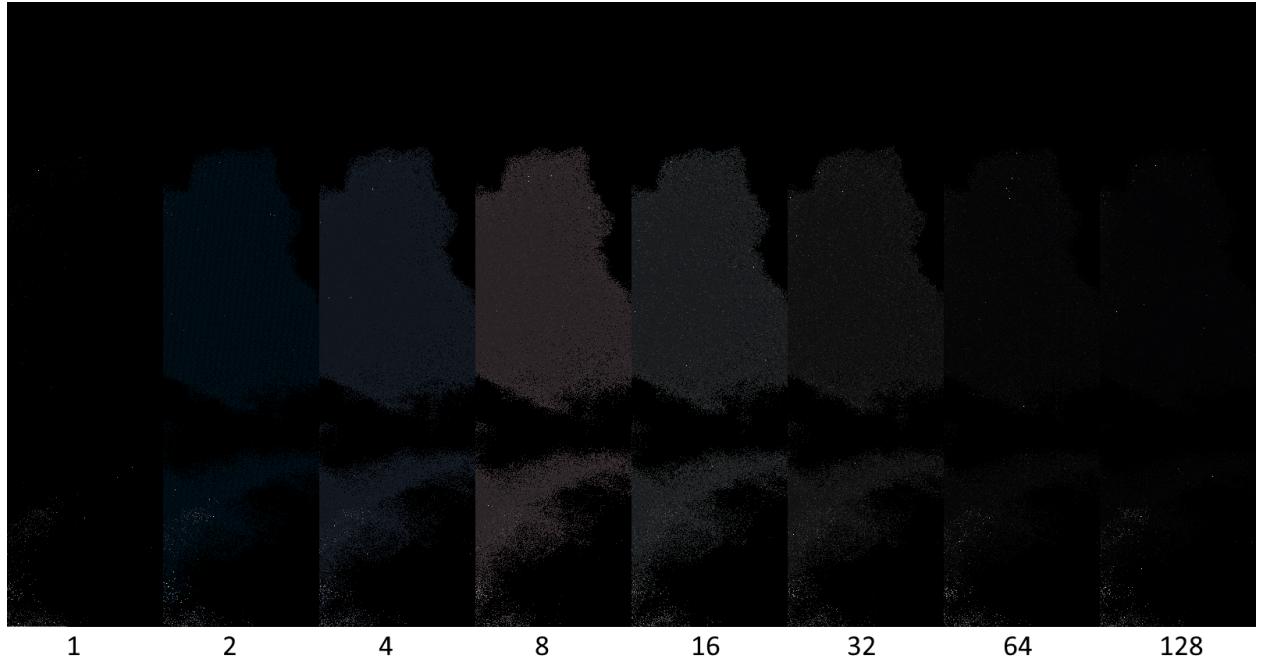


Рисунок 4.1 – Прогресс тренировки для NRC со входными данными:  
позиция+направление

Чтобы разрешить эту проблему, был проведён ряд экспериментов, которые определили наилучший вариант среди следующих кандидатов:

- плотность облака в финальной точке пути;
- градиент облака в финальной точке пути;
- результат применения оператора Лапласа к плотности в финальной точке пути.

Градиент и оператор Лапласа применялись как методы предварительной трансформации плотности облака. Само облако от компании Дисней [42], на котором производились все тесты, изначально хранит в себе только плотность облака в формате VDB.

**Сравнение**  
кандидатов проводились с использованием метрики функции потерь (loss) тренировки нейросети. В качестве функции потерь был использован Relative

L2 Loss [43], который при этом нормализуется не на предсказание нейросети, как обычный Relative L2 Loss, а на воспринимаемую яркость (perceived luminance) предсказания нейросети, как и в оригинальной статье по NRC [34]. Relative L2 Loss вычисляется по следующей формуле:

$$L2(L_s, \hat{L}_s) = \frac{(L_s - \hat{L}_s)^2}{luminance(\hat{L}_s) + \epsilon},$$

где  $L_s$  — это результат вычисления энергетической яркости длинного пути трассировщиком путей;  $\hat{L}_s$  — предсказание нейросети;  $\epsilon = 0.01$  — константа.

Воспринимаемая яркость вычисляется по стандартной формуле:

$$luminance(x) = (0.299, 0.587, 0.114)^T x,$$

где  $x$  — это RGB цвет

## 4.2 Фильтрация тренировочных пакетов с помощью многоуровневой прямоугольной сетки

У NRC есть проблема, которую в том числе отмечают оригинальные авторы работы по NRC. NRC постоянно перетренировывает нейросеть на всём изображении на всех входных данных. При том, что NRC используется для инференса только для подсчёта непрямого освещения, после хотя бы одного рассеяния, нет смысла тренировать нейросеть, если в каком-то пакете рассеяние не происходит. Поскольку облака висят в небе и вокруг них находится скайбокс, то таких «пустых» пространств, в которых рассеяния не происходят может быть довольно много. Оригинальный NRC продолжает тренировать нейросеть на всех пакетах данных, даже если облако не попадает в некоторые из пакетов, и даже, если облако не попадает в кадр совсем.

Ян Спиндлер в своей работе [40] тоже обратил внимание на аналогичную проблему при инференсе нейросети и стал исключать из инференса пакеты, если ни в одном из пикселей пакета не произошло

рассеяние. Тем не менее, по какой-то причине с тренировочными пакетами он не стал использовать эту оптимизацию.

Модификация для фильтрации тренировочных пакетов, представленная в этом подразделе развивает идею метода фильтрации инференсных пакетов Яна Спиндлера.

Входные данные для нейросети можно распределить по пакетам множеством различных способов. Самым простым вариантом, который в том числе использовал Ян Спиндлер, является разбиение изображения на полосы, где каждая полоса представляет отдельный пакет что изображено на рисунке 4.4. Если хотя бы один пиксель в полосе имеет событие рассеяния, то весь пакет требуется использовать в тренировке. В результате, пакеты практически не отбрасываются, если облако по горизонтали на изображении занимает много места, а по вертикали мало. Вместо этого было применено разделение тренировочного изображения на пакеты равномерной прямоугольной сеткой, а инференсного изображения равномерной квадратной сеткой. Это изменение призвано повысить количество ситуаций, когда часть пакетов может быть отброшена.

У фильтрации пакетов есть ограничение в том, что она приносит пользу только когда данные разбиты на много пакетов, но в таком случае тренировка сильно замедляется. Тренировка проходит быстрее, если использовать мало больших пакетов в рамках ограничений по shared memory, чем если использовать много маленьких, по множеству раз проходя через весь процесс тренировки за один кадр.

Чтобы сгладить это ограничение, использовалось следующее свойство равномерной прямоугольной сетки. Каждый прямоугольник в сетке можно поделить на 4 более мелких равных друг другу по размерам прямоугольника, что позволяет делать вложенную многоуровневую сетку. Если считать ячейку сетки вершиной, а переход на более вложенную ячейку сетки считать ребром, то эту вложенную структуру можно представить в виде

четвертичного дерева. Корнем дерева является один пакет, содержащий всё тренировочное изображение целиком, а каждый последующий ребёнок в дереве представляет одну четвёртую от родительского пакета. С помощью такой структуры данных можно быстро определить, какие мелкие пакеты из листов можно объединить в более крупные родительские пакеты, чтобы использовать небольшое число вершин на разных уровнях дерева для тренировки вместо всего множества листов.

Рисунок 4.2 визуально демонстрирует идею разбиения тренировочного изображения на многоуровневую прямоугольную сетку, состоящую из трёх уровней. Синим цветом обозначен нулевой уровень (1 большой пакет), зелёным первый (4 средних пакета), а оранжевым второй (16 мелких пакетов). Четыре закрашенных прямоугольника обозначают оптимальную выборку пакетов для тренировки.

Рисунок 4.3 показывает, как выглядит разбиение изображения с рисунка 4.2 с алгоритмической точки зрения в виде четвертичного дерева. Из трассировщика путей поступает большой тренировочный буффер  $T$ , который разбивается на более мелкие пакеты, размер которых зависит от уровня в дереве. Каждая вершина дерева содержит ссылку на ячейку в массиве  $T$  вида  $B_{ij}$ , где  $i$  — это уровень дерева, а  $j$  — это индекс пакета внутри уровня. Дерево соответствует разбиению на рисунке 4.2. Вершина, заполненная цветом показывает, что в ней и во всех её потомках произошли рассеяния. Четыре красных прямоугольника обозначают оптимальную выборку пакетов для тренировки.

Рисунок в виде дерева наглядно демонстрирует прирост производительности, который предоставляет адаптивная многоуровневая сетка. Если бы тренировочные данные  $T$  были разделены на пакеты без многоуровневости, с тем же размером сетки, что и у оранжевого уровня, то для тренировки потребовалось бы использовать десять заполненных оранжевых вершин, что в два с половиной раза больше вызовов тренировки

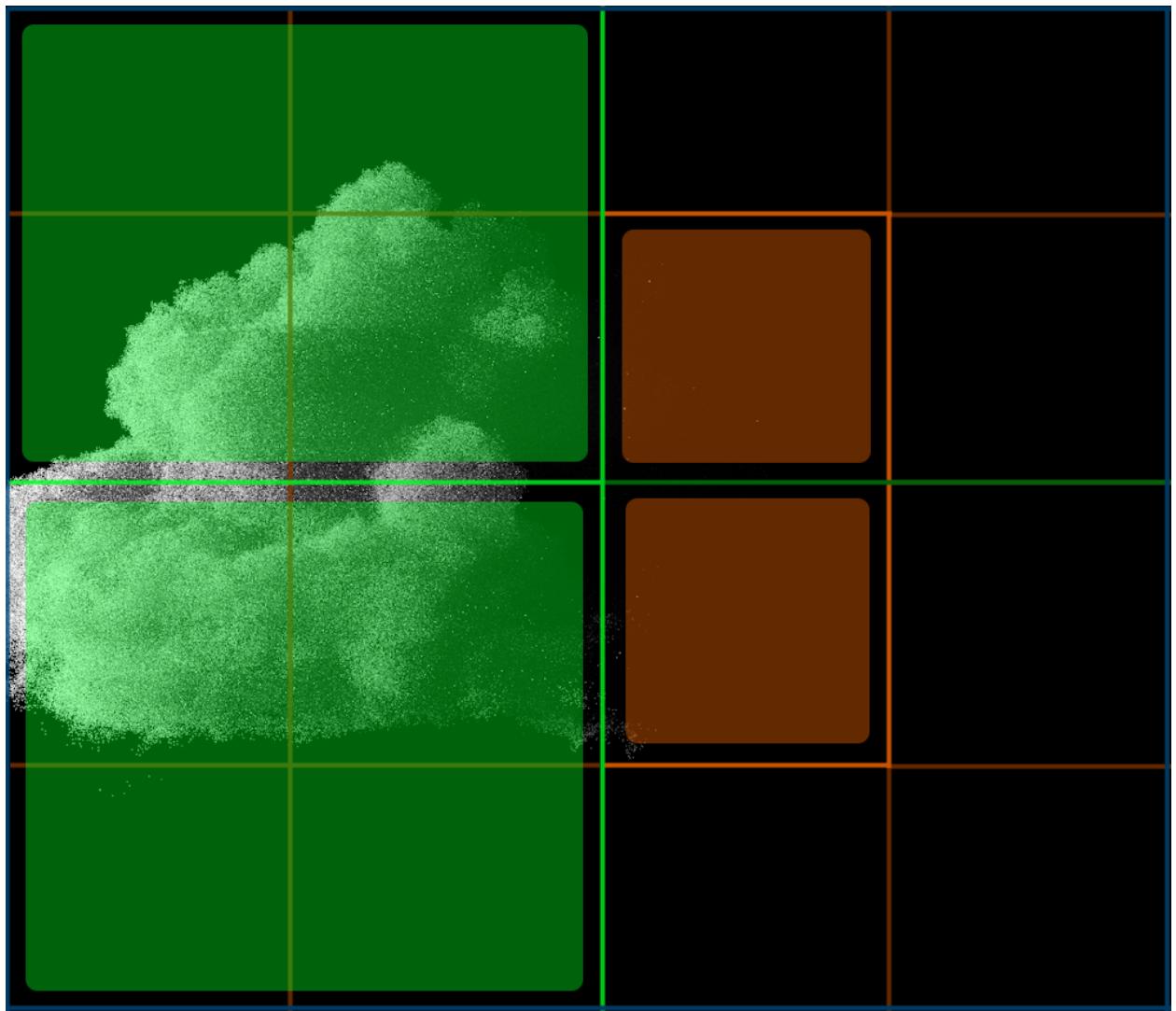


Рисунок 4.2 – Разбиение тренировочного изображения на пакеты в виде многоуровневой прямоугольной сетки

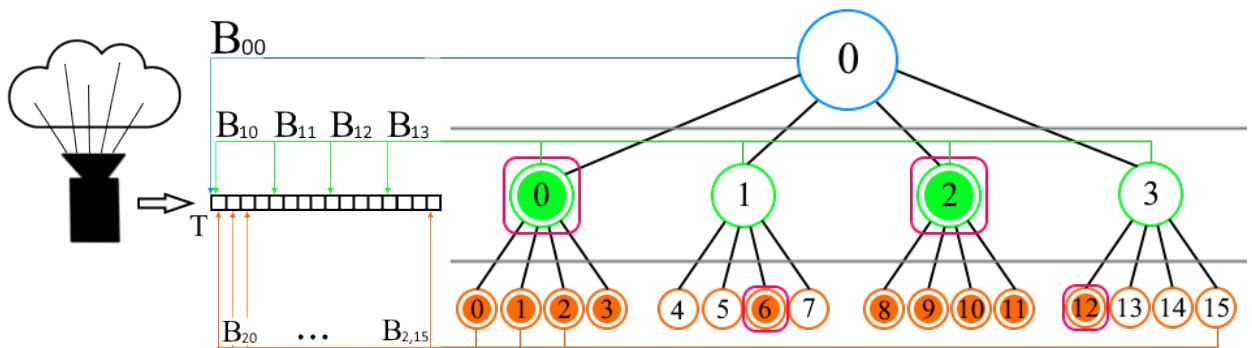


Рисунок 4.3 – Разбиение тренировочного буфера на пакеты в виде четвертичного дерева

нейросети, чем с оптимальной выборкой.

Опишем алгоритм поиска оптимального набора пакетов в четвертичном дереве пакетов. Изначально данные о рассеяниях

доступны только в листах дерева. Поскольку более высокие уровни являются «виртуальными» и лишь объединяют в себе информацию, которая и так доступна в листах, нет смысла вычислять наличие рассеяний на каждый уровень отдельно — достаточно посчитать их в листах, а далее во время поиска воспользоваться информацией из листов, чтобы сделать выводы о всех вершинах дерева. Дерево обходится рекурсивно, начиная с корневой вершины, по следующим правилам:

1. Если данные о рассеянии в вершине неизвестны, то смотрим её четырёх детей,
2. Если ни один из четырех детей не содержит событий рассеяния, то родительская вершина не имеет рассеяний и отбрасывается,
3. Если все четыре ребёнка содержат события рассеяния, то родительская вершина помечается как имеющая событие рассеяния,
4. Если хотя бы один, но меньше четырёх детей имеют рассеяния, то в список пакетов на тренировку добавляются все дети, в которых произошли рассеяния, а родительская вершина помечается как не имеющая рассеяния
5. Если в вершине произошло рассеяние и она является корнем, то в список пакетов на тренировку добавляется корень.

Вы можете ознакомиться с псевдокодом алгоритма поиска оптимальных пакетов *GetBatchesToTrain* в Приложении А.

Поскольку инференсное изображение привязано к разрешению рендеринга, в отличие от тренировочного, его размеры трудно подобрать под размеры пакетов, чтобы достичь многоуровневого разделения как в тренировке. Поэтому для инференса используется одноуровневое

разделение на одинаковые квадратные пакеты, помимо угловых пакетов, которые обрезаются границами изображения.

Рисунок 4.5 демонстрирует, как работает разбиение инференсного изображения в разрешении 1920x1080 пикселей на квадратные пакеты размером 512x512 пикселей. Верхний ряд и правый столбец пакетов обрезаются границами изображения, а потому эти пограничные пакеты имеют меньший нестандартный размер.

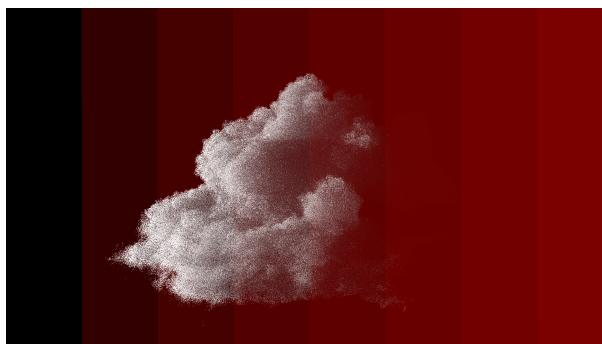


Рисунок 4.4 – Разбиение инференсного изображения на 8 пакетов вертикальными полосами



Рисунок 4.5 – Разбиение инференсного изображения на 12 квадратных пакетов

## 5 НОВЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ КОЭФФИЦИЕНТА ПРОПУСКАНИЯ

Статья An unbiased ray-marching transmittance estimator [2] предоставляет два новых метода, основывающихся на маршировке лучей (raymarching), для численного приближения коэффициента пропускания сред. Были реализованы оба новых метода и проведено сравнение их между собой, а также с алгоритмом вычисления коэффициента пропускания Ratio Tracking [44], который использовался в базовой имплементации NRC.

Первый метод основывается на разложении оценки коэффициента пропускания в степенной ряд Тейлора [45]. Нулевой элемент ряда вычисляется при помощи алгоритма jittered raymarching, а последующие элементы ряда вычисляются с некоторой вероятностью и используются как средства корректировки нулевого элемента, который вносит наибольший вклад в финальную оценку. Эта вероятность прервать ряд в любой момент представляет из себя алгоритм русской рулетки и делает метод несмещённым (unbiased).

Первый метод использует множество различных оптимизаций, чтобы сделать вычисление ряда быстрым и эффективным в плане снижения дисперсия оценки каждым из элементов ряда. Эти оптимизации включают использование U-статистики [46] для симметризации элементов ряда и снижения дисперсии, русскую рулетку с высокой вероятностью прерывания на первых элементах ряда, и другие.

Авторы статьи замечают, что поскольку нулевой элемент степенного ряда имеет наибольшее влияние на финальную оценку, то можно вычислять только его, отбросив остальной ряд, и вложив освободившиеся вычислительное время в увеличение числа проб в этом элементе. Именно таким образом работает второй метод, который описан в их работе, и он является смещённым (biased). С другой стороны, пусть он и добавляет сдвиг

в оценку, второй метод имеет меньшую дисперсию, а следовательно лучшее финальное качество изображения.

Второй метод является намного более простым, так как он отбрасывает большинство сложных оптимизаций, которые использует первый метод, но при этом имеет результат с меньшим количеством шума, что делает его более привлекательным для использования.

В следующей секции данной диссертации будут представлены результаты обоих методов и приведено их сравнение с базовым алгоритмом Ratio Tracking.

## 6 РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

В этой секции представлены результаты экспериментов над описанными ранее модификациями NRC и новыми методами вычисления коэффициента пропускания, а также анализ этих результатов.

### 6.1 Дополнительные входные данные об облаке

Был проведён эксперимент, где в качестве третьего элемента входных данных в нейросеть использовались различные данные об облаке: плотность, градиент плотности, и лапласиан плотности. На рисунке 6.1 приведено сравнение динамики функции потерь при добавлении каждого из этих испытуемых в качестве третьего элемента входных данных для нейросети. Поскольку плотность облака и её трансформации (градиент и лапласиан) показали практически идентичные результаты, то в качестве наилучшего варианта для третьего элемента входных данных была выбрана плотность облака.

Кодирование к плотности и её трансформациям применять не требуется, так как они итак имеют нормализованные значения, а при добавлении кодирования, получаемые результаты существенно не изменялись.

Добавление плотности облака во входные данные значительно ускорило тренировку нейросети, при этом никак не ухудшив производительность. Как видно на рисунке 6.2, после этого изменения явные искажения цветов во время тренировки стали пропадать примерно на 32м кадре, что при скорости работы в 50 FPS визуально выглядит намного лучше, чем изначальный результат, где искажения пропадали только на 64м кадре (рисунок 4.1). Уже на 8м кадре облако принимает довольно естественный серый цвет и плавно затемняется, что внешне похоже на визуальный эффект адаптации зрения.

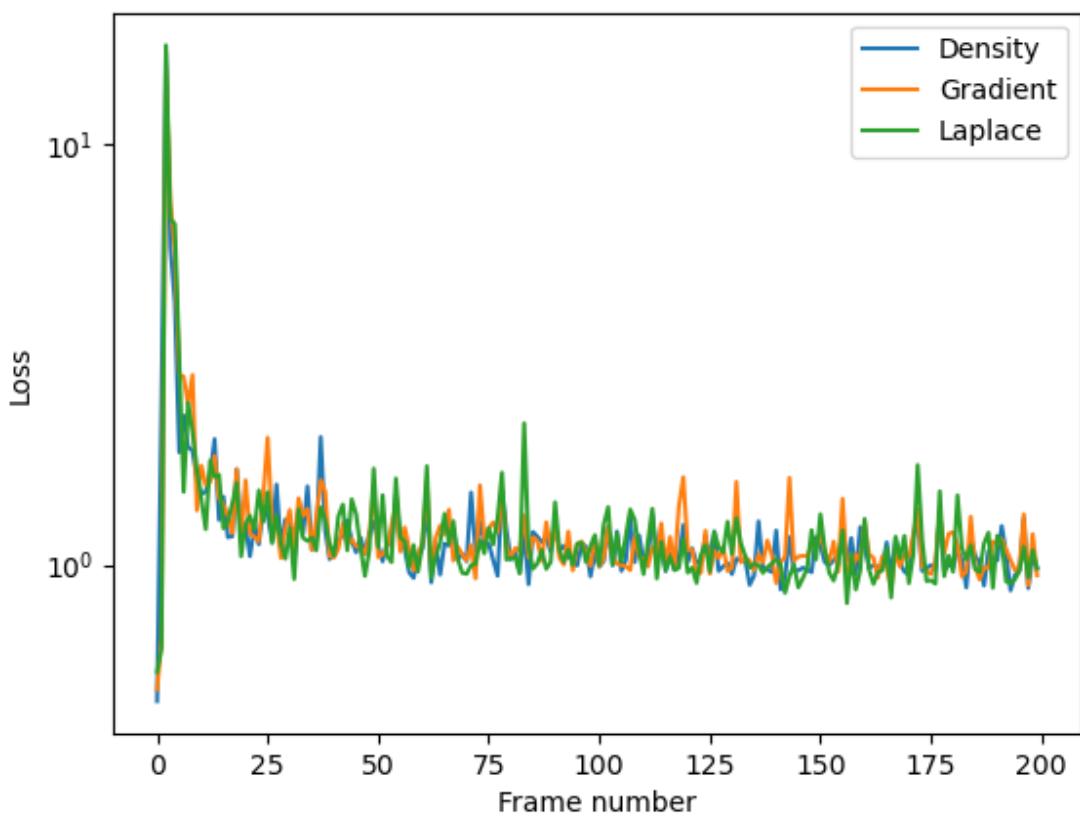


Рисунок 6.1 – Потери тренировки при добавлении различных дополнительных входных данных

На рисунке 6.3 представлено сравнение динамики потерь тренировки для нейросети с добавлением плотности во входные данные и без этого. Зелёным квадратом и зелёной пунктирной линией обозначено значение функции потери на двадцать пятом кадре у NRC с добавлением плотности.

Уже на двадцать пятом кадре тренировки с добавлением плотности потери тренировки падают примерно на такое же значение, как без добавления плотности на 150м кадре. Данные этого и других аналогичных графиков усреднены по десяти запускам алгоритма на каждую вариацию, чтобы уменьшить случайность и шум в данных, которые вносит трассировка путей.

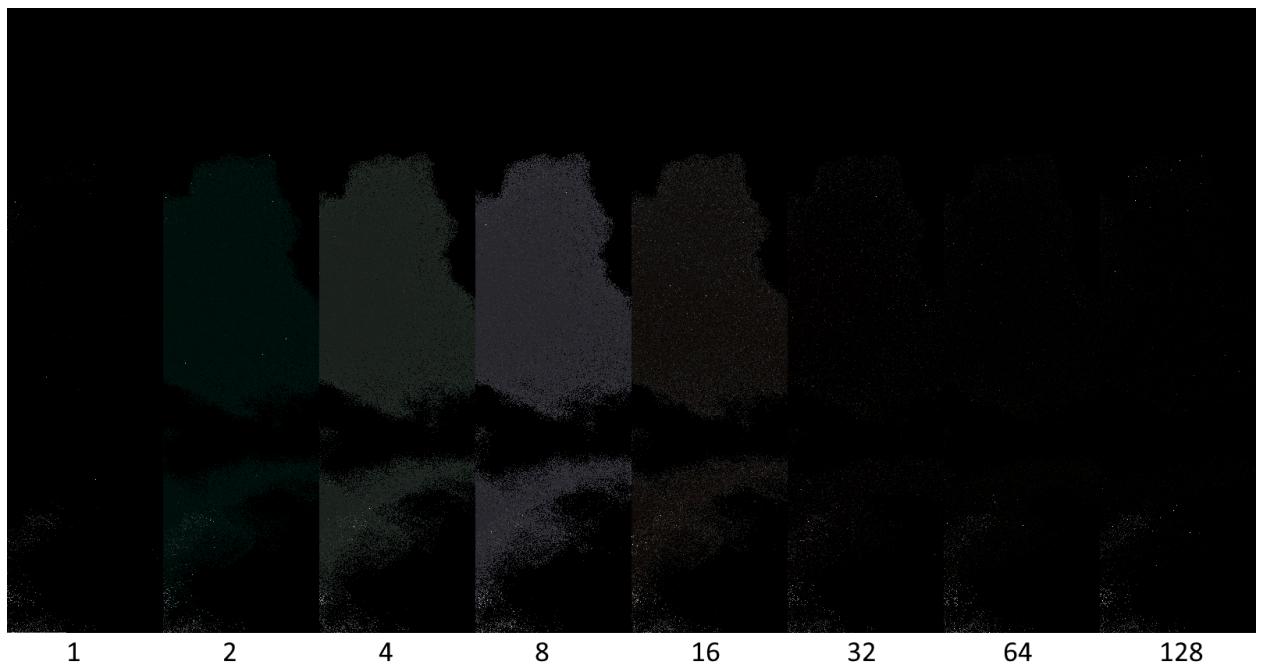


Рисунок 6.2 – Прогресс тренировки для NRC со входными данными:  
позиция+направление+плотность

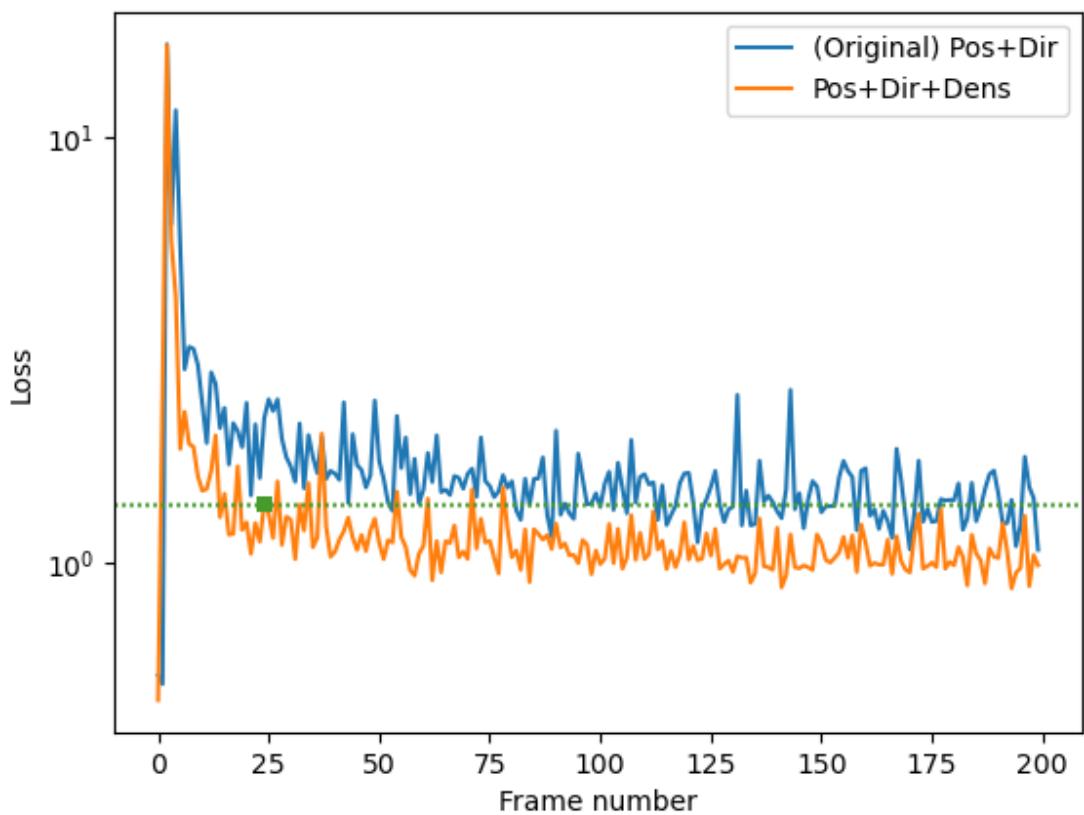


Рисунок 6.3 – Потери тренировки NRC с добавлением и без добавления  
плотности облака во входные данные

## 6.2 Фильтрация тренировочных пакетов

Сравнение времени тренировки без фильтрации тренировочных пакетов, с фильтрацией полосами, и с фильтрацией многоуровневой прямоугольной сеткой приведено в таблице 6.1. Время измеряется в миллисекундах. Буквы в ракурсах камеры в этой таблице соответствуют ракурсам камеры на рисунках 6.4–6.8. Ракурс камеры «без облака» означает, что облако совсем не попадает в кадр. Чтобы уменьшить влияние случайного шума на скорость тренировки, измерения были усреднены по сотне последовательных кадров.

В эксперименте при всех способах фильтрации изображение разделялось на четыре пакета равных размеров. «Без фильтрации» и «Полосы» разделяли изображение на четыре пакета вертикальными полосами. Для многоуровневой прямоугольной сетки использовалось два уровня: на нулевом уровне один пакет на всё изображение и на первом уровне четыре пакета. Использовать большее количество тренировочных пакетов становится невыгодно, так как каждый проход через тренировку нейросети начинает тратить намного больше времени, чем экономит фильтрации пустот в изображении.

Если рассмотреть результаты из таблицы, то можно заметить, что при использовании фильтрации, время тренировки может делать резкие скачки между различными ракурсами камеры. Это происходит из-за того, что при различных ракурсах нейросети потребовалось выполнить тренировку для разного количества тренировочных пакетов. Без фильтрации нейросеть всегда выполняет тренировку для всех четырёх пакетов. С использованием фильтрации полосами, в худшем случае (а), нейросети приходится выполнять тренировку для 4 пакетов. Многоуровневая сетка, в худшем случае (г), требует выполнить тренировку для 2 пакетов, а для ракурсов (а), (б) и (в) позволяет сократить число тренировок до 1.

Когда облако отсутствует в кадре, при обоих способах фильтрации тренировка не выполняется, в отличие от тренировки без фильтрации.

Таблица 6.1 – Сравнение времени тренировки при различных видах фильтрации тренировочных пакетов и различных ракурсах камеры

Ракурс камеры	Без фильтрации	Полосы	Многоуровневая сетка
а	13.06мс	10.61мс	<b>06.93мс</b>
б	11.52мс	<b>04.53мс</b>	05.34мс
в	10.01мс	<b>03.03мс</b>	04.27мс
г	10.88мс	<b>03.35мс</b>	06.03мс
д	10.41мс	03.37мс	<b>02.75мс</b>
без облака	09.62мс	<b>00.00мс</b>	<b>00.00мс</b>

Возможность объединять несколько соседних пакетов в один крупный, чтобы уменьшить число тренировочных проходов через нейросеть за один кадр, позволило значительно улучшить скорость тренировки в сложных ситуациях, когда облако занимает весь экран. В случаях, когда облако занимает много экранного пространства, многоуровневая фильтрация использует мало больших пакетов, а когда облако занимает малую часть экрана, используется мало маленьких пакетов на более глубоких уровнях дерева пакетов. В результате, в обоих случаях достигается повышенная, в сравнении с оригиналом, скорость тренировки и, соответственно, частота кадров, ведь тренировка нейросети является основным бутылочным горлышком алгоритма NRC.

### 6.3 Новые методы вычисления коэффициента пропускания

Были реализованы оба метода вычисления коэффициента пропускания из статьи An unbiased ray-marching transmittance estimator [2] и проведено сравнение между ними. Также в сравнение был включён алгоритм Ratio Tracking, который использовался в базовой имплементации NRC.



Рисунок 6.4 – (а)



Рисунок 6.5 – (б)

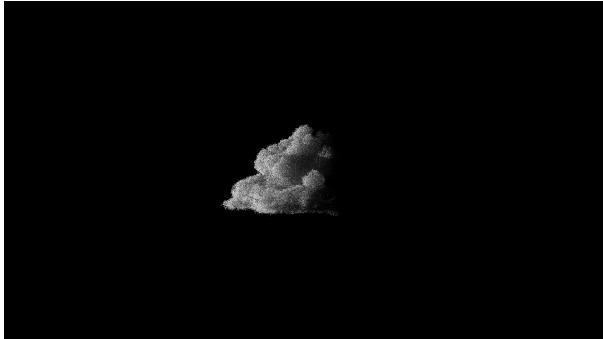


Рисунок 6.6 – (в)



Рисунок 6.7 – (г)



Рисунок 6.8 – (д)

Сравнение различных метрик при трассировке путей с вычислением коэффициента пропускания на пути к источнику света при помощи Ratio Tracking, а также несмешённого (unbiased) и смешённого (biased) методов из этой статьи представлено на рисунке 6.9. Численные результаты на этом рисунке усреднены по сотне кадров, чтобы исключить влияние случайного шума. Результаты рендеринга при применении NRC с различными методами вычисления коэффициента пропускания были получены при отрисовке сцены в 1 SPP и разрешении 1920x1080 пикселей. Метрика времени включает в себя только время на трассировку путей, без времени на обращения в нейросеть, так как вычисление коэффициента пропускания влияет только

на время трассировки путей. Референсное изображение отрисовано Монте-Карло трассировщиком путей без NRC с помощью несмешённого метода вычисления коэффициента пропускания из статьи в 8192 SPP.

Смешённый метод по своей сути является упрощённой версией несмешённого метода, так как вычисляет только нулевой элемент ряда Тейлора, что заметно упрощает вычисления. Это позволяет смешённому методу выделять большее количество ресурсов для точного вычисления нулевого элемента ряда по сравнению с несмешённым методом для повышения качества оценки. Для того чтобы исключить разницу в производительности между двумя методами и оценить разницу в качестве получаемого изображения, для обоих методов был выделен примерно одинаковый бюджет времени на вычисление коэффициента пропускаемости.

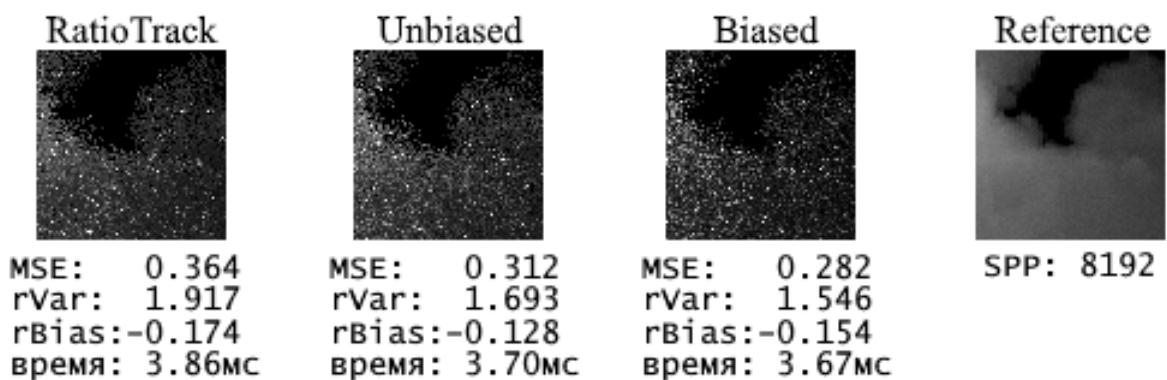


Рисунок 6.9 – Сравнение качества методов вычисления коэффициента пропускания, при использовании совместно с NRC

Из этих результатов можно сделать вывод, что новые методы из статьи во всём превосходят Ratio Tracking. Если сравнивать несмешённый и смешённый метод друг с другом, то смешённый метод имеет примерно на 10.5% процентов меньшую относительную дисперсию и MSE, но немного большее смещение, что примерно соответствует результатам из оригинальной статьи. В то же время, оба новых метода имеют меньшее относительное смещение, чем Ratio Tracking.

Для

измерения влияния новых методов оценки коэффициента пропускания на качество тренировки нейросети, посмотрим на динамику функции потерь несмещённого и смещённого методов, а также алгоритма Ratio Tracking. На рисунке 6.10 представлены усреднённые по десяти запускам показатели Relative L2 Loss функции нейросети при тренировке. Смещённый метод продемонстрировал самые лучшие показатели по функции потерь, даже несмотря на то, что он добавляет небольшое дополнительное смещение в результаты рендеринга по сравнению с несмещённым методом. Скорость достижения минимально возможного порога функции потерь у всех трёх способов вычисления коэффициента пропускания примерно одинаковая.

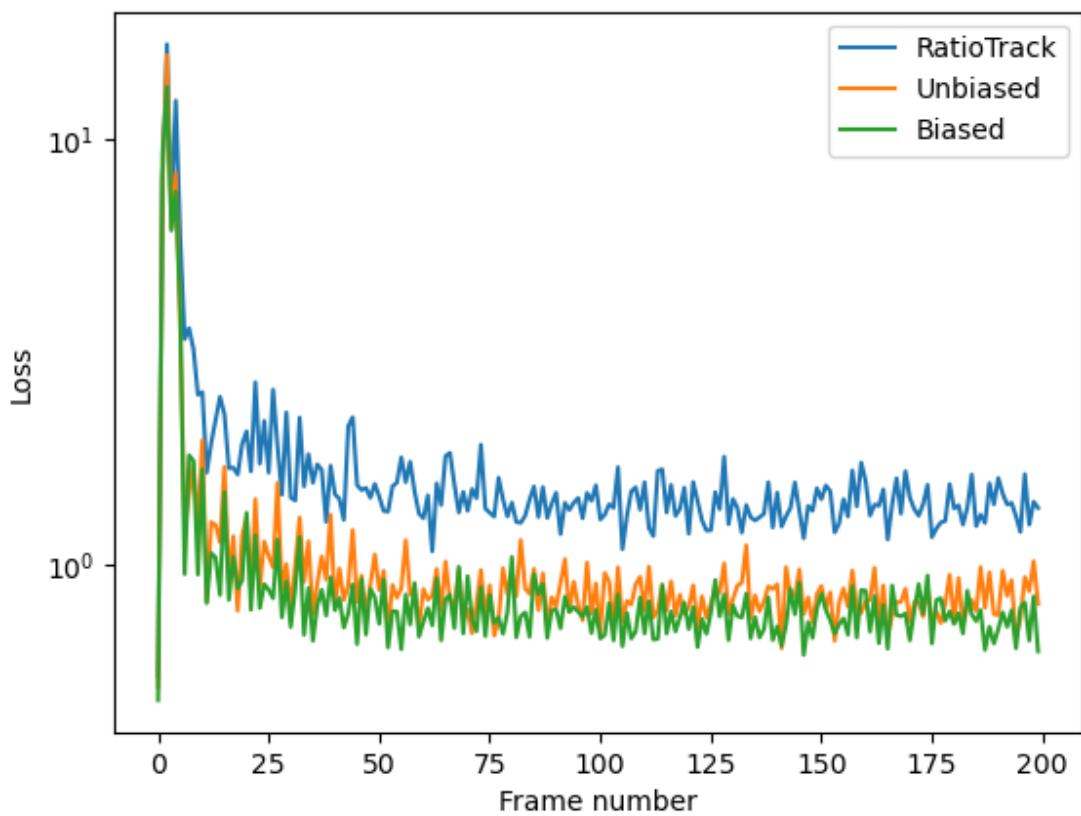


Рисунок 6.10 – Relative L2 loss функция тренировки нейросети с применением различных алгоритмов вычисления коэффициента пропускания

## ПЛАНЫ НА БУДУЩЕЕ

Предложенные в этой диссертации модификации NRC для рендеринга облаков и эксперименты с новыми методами вычисления коэффициента пропускания из статьи [2] проводились на довольно простой сцене со статичным облаком и единственным направленным источником света. В будущем было бы интересно исследовать производительность тех же методов на анимированных облаках со множеством источников света. Вероятно, при увеличении числа источников света также потребуется имплементация более эффективных способов пробирования источников света, таких как ReSTIR [47] для прямого (direct) освещения и ReSTIR GI [48] для непрямого (indirect) освещения.

Ещё одно направление, которое было бы интересно исследовать — это применение механизмов внимания, чтобы нейросеть могла самостоятельно предсказывать наилучшее распределение пикселей по пакетам, чтобы более эффективно фильтровать из инференса и тренировки пустые участки изображения.

## ЗАКЛЮЧЕНИЕ

В рамках данной магистерской диссертации было разработано две новые модификации для алгоритма NRC для рендеринга реалистичных облаков в реальном времени.

Первая модификация — адаптивная фильтрация тренировочных пакетов с применением многоуровневой прямоугольной сетки. Эта модификация уменьшает излишнюю перетренировку нейросети на «пустых» пакетах, при этом минимизируя накладные расходы по производительности при применении множества маленьких тренировочных пакетов.

Вторая модификация — добавление нормализованных данных о плотности облака во входные данные нейросети NRC, благодаря чему удалось сократить длительность тренировки нейросети примерно в шесть раз.

Также были проведены эксперименты над новыми методами вычисления коэффициента пропускания из статьи [2], и было продемонстрировано, что они могут быть успешно применены совместно с NRC. Новые методы из этой статьи уменьшают дисперсию рендеринга трассировки путей, благодаря чему также значительно уменьшается значение функции потерь нейросети NRC.

Исходный код данной работы доступен на GitHub [41] вместе с кодом для анализа результатов экспериментов.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Kokhanovsky A.* Optical properties of terrestrial clouds // Earth-Science Reviews. — 2004. — Т. 64, № 3. — С. 189—241. — ISSN 0012-8252. — DOI: [https://doi.org/10.1016/S0012-8252\(03\)00042-4](https://doi.org/10.1016/S0012-8252(03)00042-4). — URL: <https://www.sciencedirect.com/science/article/pii/S0012825203000424>.
2. An unbiased ray-marching transmittance estimator / M. Kettunen [и др.] // ACM Trans. Graph. — New York, NY, USA, 2021. — Июль. — Т. 40, № 4. — ISSN 0730-0301. — DOI: 10.1145/3450626.3459937. — URL: <https://doi.org/10.1145/3450626.3459937>.
3. *Pharr M., Jakob W., Humphreys G.* Physically Based Rendering: From Theory to Implementation. — 3rd. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2016. — ISBN 0128006455.
4. *Kajiya J. T.* The rendering equation // SIGGRAPH Comput. Graph. — New York, NY, USA, 1986. — Авг. — Т. 20, № 4. — С. 143—150. — ISSN 0097-8930. — DOI: 10.1145/15886.15902. — URL: <https://doi.org/10.1145/15886.15902>.
5. *Strugar F.* How to Build a Real-time Path Tracer. — NVIDIA, 03.2023. — Accessed: 2024-03-25. <https://www.nvidia.com/en-us/on-demand/session/gtcspring23-s51871/>.
6. *Trembilski A., Broßler A.* Transparency for Polygon Based Cloud Rendering // Proceedings of the 2002 ACM Symposium on Applied Computing. — Madrid, Spain : Association for Computing Machinery, 2002. — С. 785—790. — (SAC '02). — ISBN 1581134452. — DOI: 10.1145/508791.508943. — URL: <https://doi.org/10.1145/508791.508943>.
7. *Zamri M. N., Sunar M. S.* Atmospheric cloud modeling methods in computer graphics: A review, trends, taxonomy, and future directions // Journal of King

Saud University - Computer and Information Sciences. — 2022. — T. 34, № 6, Part B. — C. 3468—3488. — ISSN 1319-1578. — DOI: <https://doi.org/10.1016/j.jksuci.2020.11.030>. — URL: <https://www.sciencedirect.com/science/article/pii/S1319157820305711>.

8. *Goswami P., Neyret F.* Real-time Landscape-size Convective Clouds Simulation and Rendering // Workshop on Virtual Reality Interaction and Physical Simulation / под ред. F. Jaillet, F. Zara. — The Eurographics Association, 2017. — ISBN 978-3-03868-032-1. — DOI: 10.2312/vrphys.20171078.
9. A Comprehensive Review of Vision-Based 3D Reconstruction Methods / L. Zhou [и др.] // Sensors. — 2024. — Т. 24, № 7. — ISSN 1424-8220. — DOI: 10.3390/s24072314. — URL: <https://www.mdpi.com/1424-8220/24/7/2314>.
10. Real-Time Rendering of Point Clouds With Photorealistic Effects: A Survey / P. E. J. Kivi [и др.] // IEEE Access. — 2022. — Т. 10. — С. 13151—13173. — DOI: 10.1109/ACCESS.2022.3146768.
11. *Rusinkiewicz S., Levoy M.* QSplat: A multiresolution point rendering system for large meshes // Proceedings of the 27th annual conference on Computer graphics and interactive techniques. — 2000. — С. 343—352.
12. *Wald I., Seidel H.-P.* Interactive ray tracing of point-based models // Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005. — 2005. — С. 9—16. — DOI: 10.1109/PBG.2005.194058.
13. Real time ray tracing of point-based models / S. Kashyap [и др.] // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. — 2010. — С. 1—1.
14. *Zheng Q., Zwicker M.* Learning to Importance Sample in Primary Sample Space // Computer Graphics Forum. — 2019. — Т. 38, № 2. — С. 169—

179. — DOI: <https://doi.org/10.1111/cgf.13628>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13628>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13628>.
15. Neural Importance Sampling / T. Müller [и др.] // ACM Trans. Graph. — New York, NY, USA, 2019. — Окт. — Т. 38, № 5. — ISSN 0730-0301. — DOI: 10.1145/3341156. — URL: <https://doi.org/10.1145/3341156>.
16. Tables of Mie Scattering Functions for Particles with Refractive Index 1.5 / M. McCormick [и др.]. — National Aeronautics, Space Administration, 1969. — (NASA technical note). — URL: <https://books.google.ru/books?id=wewl6U7eOyYC>.
17. *Henyey L. G., Greenstein J. L.* Diffuse radiation in the Galaxy. // The Astrophysical Journal. — 1941. — Янв. — Т. 93. — С. 70—83. — DOI: 10.1086/144246.
18. *Cornette W., Shanks J.* Physically reasonable analytic expression for the single-scattering phase function // Applied optics. — 1992. — Июнь. — Т. 31. — С. 3152—60. — DOI: 10.1364/AO.31.003152.
19. *Draine B. T.* Scattering by Interstellar Dust Grains. II. X-Rays // The Astrophysical Journal. — 2003. — Дек. — Т. 598, № 2. — С. 1026—1037. — DOI: 10.1086/379123. — arXiv: astro-ph/0308251 [astro-ph].
20. *Jendersie J., d'Eon E.* An Approximate Mie Scattering Function for Fog and Cloud Rendering // SIGGRAPH 2023 Talks. — Los Angeles, CA, USA : Association for Computing Machinery, 2023. — DOI: 10.1145/3587421.3595409. — URL: <https://doi.org/10.1145/3587421.3595409>.
21. *Knoll A.* A survey of octree volume rendering methods. — 2006. — Янв.
22. *Felix.* Rendering volumetric clouds using signed distance fields. — UHawk VR, 05.2020. — URL: <https://blog.uhawkvr.com/rendering/rendering-volumetric-clouds-using-signed-distance-fields/> (дата обр. 18.12.2022).

23. Müller T., Gross M., Novák J. Practical Path Guiding for Efficient Light-Transport Simulation // Computer Graphics Forum. — 2017. — Т. 36, № 4. — С. 91—100. — DOI: <https://doi.org/10.1111/cgf.13227>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13227>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13227>.
24. Marschner S. R. Inverse rendering for computer graphics. — Cornell University, 1998.
25. Taniai T., Maehara T. Neural inverse rendering for general reflectance photometric stereo // International Conference on Machine Learning. — PMLR. 2018. — С. 4857—4866.
26. Chen Z., Nobuhara S., Nishino K. Invertible neural BRDF for object inverse rendering // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2021. — Т. 44, № 12. — С. 9380—9395.
27. Yu Y., Smith W. A. Inverserendernet: Learning single image inverse rendering // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2019. — С. 3155—3164.
28. Advances in Neural Rendering / A. Tewari [и др.] // Computer Graphics Forum. — 2022. — Т. 41, № 2. — С. 703—735. — DOI: <https://doi.org/10.1111/cgf.14507>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14507>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14507>.
29. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis / B. Mildenhall [и др.] // Commun. ACM. — New York, NY, USA, 2021. — Дек. — Т. 65, № 1. — С. 99—106. — ISSN 0001-0782. — DOI: 10.1145/3503250. — URL: <https://doi.org/10.1145/3503250>.

30. *Lindell D. B., Martel J. N., Wetzstein G.* Autoint: Automatic integration for fast neural volume rendering // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2021. — C. 14556—14565.
31. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks / T. Neff [и др.] // Computer Graphics Forum. — 2021. — Т. 40, № 4. — С. 45—59. — DOI: <https://doi.org/10.1111/cgf.14340>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14340>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14340>.
32. *Weiss S., Hermüller P., Westermann R.* Fast Neural Representations for Direct Volume Rendering // Computer Graphics Forum. — 2022. — Т. 41, № 6. — С. 196—211. — DOI: <https://doi.org/10.1111/cgf.14578>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14578>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14578>.
33. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding / T. Müller [и др.] // ACM Trans. Graph. — New York, NY, USA, 2022. — Июль. — Т. 41, № 4. — ISSN 0730-0301. — DOI: 10.1145/3528223.3530127. — URL: <https://doi.org/10.1145/3528223.3530127>.
34. Real-Time Neural Radiance Caching for Path Tracing / T. Müller [и др.] // ACM Trans. Graph. — New York, NY, USA, 2021. — Июль. — Т. 40, № 4. — ISSN 0730-0301. — DOI: 10.1145/3450626.3459812. — URL: <https://doi.org/10.1145/3450626.3459812>.
35. *Lin D., Wyman C., Yuksel C.* Fast Volume Rendering with Spatiotemporal Reservoir Resampling // ACM Trans. Graph. — New York, NY, USA, 2021. — Дек. — Т. 40, № 6. — ISSN 0730-0301. — DOI: 10.1145/3478513.3480499. — URL: <https://doi.org/10.1145/3478513.3480499>.

36. Interactive Path Tracing and Reconstruction of Sparse Volumes / N. Hofmann [и др.] // Proceedings of the ACM on Computer Graphics and Interactive Techniques. — 2021. — Апр. — Т. 4. — С. 1—19. — DOI: 10.1145/3451256.
37. NVIDIA. NVIDIA® OptiX™ AI-Accelerated Denoiser. — 2017. — Accessed: 2024-05-30. <https://developer.nvidia.com/optix-denoiser>.
38. Galvan A. Ray Tracing Denoising. — 10.2020. — Accessed: 2022-12-16. <https://alain.xyz/blog/ray-tracing-denoising>.
39. Attention Is All You Need / A. Vaswani [и др.]. — 2023. — arXiv: 1706.03762 [cs.CL].
40. Spindler J. NRC-HPM-Renderer. — 11.2023. — Accessed: 2024-03-25. <https://github.com/JanSpindler/NRC-HPM-Renderer>.
41. Fedotov B. NRC-HPM-Renderer. — 11.2023. — Accessed: 2024-03-25. <https://github.com/Fenrir7Asteron/NRC-HPM-Renderer>.
42. Disney Cloud Dataset. — 2017. — Accessed: 2024-03-27. <https://disneyanimation.com/resources/clouds/>.
43. Noise2Noise: Learning Image Restoration without Clean Data / J. Lehtinen [и др.]. — 2018. — arXiv: 1803.04189 [cs.CV].
44. Novák J., Selle A., Jarosz W. Residual ratio tracking for estimating attenuation in participating media // ACM Trans. Graph. — New York, NY, USA, 2014. — Нояб. — Т. 33, № 6. — ISSN 0730-0301. — DOI: 10.1145/2661229.2661292. — URL: <https://doi.org/10.1145/2661229.2661292>.
45. Integral formulations of volumetric transmittance / I. Georgiev [и др.] // ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia). — 2019. — Нояб. — Т. 38, № 6. — DOI: 10/dffn.

46. *Lee A.* U-Statistics: Theory and Practice (1st ed.) — Routledge, 1990. — DOI: 10 . 1201 / 9780203734520. — URL: <https://doi.org/10.1201/9780203734520>.
47. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting / B. Bitterli [и др.] // ACM Trans. Graph. — New York, NY, USA, 2020. — Авг. — Т. 39, № 4. — ISSN 0730-0301. — DOI: 10.1145/3386569.3392481. — URL: <https://doi.org/10.1145/3386569.3392481>.
48. ReSTIR GI: Path resampling for real-time path tracing / Y. Ouyang [и др.] // Computer Graphics Forum. T. 40. — Wiley Online Library. 2021. — C. 17—29.

## ПРИЛОЖЕНИЕ А

### Алгоритм поиска тренировочных пакетов в дереве

---

#### Algorithm 1 GetBatchesToTrain

---

**Входные данные:**  $level, l, r$ , максимальный уровень дерева  $h$ , массив со значениями true/false, произошли ли рассеивания в листах  $S_0 \dots S_{2^h}$ , множество пакетов  $\{i, j | B(i, j), i \in \mathbb{Z}, j \in \mathbb{Z}, 0 \leq i \leq h, 0 \leq j \leq 2^{2h}, i — это уровень дерева, j — это индекс пакета внутри уровня\}$ , список пакетов, прошедших фильтрацию  $F$

**Выходные данные:** произошло ли рассеивание (true/false)

```
if IsBatchScatteringOccured( $l, r, S$ ) = false then
    return false
if  $level = 0$  then return true
 $s = \frac{r - l}{4}$                                 ▷ Вычисляем размер пакета следующего уровня
for  $i = 0$  to  $3$  do
     $P_i = GetBatchesToTrain(level - 1, l + si, l + s(i + 1), S, B, F)$ 
if  $P_0 \dots P_3 = \text{true}$  then
    if  $level = h$  then
         $F = F \cup \{B_{level,0}\}$ 
    return true
for  $i = 0$  to  $3$  do
    if  $P_i = \text{true}$  then
         $batchIdx = \frac{l}{2^{2(h-level+1)}}$ 
         $F = F \cup \{B_{level-1,batchIdx}\}$ 
return false
```

---

---

#### Algorithm 2 IsBatchScatteringOccured

---

**Входные данные:**  $l, r, S$

**Выходные данные:** произошло ли рассеивание (true/false)

```
for  $i = l$  to  $r$  do
    if  $F_i > 0$  then
        return true
return false
```

---