УДК 378:004
DOI: 10.25686/978-5-8158-2474-4-2025-41-52

Трассировка лучей нейронных геометрических моделей

А. А. Николаев[2], Р. О. Родионов[1], В. А. Фролов[1, 2]
[1]Институт искусственного интеллекта МГУ, Москва, Россия
[2]Институт прикладной математики им.М.В.Келдыша РАН, Москва, Россия

Аннотация. В работе предлагается метод неявного представления поверхностей на основе нейронных направленных полей расстояний. Такие поля для начала и направления луча содержат расстояние до его пересечения поверхностью. Данный метод позволяет компактно хранить поверхности за счёт грубого приближения BVH-деревом и последующего уточнения нейронной сетью, усиленной пространственным кодированием хэш-таблиц с вектор-признаками. В сравнении с существующими методами предложенный метод потребляет в 3 раза меньше памяти при том же уровне точности представления поверхностей.

Ключевые слова: нейронный рендеринг, нейронная геометрия, трассировка лучей, сжатие геометрии, поля расстояний

Ray Tracing with Neural Geometry

A. A. Nikolaev[2], R. O. Rodionov[1], V. A. Frolov[1, 2]
[1]IAI Moscow State University, Moscow, Russia
[2]Keldysh Institute of Applied Mathematics, Moscow, Russia

Abstract. We introduce an implicit surface representation method built on neural directed distance fields. Given a ray defined by origin and direction, these fields output the distance to the intersection point. We compress surfaces by combining a coarse BVH approximation with a neural network refined through spatial hash-table encoding of vector features. Compared with existing approaches, our method achieves the same surface representation accuracy while using a third of a storage.

Keywords: neural rendering, neural geometry, ray tracing, geometry compression, distance fields

Введение

Полигональные сетки (меши) широко используются для представления объектов в компьютерной графике, однако плохо предназначены для хранения сложных моделей – попытки изобразить плавные поверхности и высокочастотные детали на объекте приводят к значительному увеличению размера представления.

Альтернативным представлением является представление нейронными сетями. Распространение нейронных моделей и улучшение их аппаратной и программной поддержки значительно расширило класс функций, которые можно эффективно приближать такими моделями, включив в него кодирование трехмерных сцен. Так или иначе, все алгоритмы синтеза изображений (трассировка лучей, поля расстояний со знаком SDF [1]) являются функциями, которые можно приближать нейронными сетями. Объект переводится в набор весов, а необходимая для рендера информация вычисляется прямым проходом нейронной сети.

Класс функций нейронных сетей достаточно широк [2]. Правильно подобранная архитектура и процедура обучения позволяют получить функции, соответствующие поверхностям, которые трудно представить полигональной сеткой. В то же время такой подход позволяет довольно быстро интегрировать нейронные представления в существующие рендер-системы – достаточно заменить запросы к свойствам трехмерного объекта на запросы к нейронной сети.

Основные характеристики нейронных представлений – точность, с которой восстанавливается исходная поверхность, скорость синтеза изображений и компактность. Оптимизация по всем трем значениям определяет фундаментальную проблему представления трехмерных объектов. В данной работе область ограничивается методами, подходящими для рендера в реальном времени, а фокусом является оптимальность по двум остальным свойствам.

Обзор существующих решений

1. Нейронные представления

Методы DeepSDF [3] и DeepLS [4] приближают функцию расстояния со знаком (SDF [1]). Авторы выделяют широкий набор сцен и обучают векторные представления, уникальные для каждой сцены,

и универсальный декодировщик. Такое представление имеет меньший размер, чем исходная полигональная сетка, однако для синтеза изображений требуется многошаговый алгоритм Sphere Tracing [5]. Работа [6] в дальнейшем сместила фокус с обучения латентного вектора под каждую сцену с общим декодировщиком к обучению отдельной нейронной сети для каждого объекта.

2. Пространственное кодирование

В NeRF [7] и других методах прямым проходом координаты точки в пространстве дополнительно кодируются через подстановку их в периодические функции с разными частотами.

ReLU Fields [8] предлагает определить в пространстве воксельную сетку и поместить в ее узлы обучаемые F-мерные векторы. Тогда кодирование точки в пространстве вычисляется как линейная интерполяция векторов в ближайших узлах сетки. Такой метод позволяет улучшить точность представления, но имеет количество обучаемых весов, растущее пропорционально количеству вокселей в сетке.

В работе [9] предлагается кодирование HashGrid: определяется таблица из T обучаемых F-мерных вектор-признаков и хэш-функция, осуществляющая (неинъективное) отображение узлов сетки в векторы. Так количество параметров перестает зависеть от разрешения сетки, однако возникающие коллизии ухудшают результат.

VQAD [10] является модификацией HashGrid, в котором отображение из узлов в таблицу становится обучаемым. В каждый узел сетки помещается индекс в таблицу признаков, который настраивается во время обучения. Недостатком метода является большое потребление памяти во время обучения, так как для непрерывной оптимизации каждый индекс требуется заменить на T весов для каждой строки таблицы.

Наконец, в работе [11] обобщается понятие функции отображения вокселей в таблицу вектор-признаков и предлагается альтернатива такой функции в VQAD [10], которая требует меньшее количество вычислений.

Появление описанных пространственных кодирований во многом снизило необходимость подбора функций активации и других конфигураций, как предлагалось в [8, 12, 13].

3. Направленные поля расстояний

Поля расстояний со знаком для данной точки пространства хранят расстояние до ближайшей точки поверхности. Для получения пересечения вдоль заданного направления приходится использовать алгоритм Sphere Tracing [5], требующий многократного обращения к нейронной сети. Альтернативным представлением являются направленные поля расстояний (Directed Distance Fields, DDF) [14], которые за одну итерацию позволяют получить расстояние до пересечения вдоль направления. Такой подход имеет свои недостатки, например, в отличие от SDF, DDF представляет разрывную функцию, что усложняет его приближение нейронными сетями.

В работе [14] приводится строгое определение такого поля, а также выявляются некоторые математические ограничения на функции, являющиеся направленными полями расстояний. В исследовании [15] рассматривается более эффективное представление луча для DDF, содержащее меньше степеней свободы. В работе [16] предлагается определять луч через две точки на сфере, содержащей сцену, а также модифицировать процедуру обучения для получения лучшей консистентности.

4. Применение ускоряющих структур

Для задачи построения нейронного представления по заданной полигональной сетке имеет смысл использовать ускоряющие структуры – методы классической компьютерной графики, предназначенные для более быстрого проведения запросов на пересечении лучей.

В работе [17] представлена обучающаяся нейронная модель, предсказывающая наличие пересечения для вторичных лучей. Модель затем интегрируется в классическую схему трассировки лучей, ускоряя ее. Алгоритм NGLoD [18], в свою очередь, строит вокруг заданного объекта октодерево, помещая в его узлы вектор-признаки. Такая схема позволяет перед запросом к нейронной SDF пропустить заведомо пустое пространство с помощью октодерева. При этом глубину обхода дерева можно менять после обучения, определяя разные уровни детализации.

NBVH [19] в сравнении с NGLoD [18, 19] предлагает использовать вместо октодерева дерево BVH [20], а вектор-признаки хранить по принципу HashGrid [9]. SDF заменено на направленное поле расстояний, которое также хранит информацию о нормалях и других параметрах поверхности. Для генерации

обучающей выборки (множества лучей) лучи генерируются случайно в увеличенном корневом узле дерева. Для каждого луча проводится обход дерева до первой точки пересечения.

При пересечении луча с узлом BVH внутри узла на луче семплируются k точек. Каждая точка кодируется HashGrid-вектором, векторы всех точек конкатенируются и формируют вход в нейронную сеть.

При хорошей степени сжатия NBVH [19] ограничен в точности представления геометрии. Изменение конфигурации для улучшения точности приводит к увеличению количества параметров нейронной сети и к уменьшению степени сжатия.

Работа [21] показывает, что применение нейросетей позволяет смягчить проблему дивергенции лучей при параллельной реализации трассировки лучей и получить ускорение. Алгоритм Neural Prefiltering [22] делит сцену на воксели и кодирует в весах нейронной сети результаты предрассчета трассировки лучей внутри каждого вокселя. Разные размеры вокселей определяют уровни детализации.

LSNIF [23] предлагает на нижних уровнях BVH определять воксельную сетку и обучать нейронную модель под пересечение внутри вокселей. Кроме того, узлы сетки вокселей и сетки HashGrid [9] выровнены. Начало и конец отрезка луча всегда попадают на грань вокселей HashGrid, что ускоряет интерполяцию значений вектор-признаков.

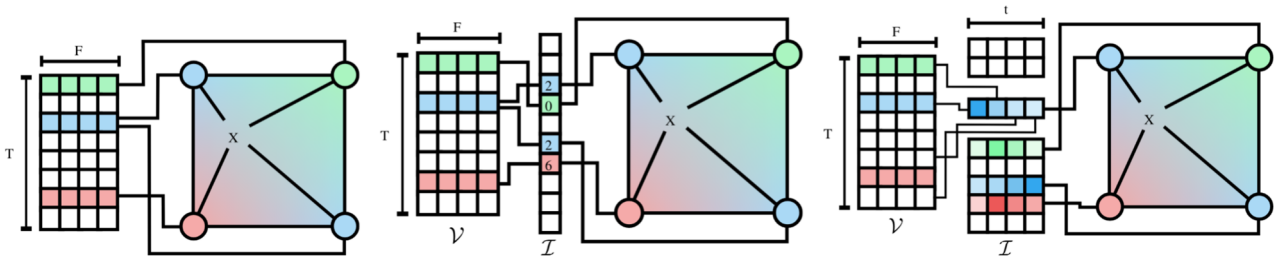
Во всех упомянутых методах нейронная сеть представляет собой многослойный перцептрон (MLP).

Предложенный метод

В работе предлагается метод, основанный на NBVH [19], улучшающий его сжимающую способность и гибкость в выборе параметров. Предложены модификации следующих компонентов: пространственного кодирования, нейросетевой архитектуры, генерации обучающей выборки.

1. Пространственное кодирование

Предложенный метод кодирования – HashVQAD – вдохновлен [11], сохраняет преимущества VQAD [10] и требует меньший объем памяти как для обучения, так и для хранения после фиксации весов.



(*a*) HashGrid (*b*) Прямой проход HashVQAD (*c*) Обучение HashVQAD

Рисунок 1. (*a*) HashGrid [9] определяет таблицу из T обучаемых вектор-признаков и фиксированное отображение узлов в строки таблицы хэш-функцией. (*b*) HashVQAD определяет промежуточную таблицу обучаемых индексов, указывающих на строки таблицы признаков. (*c*) Во время обучения в каждой ячейке таблицы индексов подбираются t < T весов. Индексы [0, t) отображаются в [0, T) с помощью хэш-функции

Пусть воксельная сетка имеет размерность N×N×N. F-мерные вектор-признаки хранятся в таблице V размера T аналогично [9]. При этом задается вторая таблица I размера H, каждая строчка которой хранит единственное число [0, T – 1] – индекс в таблицу признаков.

Каждому вокселю ставится в соответствие элемент таблицы I с помощью хэш-функции, аналогичной [9]. В свою очередь, каждый элемент i таблицы I указывает на некоторую строку V. Так для каждого вокселя определяется вектор-признак (рис. 1 (b).

Элементы I обучаемы. Чтобы избежать решения задачи дискретной оптимизации в целых числах, можно на время обучения каждый элемент I заменить на T вещественных весов. Вектор-признак тогда считается как взвешенная сумма строк V. Для уменьшения потребления памяти предлагается настраивать вес не для каждого из T векторов, а для t < T.

В таблице 1 приведены примеры значений гиперпараметров.

Таблица 1. **Конфигурации позиционного кодирования (F = 16, T = 2^8, N = 7)**

Метод	Хранение	Обучение	Параметры
VQAD	1 МБ	1 ГБ	-
HashVQAD	64 КБ	512 КБ	H = 2^{15}, t = 16

После обучения элементы I переводятся в целые числа операцией argmax. Вообще говоря, выходные значения кодирования до и после применения такой операции отличаются. Поэтому после основного этапа обучения проводятся еще несколько итераций оптимизации с уменьшенной температурой softmax (x – вектор весов, τ – температура):

$$\mathrm{softmax}(x,\tau)_i = \frac{\exp(x_i/\tau)}{\sum_j \exp(x_j/\tau)}.$$

При уменьшении температуры распределение по строчкам V приближается к вырожденному, потому переход на использование лишь одного индекса с наибольшим весом вносит минимальные изменения в выход модели. После завершения обучения индексы [0, t] переводятся хэш-функцией в значения [0, T].

Вычисление HashVQAD по количеству операций сравнимо с вычислением HashGrid. Такой подход не требует построения воксельного дерева, как в [10], в котором при N = 7 количество узлов достигает 2^{20}. Индекс в таблице признаков хранится не в каждом узле дерева, а в отдельной таблице, размер которой можно регулировать.

2. Нейросетевая архитектура

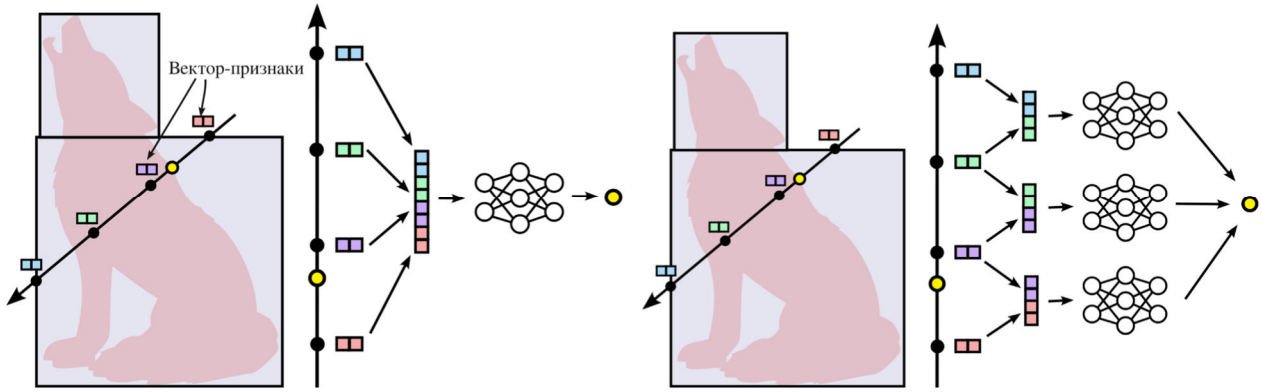
На отрезке, в котором ищется точка пересечения, семплируются дополнительные внутренние точки. Как правило, увеличение количества точек k приводит к лучшему качеству. При количестве точек ≥ 8 размерность первого слоя становится слишком большой и может составлять больше половины всех параметров MLP. К тому же ширина второго слоя не меняется, что приводит к потере значительной части дополнительной информации, предоставляемой отсемплированными точками.

В этой работе предлагается рассмотреть альтернативную архитектуру нейронной сети – KMLP.

Метод прямого прохода организован следующим образом (рис. 2):

- выделяется один MLP;
- среди k точек выделяются k − 1 пар соседних точек. Представления пространственного кодирования для каждой пары точек конкатенируются и по отдельности подаются на вход MLP;
- MLP для каждой пары точек вычисляет данные о пересечении на локальном отрезке между двумя точками. Далее данные для всех пар объединяются для получения данных о пересечении на всем отрезке.

В таком подходе количество параметров нейронной сети не зависит от количества семплируемых точек, а MLP не ограничивает точность метода. Кроме того, решается проблема разрывов значений поля DDF, если при разрыве происходит смена отрезка.



(*a*) Базовый метод (NBVH [19]) (*b*) Предложенный метод

Рисунок 2. (*a*) NBVH [19] конкатенирует векторы всех точек. При увеличенном количестве точек точность метода ограничена одним MLP, а первый слой принимает на вход неестественно большой вектор. (*b*) Несколько MLP с идентичными весами позволяют увеличивать количество точек и получать улучшение точности

3. Генерация обучающей выборки

Общепринятые алгоритмы построения BVH-дерева для каждого узла определяют множество примитивов, в нем содержащихся. При разбиении узла это множество примитивов делится на два непересекающихся множества. Форма ограничивающих объемов (выровненные по осям координат параллелепипеды, AABB) и частое отсутствие разделяющей гиперплоскости для двух подмножеств приводит к тому, что ограничивающие объемы двух новых узлов могут пересекаться.

Эта особенность делает возможной ситуацию, когда при обходе дерева точка пересечения луча с поверхностью находится внутри AABB-узла, но примитив, содержащий эту точку, принадлежит другому узлу (рис. 3). Это создает неконсистентную обучающую выборку – корректную с точки зрения обхода дерева, но некорректную с точки зрения реального нахождения пересечения в узле.

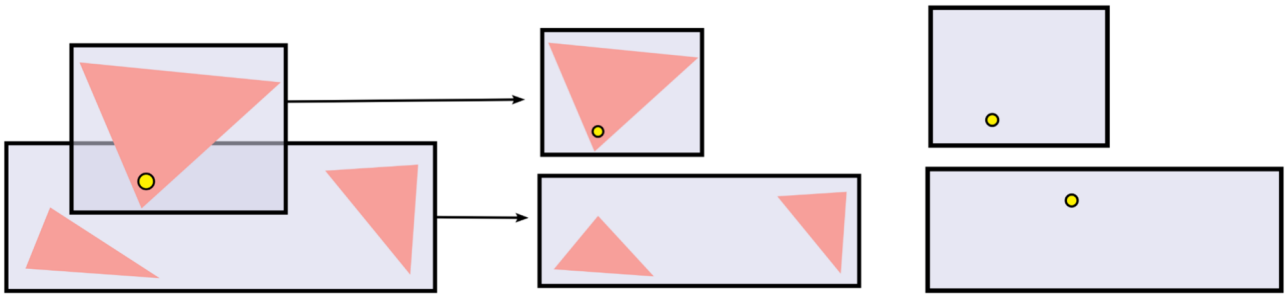
В LSNIF [23] такой проблемы не возникает, так как нейронная сеть вычисляет пересечения внутри сетки вокселей, а не произвольных ограничивающих объемов.

В данной работе предлагается генерировать обучающую выборку в два этапа. Сначала для каждого луча находятся все точки пересечения путем полного обхода дерева. Второй обход каждому листу ставит в соответствие ближайшую точку, которая лежит внутри AABB этого листа (если такая существует). Во время второго обхода принадлежность примитива точки листу не имеет значения, и в очередном узле точка учитывается, даже если ее примитив не принадлежит узлу (рис. 3).

Результаты

1. Набор данных и параметры обучения

Тестирование проводилось на нескольких классических моделях, а также на моделях из датасета Thingi10k [24]. Было отобрано 20 моделей разных размеров (рис. 4). Отобранные модели содержат достаточно разных участков поверхностей, чтобы на них можно было провести уверенное сравнение.



(*a*) Базовый метод (NBVH [19]) (*b*) Предложенный метод

Рис. 3. (*a*) Точка пересечения учтена только в одном из узлов. (b) Точка пересечения учтена оба раза вне зависимости от примитива, которому она принадлежит

Использовалась следующая конфигурация обучения и валидации:

- оптимизатор Adam [25];
- Learning Rate 0.001, остальные параметры оптимизатора по умолчанию;
- хранение весов в float32, Mixed Precision [26] не применяется;
- размер батча 400000 лучей, длительность обучения – 20 эпох по 800 батчей;
- размер изображения 1000 × 1000.

Каждый узел BVH-дерева содержит шесть чисел float32 (границы ограничивающего параллелепипеда) и индекс int32 левого сына. Левый и правый сыновья всегда идут подряд. С выравниванием до 32 байт каждый узел содержит 32 байта. Во всех экспериментах строилось 4096 узлов дерева, то есть все дерево занимает 128 КБ.

MLP содержит 4 внутренних слоя по 64 нейрона. На отрезке семплируется 8 точек. С учетом первого слоя базовая архитектура MLP занимает 200 КБ, архитектура KMLP – 100 КБ.

В экспериментах для HashGrid и HashVQAD зафиксированы количество уровней L = 4, разрешения нижнего и верхнего уровней $N_{min} = 16$, $N_{max} = 128$, размерность вектор-признаков F = 16. Для HashVQAD размер таблицы индексов H связан с размером таблицы признаков T соотношением $H = T \cdot 2^7$.

Для каждого индекса обучается t = 16 весов. Таким образом, единственный регулируемый параметр – размер таблицы признаков T.

Наблюдения и результаты других работ [9, 19, 10] показывают, что регулировки T достаточно, чтобы рассматривать кодировщики разного размера.

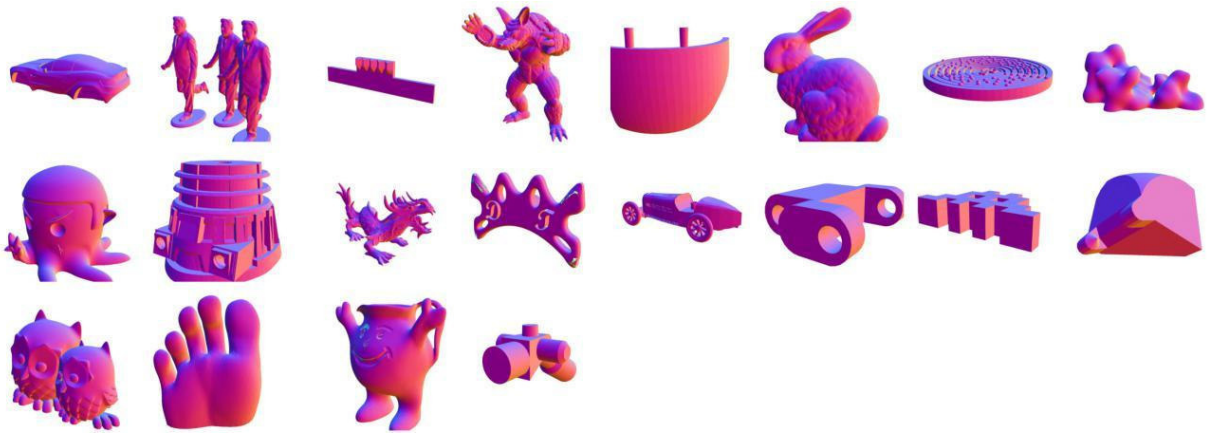


Рисунок 4. Примеры моделей, на которых проводилось тестирование

Качество нейронного представления определяется среднеквадратичной ошибкой между оригинальным изображением и изображением, полученным из этого представления. Синтез изображения проводится следующим образом: в сцене фиксируется точечный белый источник света, цвет пикселя определяется как косинус между нормалью в точке пересечения и направлением к источнику света. Такая формулировка учитывает предсказанные значения и для нормалей, и для расстояний до пересечения.

После завершения обучения синтезировалось 100 изображений с разных положений камеры, результаты усреднялись. В результатах экспериментов приведены значения MSE, усредненные по отобранным моделям.

2. Результаты

HashVQAD. В таблице 2 приведены результаты тестирования. Можно утверждать, что предложенное кодирование показывает аналогичные результаты при вдвое меньшем числе параметров.

Таблица 2. **Результаты тестирования кодирования HashVQAD.**

Тип кодирования	T	Размер кодирования	Полный размер	MSE
HashGrid	2^9	128 КБ	456 КБ	0.00255
HashVQAD	2^8	128 КБ	456 КБ	0.00230
HashGrid	2^{10}	256 КБ	584 КБ	0.00232
HashVQAD	2^{10}	256 КБ	584 КБ	0.00201
HashGrid	2^{11}	512 КБ	840 КБ	0.00197
HashVQAD	2^{10}	512 КБ	840 КБ	0.00188

Кодирование HashVQAD стабильно показывает лучшие результаты при том же количестве параметров.

KMLP. Результаты тестирования приведены в таблице 3. Разделение точек на соседние пары позволяет одновременно уменьшить количество параметров нейронной сети и получить прирост в качестве предсказаний.

Генерация обучающей выборки (лучей). В таблице 4 приведены результаты тестирования, показывающие улучшение точности при использовании новой стратегии генерации лучей.

Комбинация улучшений. В таблице 5 приведены результаты при последовательном применении всех предложенных улучшений. Размеры кодировщика подобраны так, чтобы качество предсказания оказывалось примерно одинаковым. Можно наблюдать, что качество, аналогичное базовому методу, достигается при втрое меньшем количестве параметров.

Таблица 3. **Результаты тестирования архитектуры KMLP**

Архитектура	T	Полный размер	MSE
MLP	512	456 КБ	0.00255
KMLP	512	356 КБ	0.00241
MLP	1024	584 КБ	0.00232
KMLP	1024	484 КБ	0.00219
MLP	2048	840 КБ	0.00197
KMLP	2048	740 КБ	0.00186

Таблица 4. **Результаты тестирования предложенного метода генерации лучей**

Метод генерации	T	Полный размер	MSE
Базовый	512	456 КБ	0.00255
Предложенный	512	456 КБ	0.00233
Базовый	1024	584 КБ	0.00232
Предложенный	1024	584 КБ	0.00210
Базовый	2048	840 КБ	0.00197
Предложенный	2048	840 КБ	0.00181

Таблица 5. **Результаты тестирования всех предложенных методов: I: базовый метод + KMLP; II: I + улучшенная генерация лучей; III: II + HashVQAD**

Метод	T	Полный размер	MSE
Базовый	2560	968 КБ	0.00188
I	1536	612 КБ	0.00198
II	1024	484 КБ	0.00190
III	512	356 КБ	0.00192

На рисунках 5 и 6 показаны примеры синтезированных изображений из сжатых представлений. Некоторые модели можно сжать в ≈ 40 раз и получать изображения, не отличимые с расстояния от оригинальных.

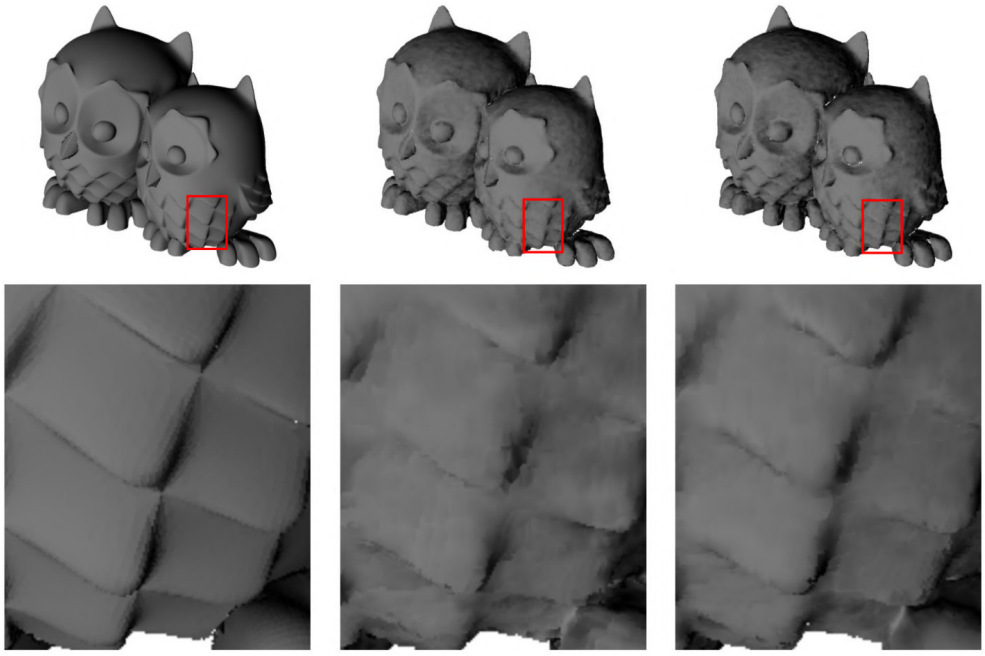


Рисунок 5. Синтез изображений из нейронных представлений. Слева направо: эталон (27 МБ), NBVH [19] (968 КБ, 0.00228 MSE), предложенный метод (868 КБ, 0.00176 MSE)

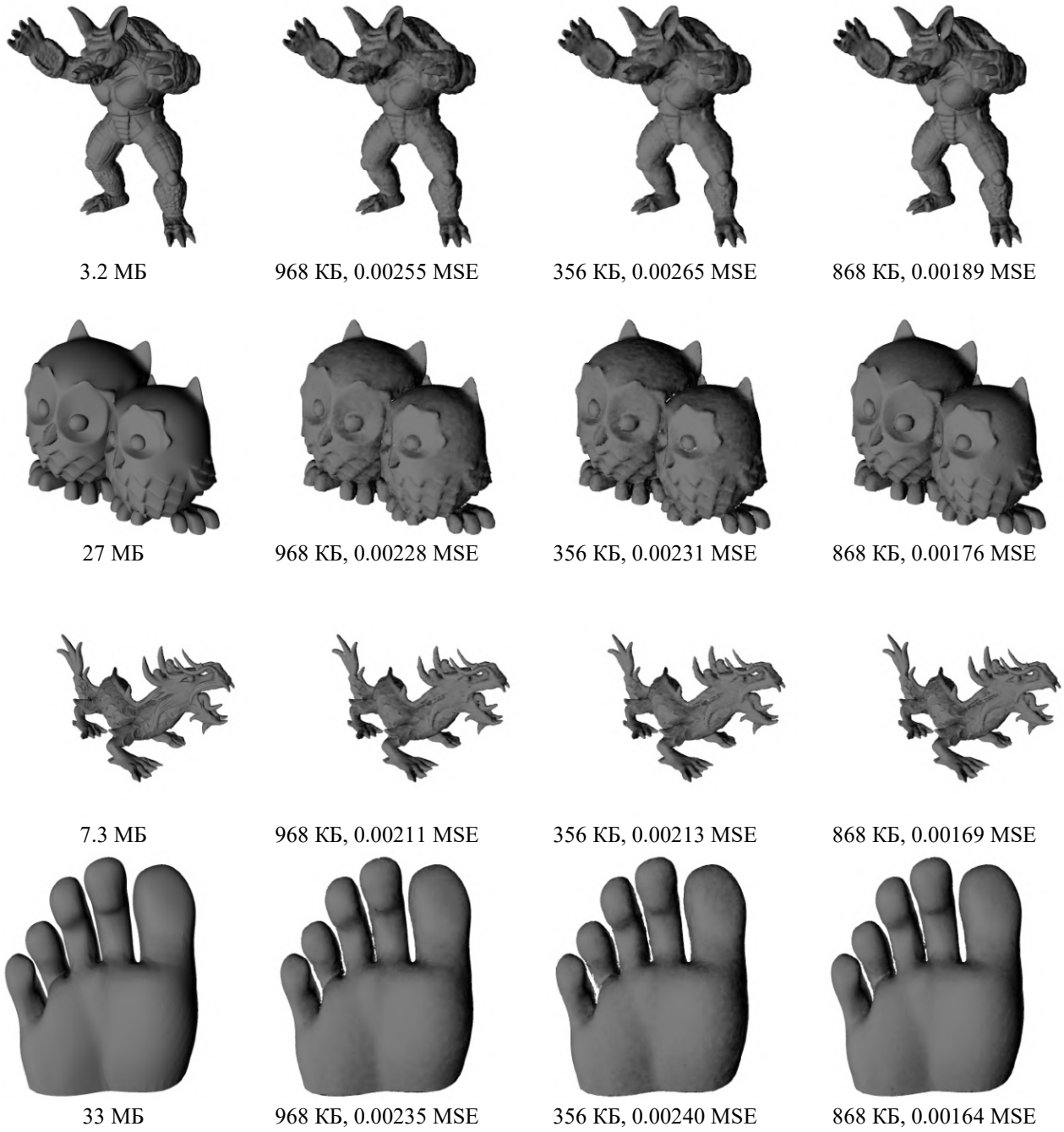


Рисунок 6. Изображения, по которым считается MSE. Фиксируется один точечный источник белого света, цвет пикселя определяется как косинус между нормалью и направлением к свету в точке пересечения (отрицательные значения заменяются нулем). Слева направо: оригинальное изображение, базовый метод, предложенный метод (356 КБ), предложенный метод (356 КБ).

На рисунке 7 показано сравнение базового метода и предложенного с визуализацией нормалей.

Приближение по центру показывает поверхность из большого количества мелких повторяющихся деталей, каждая из которых в полигональной сетке представляется отдельными треугольниками. В то же время нейронное представление позволяет сжимать поверхности такого вида.

Приближение снизу слева показывает сегмент, на котором проявляется преимущество KMLP: центральные отверстия неглубоки и попадают в один узел BVH. Разбиение отрезка на несколько меньших позволяет более точно приблизить разрывы, связанные с отверстиями.

Приближение снизу справа показывает недостаток, которым обладают оба метода: недостаточно качественное предсказание точек пересечения, которые находятся не в первых узлах при обходе BVH. Улучшенная генерация лучей в предложенном методе слегка улучшает качество в подобных участках.

Приведенные результаты показывают, что предложенный метод проявляет лучшую способность к сжатию, чем базовый метод. Предложенный метод позволяет получить то же качество изображений, что и базовый метод, используя в ≈ 3 раза меньше параметров. При этом преимущества предложенного метода можно проинтерпретировать, рассмотрев особо сложные участки поверхностей моделей.

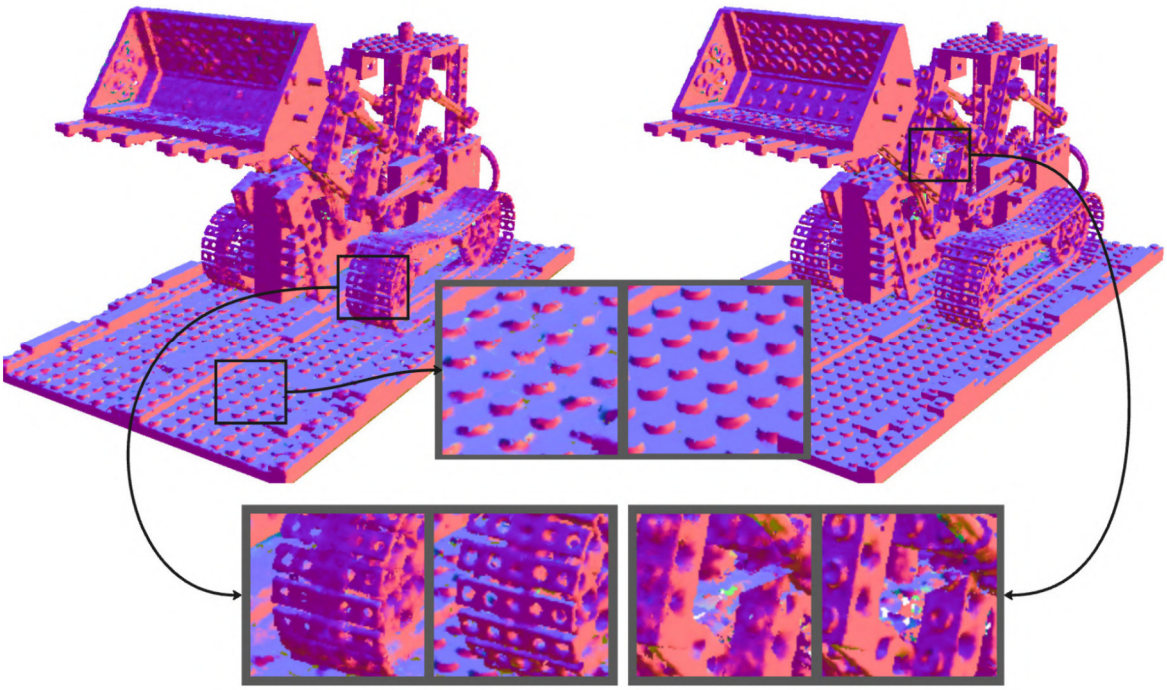


Рисунок 7. Базовый метод (слева) и предложенный (справа). Оба нейронных представления занимают 5 МБ, оригинальная модель занимает 82 МБ

Заключение

В работе описан и протестирован метод сжимающего нейронного представления, которое повышает точность базового метода NBVH [19]. Показано, как проявляют себя предложенные модификации в применении к разным поверхностям.

Улучшение точности достигается ценой относительно небольшого увеличения вычислительной сложности алгоритма: внедрение предложенных методов генерации обучающей выборки и пространственного кодирования не меняет количество операций, потому KMLP – единственный компонент, с увеличенным временем работы в сравнении с NBVH. В таблице 6 указано количество сложений и умножений, необходимых для прямого прохода MLP и KMLP.

Таблица 6. **Количество операций для KMLP и MLP. Используются четыре скрытых слоя по 64 нейрона, выходной слой имеет 5 нейронов. Размерность вектор-признаков каждой из k точек F = 16.**

Метод	k	Количество операций
MLP	4	16965
KMLP	4	44751
MLP	8	21061
KMLP	8	104419

Описанный метод имеет ограничения, унаследованные от NBVH [19]. Так, после изменения поверхности нейронное представление потребуется обучать заново. Кроме того, предложенные модификации пространственного кодирования и архитектуры нейронной сети используют больше обращений к памяти, что усложнит эффективную реализацию. Решению описанных проблем будут посвящены будущие исследования.

Отдельной задачей для будущих работ является эффективная реализация на графическом процессоре с учетом особенности метода – совмещения обхода дерева и запуска нейронной сети для каждого листа. Наивная реализация этапов обучения и применения метода на фреймворке PyTorch [27] позволяет обучить представление на рисунке 7 примерно за минуту (10 эпох) и отрендерить изображение из нейронного представления за ≈ 220 мс. Для экспериментов использовалась видеокарта RTX 4080 SUPER.

Список литературы

1. *Curless Brian*. A volumetric method for building complex models from range images // Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH ’96. New York, NY, USA: Association for Computing Machinery, 1996. P. 303–312. https://doi.org/10. 1145/237170.237269.

2. *Augustine Midhun T*. A Survey on Universal Approximation Theorems. 2024.

3. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation / Jeong Joon Park, Peter Florence, Julian Straub, et al. // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019. P. 165–174. http://dx.doi.org/10.1109/CVPR.2019.00025.

4. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction / Rohan Chabra, Jan E. Lenssen, Eddy Ilg, et al. // Computer Vision – ECCV 2020. Springer International Publishing, 2020. P. 608–625. http://dx.doi.org/10.1007/978-3-030-58526-6_36.

5. Hart John. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces // *The Visual Computer*. 1995. 06. Vol. 12.

6. Davies Thomas. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. 2020.

7. NeRF: representing scenes as neural radiance fields for view synthesis / Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, et al. // *Communications of the ACM*. 2021. Vol. 65, no. 1. P. 99–106. http://dx.doi.org/10.1145/3503250.

8. ReLU Fields: The Little Non-linearity That Could / Animesh Karnewar, Tobias Ritschel, Oliver Wang, Niloy Mitra // Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings. SIGGRAPH ’22. ACM, 2022. http://dx.doi.org/10.1145/3528233.3530707.

9. Instant neural graphics primitives with a multiresolution hash encoding / Thomas Muller, Alex Evans, Christoph Schied, Alexander Keller // *ACM Transactions on Graphics*. 2022. Vol. 41, no. 4. P. 1–15. http://dx.doi.org/10.1145/3528223.3530127.

10. Variable Bitrate Neural Fields / Towaki Takikawa, Alex Evans, Jonathan Tremblay, et al. // Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings. SIGGRAPH ’22. ACM, 2022. P. 1–9. http://dx.doi.org/10.1145/3528233.3530727.

11. *Takikawa Towaki*. Compact Neural Graphics Primitives with Learned Hash Probing. 2023. https://arxiv.org/abs/2312.17241.

12. *Tancik Matthew*. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. 2020.

13. *Sitzmann Vincent*. Implicit Neural Representations with Periodic Activation Functions. 2020

14. Representing 3D Shapes with Probabilistic Directed Distance Fields / Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, Allan Jepson // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2022. P. 19321–19332. http://dx.doi.org/10.1109/CVPR52688. 2022.01874.

15. *Feng Brandon Yushan*. PRIF: Primary Ray-based Implicit Function. 2022.

16. *Liu Zhuoman*. RayDF: Neural Ray-surface Distance Fields with Multi-view Consistency. 2023.

17. *Fujieda Shin*. Neural Intersection Function. 2023.

18. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes / Towaki Takikawa, Joey Litalien, Kangxue Yin, et al. // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021. P. 11353–11362. http://dx.doi.org/10.1109/CVPR46437.2021.01120.

19. N-BVH: Neural ray queries with bounding volume hierarchies / Philippe Weier, Alexander Rath, Elie Michel, et al. // Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers ’24. SIGGRAPH ’24. ACM, 2024. P. 1–11. http://dx.doi.org/10.1145/3641519.3657464.

20. A Survey on Bounding Volume Hierarchies for Ray Tracing / Daniel Meister, Shinji Ogaki, Carsten Benthin, et al. // *Computer Graphics Forum*. 2021. Vol. 40, no. 2. Pp. 683–712. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.142662.

21. Data Parallel Ray Tracing of Massive Scenes based on Neural Proxy / Shunkang Xu, Xiang Xu, Yanning Xu, Lu Wang // Pacific Graphics Conference Papers and Posters / Ed. by Renjie Chen, Tobias Ritschel, Emily Whiting. The Eurographics Association, 2024.

22. Neural Prefiltering for Correlation-Aware Levels of Detail / Philippe Weier, Tobias Zirr, Anton Kaplanyan, et al. // *ACM Trans. Graph.* 2023. Vol. 42, no. 4. – 16 pp. https://doi.org/10.1145/3592443.

23. *Fujieda Shin*. LSNIF: Locally-Subdivided Neural Intersection Function / Shin Fujieda, Chih-Chen Kao, Takahiro Harada // *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. 2025. Vol. 8, no. 1. P. 1–18. http://dx.doi.org/10.1145/3728295.

24. *Zhou Qingnan*. Thingi10K: A Dataset of 10,000 3D-Printing Models / Qingnan Zhou, Alec Jacobson // *arXiv preprint arXiv:1605.04797*. 2016.

25. *Kingma Diederik P.* Adam: A Method for Stochastic Optimization. 2017. https://arxiv.org/abs/1412. 6980.

26. *Micikevicius Paulius*. Mixed Precision Training. 2017.

27. *Paszke Adam*. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019. https://arxiv.org/abs/1912.01703.

28. Plenoxels: Radiance Fields without Neural Networks / Sara Fridovich-Keil, Alex Yu, Matthew Tancik, et al. // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2022. P. 5491–5500. http://dx.doi.org/10.1109/CVPR52688.2022.00542.

29. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions / Andy Zeng, Shuran Song, Matthias Niessner, et al. // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017. P. 199–208. http://dx.doi.org/10.1109/CVPR.2017.29.

30. *Wu Zhirong*. 3D ShapeNets: A Deep Representation for Volumetric Shapes. 2014.

31. *Perlin K., Hoffert E. M.* Hypertexture // *SIGGRAPH Comput. Graph.* 1989. Vol. 23, no. 3. P. 253–262. https://doi.org/10.1145/74334.74359.

32. *Perlin K., Hoffert E. M.* Hypertexture // Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '89. New York, NY, USA: Association for Computing Machinery, 1989. P. 253–262. https://doi.org/10.1145/74333.74359.

33. *Hu Edward J.* LoRA: Low-Rank Adaptation of Large Language Models. 2021. https://arxiv.org/abs/2106.09685.

34. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields / Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, et al. // 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2021. http://dx.doi.org/10.1109/ICCV48922.2021.00580.

35. Adaptive Shells for Efficient Neural Radiance Field Rendering / Zian Wang, Tianchang Shen, Merlin Nimier-David, et al. // *ACM Transactions on Graphics*. 2023. Vol. 42, no. 6. P. 1–15. http://dx.doi.org/10. 1145/3618390.

36. 3D Gaussian Splatting for Real-Time Radiance Field Rendering / Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, George Drettakis // *ACM Transactions on Graphics*. 2023. Vol. 42, no. 4. P. 1–14. http://dx.doi.org/10.1145/3592433.

37. *Guedon Antoine*. Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering. 2024.

38. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs / Christian Reiser, Songyou Peng, Yiyi Liao, Andreas Geiger // 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2021. http://dx.doi.org/10.1109/ICCV48922.2021.01407.

39. *Yariv Lior*. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. 2023.

40. Baking Neural Radiance Fields for Real-Time View Synthesis / Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, et al. // 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2021. http://dx.doi.org/10.1109/ICCV48922.2021.00582.

41. *Knodt Julian*. Neural Ray-Tracing: Learning Surfaces and Reflectance for Relighting and View Synthesis. 2021.

42. *Kim Doyub, Lee Minjae, Museth Ken.* NeuralVDB: High-resolution Sparse Volume Representation using Hierarchical Neural Networks // *ACM Transactions on Graphics*. 2024. Vol. 43, no. 2. P. 1–21. http://dx.doi.org/10.1145/3641817.

43. *Wang Peng*. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. 2021.

44. Occupancy Networks: Learning 3D Reconstruction in Function Space / Lars Mescheder, Michael Oechsle, Michael Niemeyer, et al. // 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019. http://dx.doi.org/10.1109/CVPR.2019.00459.

45. *Chibane Julian*. Neural Unsigned Distance Fields for Implicit Function Learning. 2020.

46. *Sitzmann Vincent*. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. 2021.

47. *Li Zhong*. NeuLF: Efficient Novel View Synthesis with Neural 4D Light Field. 2021.

48. *Kulkarni Nilesh*. What's Behind the Couch? Directed Ray Distance Functions (DRDF) for 3D Scene Reconstruction. 2021.

49. *Feng Brandon Yushan, Varshney Amitabh*. SIGNET: Efficient Neural Representation for Light Fields // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2021. October. Pp. 14224–14233.

50. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations / Mehdi S.M. Sajjadi, Henning Meyer, Etienne Pot, et al. // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2022. P. 6219–6228. http://dx.doi.org/10.1109/ CVPR52688.2022.00613.

51. *Wang Peng*. Progressively-connected Light Field Network for Efficient View Synthesis. 2022.

52. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis / Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, et al. // 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2021. P. 7491–7500. http://dx.doi.org/10.1109/CVPR46437.2021.00741.

53. *Moenne-Loccoz Nicolas*. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. 2024. https://arxiv.org/abs/2407.07090.

54. *Epic Games Inc*. Nanite. 2020.

55. *Zhang Kai*. NeRF++: Analyzing and Improving Neural Radiance Fields. 2020. https://arxiv.org/abs/ 2010.07492.

56. *Deng Kangle*. Depth-supervised NeRF: Fewer Views and Faster Training for Free. 2024. https://arxiv.org/abs/2107.02791.