Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Интеллектуальных Информационных Технологий

Никитин Сергей Денисович

Сравнительный анализ представлений функций дистанции со знаком

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н.
В.А.Фролов

Москва, 2025

Аннотация

Дипломная работа посвящена сравнительному анализу функций дистанции со знаком (SDF) в задачах рендеринга 3D-моделей. Целью исследования является разработка и реализация бенчмарка для оценки различных представлений SDF по метрикам качества изображения (PSNR), времени рендеринга и размера модели. В работе рассмотрены теоретические основы SDF, включая аналитические, воксельные и нейронные представления, а также проведён обзор существующих подходов и их ограничений. Разработанный бенчмарк позволяет рендерить 3D-модели в различных представлениях и сравнивать их производительность на единой платформе. Эксперименты проведены на наборе тестовых моделей с использованием современного оборудования и программных инструментов. Работа имеет практическую значимость для компьютерной графики, игровой индустрии и 3D-моделирования.

Содержание

1 Введение

В современном мире комьютерная графика используется для решения широкого класса задач. Зародившись, как развлечение для небольшого числа людей, на текущий момент она породила целые индустрии, в который работают тысячи людей по всему миру. С каждым годом проработанность виртуальных миров возрастает, что в свою очередь требует увеличение вычилительных мощностей для рендера таких сцен и ресурсов для хранения полученных результатов.

В компьютерной графике существует множество представлений объектов. Однако самый популярный - полигональная сетка (меш), потому что с помощью неё можно сколь угодно точно задать поверхность. Чем сложнее поверхность, тем больше треугольников будет создано для поддержания заданной детализованности. Значит, возрастут затраты на память и рендер таких объектов.

Другим популярным представлением объектов можно назвать функцию расстояния со знаком (SDF). Функции дистанции со знаком (Signed Distance Functions, SDF) играют ключевую роль в современных задачах компьютерной графики, 3D-моделирования, рендеринга и физического моделирования. Они обеспечивают компактное и гибкое представление геометрических объектов, позволяя эффективно выполнять операции рендеринга, столкновений и оптимизации. Разнообразие подходов к реализации SDF, таких как аналитические, воксельные и нейронные представления, создаёт необходимость их сравнительного анализа для выбора оптимального решения в конкретных задачах.

SDF характеризуются рядом фундаментальных свойств, определяющих их практическую ценность в задачах компьютерной графики. Ключевым атрибутом является инвариантность относительно масштабирования и аффинных преобразований, что позволяет эффективно манипулировать геометрическими объектами без потери точности представления. Более того, SDF обладают свойством композиционности, что обеспечивает возможность построения сложных геометрических форм посредством математических операций над базовыми примитивами. Актуальность данной темы обусловлена растущим спросом на высококачественный и производительный рендеринг в игровой индустрии, виртуальной реальности и автоматизированном проектировании, где SDF находят широкое применение.

В данной работе решается актуальная задача приведения нескольких методов к формату, в котором их можно сравнить по нескольким метрикам и тем самым получить данные о плюсах и минусах рассматриваемых реализаций.

2 Постановка задачи

2.1 Цель работы

Цель данной работы - проведение сравнительного анализа существующих методов представления поверхностей для задача рендеринга. Для достижения такой цели надо решить следующие задачи:

1. Изучить теоретические основы и существующие подходы к функциям дистанции со знаком.
2. Выработка методологии.
3. Разработка программной системы для сравнительного анализа.
4. Проведение экспериментальных сравнений выбранных представлений SDF.
5. Анализ и выводы.

2.2 Формальная постановка задачи

Для формальной постановки задачи используются следующие определения:

- **Пиксель** - вектор $(p_r, p_g, p_b, p_w) \in [0,1]^3$, где первые три числа отвечают цветовым каналам, а последнее - прозрачность. Пространство таких векторов обозначим $\mathcal{P}$.
- **Изображение** разрешения $H \times W$ назовем матрицу $I \in \mathcal{P}^{H \times W}$.
- **Функция дистанции со знаком** - функция f, переводящая точку пространства $\mathbb{R}^3$ в расстояние до ближайшей границы объекта или заданного множества. Если точка внутри объекта, то значение будет с минусом. Таким образом можно различать ситуации, когда точка внутри и вне множества.
  То есть, $f \in \mathbb{F} : \mathbb{R}^3 \to \mathbb{R}$.
- Пусть $M_1, M_2, ..., M_k$ - метрики для оценивания точности.

Вход:

- $\mathbf{f} \in \mathbb{F}$ - функция дистанции со знаком.
- $\mathbf{C} \in \mathbb{R}^{4\times3}$ - матрица камеры.
- $\mathbf{L} \in \mathbb{R}^3$ - точечный источник света.

Выход:

- Отрендеренное изображение $\mathbf{I} \in \mathcal{P}^{H \times W}$.

Подсчет точности каждого метода:
После получения изображений, отрендеренных каждым методом, они начинают сравниваться с референсными. Принцип работы - $M_i : I \times I \to \mathbb{R}$, $i = \overline{1,k}$. В итоге, появляются оценки точности каждого метода, относительно выбранного референсного представления.

Подсчет других метрик:
Кроме точности, считаются такие показатели, как время рендера и занимаемая память, их можно охарактеризовать следующим образом:

- Время рендера: $t_2 - t_1$ - количество миллисекунд, прошедших между началом рендера и его концом.
- Память: так как все методы хранят информацию по-разному, то пусть есть функция $\mathbf{g} : \mathbb{F} \to \mathbb{R}$, которая по конкретному методу считает количество Мегабайт, занимаемых представлением.

3 Обзор существующих работ

Несмотря на то, что существует множество методов сжатия 3D-геометрии, разработка метода сжатия или представления поверхности, который можно было бы использовать непосредственно во время рендеринга, остается серьезной проблемой. Это связано с фундаментальным компромиссом между достижением высокой степени сжатия, качеством и вычислительной сложностью метода декомпрессии, который может быть трудно распараллелить [1][2][3]. Другой проблемой являются уровни детализации (LoD), которые обычно увеличивают потребление памяти, поскольку они хранятся в виде отдельных структур данных и также могут создавать артефакты [4][5].

Отдельно стоит отметить выбор метода рендеринга, который так или иначе будет использоваться на протяжении всех работ. Это выбор между растеризацией и трассировкой лучей. В данном исследовании уделяется больше внимания методам, основанным на трассировке лучей, и на это есть три причины.

1. Обычно алгоритмическая сложность рендеринга с растеризацией линейно зависит от количества полигонов (либо геометрических примитивов), в то время как при трассировке лучей она линейно зависит от количества пикселей изображения и логарифмически зависит от количества полигонов. Существующие нестандартные методы рендеринга используют трассировку лучей или пирамид для определения видимости [6].

2. Вторая причина - универсальность трассировки лучей, которая позволяет визуализировать примитивы любого вида путем алгоритмического определения пересечения луча и примитива: т.е. не требует обязательной тесселяции в треугольники.

3. Наконец, третья причина - это возможность рендеринга как на центральном, так и на графическом процессорах, даже одновременно, если это необходимо. Это важно для научной визуализации и для рендеринга CAE-данных, поскольку позволяет работать с огромными объемами данных в оперативной памяти. Например, это активно используют визуализаторы на базе OSPRay [7], а в индустрии кинопроизводства MoonRay [8]. [8] использует этот путь из-за высокой гибкости рендерера на базе центрального процессора.

3.1 Представления на основе полигональной сетки

Стандартный подход с использованием уровня детализации (LoD) [9][10] не позволяет достичь двух важнейших целей.

Во-первых, он не обеспечивает компактного хранения геометрии, поскольку каждый уровень детализации постепенно увеличивает потребление памяти. Во-вторых, он по своей сути не поддерживает плавные переходы с высоким уровнем детализации и потоковую передачу. В то же время алгоритмы сжатия сетки часто создают представления, которые трудно визуализировать, особенно при трассировке лучей [11].

Геометрические изображения [12] представляют собой метод преобразования произвольной поверхности в полностью правильную структуру. Это позволяет легко создавать новые сетки и обрабатывать геометрию как изображение, которое может быть дополнительно сжато [13][14] и с трассировкой лучей, обеспечивая при этом как динамическую геометрию, так и эффективный уровень детализации схемы без дополнительных затрат. Создание геометрического изображения требует хорошей параметризации 3D-поверхности, которую не всегда легко получить.

Micromesh[5] преобразует сетку с большим количеством полигонов в "базовую" сетку с низким количеством полигонов вместе со смещением. Векторы смещений сохраняются для каждой вершины в расчлененной (базовой) сетке, а карта смещений используется внутри полигона. В некотором смысле, этот подход можно рассматривать как эволюцию геометрических изображений, адаптированных к конкретным требованиям к качеству и оптимизированных для современного графического оборудования.

Концепция дерева тесселяции, представленная в [4], направлена на сокращение передачи данных из памяти в блоки обработки при растеризации и трассировке лучей треугольных сеток. Это исследование подчеркивает важность уровней детализации для повышения производительности трассировки лучей и энергоэффективности. Однако оно не решает проблему компактного хранения геометрических данных. Вполне вероятно, что подход [4] принят Nvidia в расширении Vulkan под названием VK_NV_cluster_acceleration_structure [15].

Представления на основе сетки были и остаются современным подходом (SOTA) к рендерингу в компьютерной графике. Это подтверждается недавними демонстрациями от Nvidia [15]. Ключевым аспектом здесь является то, что как методы тесселяции [VK_NV_TC 2025], так и методы определения уровня детализации (LOD) [VK_NV_LC 2025] совместимы как с растеризацией, так и с трассировкой лучей. Учитывая, что производители графических процессоров внедряют аппаратное ускорение для этих методов, мы ожидаем, что наиболее эффективная реализация будет включать использование соответствующих функций расширения Vulkan, которые напрямую поддерживаются драйвером графического процессора. Таким образом, программная архитектура бенчмарка сохраняет возможность обработки сеток с помощью трассировки лучей с возможностью интеграции таких подходов при необходимости.

3.2 Неявные аналитические представления

Набор разреженных блоков (SBS) [16] объединяет BVH с небольшими регулярными сетками внутри его листьев, называемыми блоками. В нем используются эффективные алгоритмы пересечения (аналитический метод и метод Ньютона) для поиска пересечений лучей и SDF. Эти алгоритмы запрашивают значения SDF в углах вокселя, которые должны явно поддерживаться структурой данных. VDB [17] использует иерархические многоуровневые сетки (своего рода B+дерево в трехмерном пространстве), что помогает поддерживать баланс высот при построении и обходе. VDB использует битовую маску, которая хранит информацию о том, какие вокселы имеют значение; они называются “активными вокселами”. Однако VDB не поддерживает значения самой функции SDF, и в то же время иерархические сетки обычно демонстрируют низкую производительность рендеринга. Оптимизированная графическая реализация преобразования лучей с помощью структур VDB представлена в [18], где луч пересекает объем узла с помощью дифференциального 3D-анализатора (3DDA).

Октодерево разреженных вокселей (SVO) [19] включает структуру, которая хранит только занятые области; большая часть воксельной информации хранится на родительском уровне, чтобы уменьшить использование памяти. Другим применением SVO является работа [20], в которой представлено исходное октадерево с добавлением того, что коэффициенты многочлена сохраняются в листьях, если они имеют часть поверхности; в противном случае, на этапе предварительной обработки [20] пытается максимизировать размер из поля, обозначающего пустую область. Это усовершенствование позволяет хранить данные, расположенные только на границе объекта. На следующем этапе [20] использует полиномы для интерполяции значений SDF путем нахождения аналитического пересечения. Подход Пуйоля и Чики позволяет использовать различные типы интерполяции, но он также накладывает ограничения и проблемы, связанные с аналитическим поиском корней. Алгоритм построения SDF основан на их предыдущей работе [21], в которой представлен подход к эффективному представлению SDF и запросам к ним путем построения адаптивной структуры данных. Он использует RMSE для принятия решения о разделении узлов. [21] могут использовать несколько методов интерполяции, таких как трикубический, для решения линейной системы [22], которая требует больших вычислительных затрат.

Хэш-таблицы [23] обычно используются вместо октодерева в приложениях реконструкции, где геометрия постоянно меняется и важно быстро обновлять структуру данных.

Методы рендеринга на основе SDF обладают несколькими ключевыми преимуществами:

1. Унифицированные структуры данных: Одни и те же представления данных могут использоваться как для поверхностного, так и для объемного рендеринга.

2. Простота генерации уровня детализации (LOD) и переключения во время рен-

деринга: Лоды могут быть получены простым способом, по крайней мере, для "водонепроницаемых"объектов.

3.3 Рендеринг геометрии CSG

В моделях CAD геометрия может быть определена с помощью так называемого представления Constructive Solid Geometry (CSG). Это представление описывает геометрию как набор базовых примитивов (таких как сфера, цилиндр, конус и другие) и операции, выполняемые над этими примитивами, как над наборами. Основным преимуществом такого представления является его точность, как правило, высокая гибкость и чрезвычайная компактность (например, MPR CSG [24]). Основным недостатком является его высокая вычислительная сложность из-за потенциального значительного совпадения между примитивами в некоторых моделях, что вынуждает алгоритм рендеринга интенсивно перебирать поддеревья CSG. Чтобы эффективно отобразить такое представление, необходимо преобразовать древовидную иерархию CSG, минимизируя взаимные совпадения, когда это возможно [25]. Один из таких подходов называется “Повторное плетение” [26], который уменьшает количество перекрывающихся элементов внутри объема листа. Однако недостатком ребрендинга является увеличение объема памяти и усложнение процедуры сборки.

MPR использует “ленту” для эффективного хранения данных и передачи их в графический процессор. Этот подход обеспечивает эффективную производительность и сжатие модели, состоящей из операций с множествами и самими типами множеств. Кроме того, метод предоставляет возможности для создания новых множеств с помощью пользовательских команд. С другой стороны, в отличие от формата SDF, этот формат специализируется только на обработке CSG, а не на чем-то общем. Еще одним недостатком является то, что программная реализация MPR может обрабатывать небольшое количество фигур. Поэтому сложно визуализировать что-то большое и с большим количеством деталей.

Визуализация геометрии CSG остается интересным направлением для будущих исследований и разработок. Наиболее многообещающие результаты можно увидеть в [25]. Объединив идеи [25][26] и [24], можно было бы разработать передовую технологию.

3.4 Метрики

Качество изображения может ухудшаться из-за различных искажений в процессе получения и обработки, включая шум, размытие и артефакты сжатия. Объективные метрики качества изображений позволяют количественно оценить степень искажений и часто используются при разработке алгоритмов обработки изображений, оптимизации кодеков и контроле качества мультимедийного контента.

PSNR - наиболее распространенная объективная метрика, измеряющая отношение максимального значения сигнала к мощности шума. Она широко используется благодаря простоте вычисления, однако имеет существенный недостаток - слабую корреляцию с человеческим восприятием качества. Существует соответствие между значениями PSNR и субъективными оценками MOS (Mean Opinion Score).

SSIM [27] был предложен как метрика, лучше согласующаяся с человеческим восприятием. В отличие от PSNR, SSIM оценивает визуальное качество на основе трех компонентов: яркости, контрастности и структурной информации.

FLIP [28] - относительно новая метрика, разработанная компанией NVIDIA специально для оценки качества рендеринга. Ключевая особенность [28] - моделирование восприятия различий при быстром переключении между двумя изображениями. FLIP учитывает особенности человеческого зрения, включая чувствительность к контрастным функциям, обнаружение признаков и использует перцептивно-однородное цветовое пространство. Метрика также уделяет особое внимание точечным структурам, таким как "светлячки"(fireflies) - изолированным пикселям с резко отличающимся цветом. Результатом работы FLIP является карта различий, где значения пропорциональны воспринимаемой человеком разнице между изображениями. Эта метрика показала высокую эффективность не только для рендеринговых изображений, но и для естественных изображений и изображений, сгенерированных нейронными сетями.

LPIPS [29] представляет современный подход к оценке качества изображений с использованием глубоких нейронных сетей. Вместо разработки формул, моделирующих человеческое восприятие, [29] "обучается"непосредственно на данных человеческих оценок. Метрика измеряет расстояние между признаками, извлекаемыми из эталонного и оцениваемого изображений с помощью предобученной нейронной сети (например, AlexNet [30]). Более высокие значения LPIPS означают большие различия между изображениями.

Важно отметить, что не существует универсальной объективной метрики, подходящей для решения всех задач. Эффективность любой метрики зависит от динамичности содержания изображения, сложности сцены и качества эталонного изображения. В критически важных приложениях рекомендуется использовать комбинацию различных метрик или проводить субъективное тестирование.

3.5 Датасеты

Реальные датасеты для SDF-методов с прямым освещением различаются масштабом, контролем параметров материалов и специализацией под конкретные задачи. DiLiGenT102[31] (100 объектов, 10 материалов × 10 форм) предлагает систематический контроль формы и отражательных свойств - от диффузных пластиков до анизотропных металлов и полупрозрачного акрила. Его стоит использовать для анализа влияния материала на точность реконструкции, особенно при разработке алгоритмов, чувствительных к BRDF. LUCES-MV[32] (15 промышленных объектов) акцентирован на мультивьювой съёмке с калиброванными LED-источниками, что делает его идеальным для тестирования методов слияния данных с нескольких ракурсов в условиях ближнего поля. OpenIllumination[33] содержит более 60 моделей и выделяется детальным контролем за освещением - каждый источник активируется изолированно, позволяя изучать разделение геометрии и освещения. Этот датасет критически важен для валидации нейросетей обратного рендеринга. DiLiGenRT[34] (54 гемисферы) уникален количественным контролем шероховатости (Ra 0.05–12.5 мкм) и прозрачности (1–10 мм), что необходимо для анализа subsurface scattering в биомедицинских приложениях или материалах типа матового стекла. Меньшие датасеты вроде Gourd&Apple[35], где всего 3 объекта, полезны для быстрого тестирования методов на сценах с сильной неоднородностью поверхностей (неровности кожуры фруктов), но их ограниченный масштаб делает выводы менее статистически значимыми. Если идет сравнение нескольких рендер-систем, работающих по-разному, то есть смысл задать ограничения на материалы, источники света и геометрию, как сделали в [36]. Тогда с помощью подобной стандартизации появляется возможность сравнивать смещенные и несмещенные алгоритмы, тестировать аппаратную независимость через OpenCL/CUDA-реализации и выявлять скрытые компромиссы между скоростью рендера и физической корректностью.

4 Методология

4.1 Выбор метрик

В рамках исследования методов представления функций дистанции со знаком (SDF) критически важным этапом является выбор метрик, позволяющих объективно оценить компромисс между качеством реконструкции, производительностью и эффективностью сжатия. Предлагаемый набор метрик - оценка точности, размер модели и время рендеринга - формирует комплексную систему оценки, учитывающую ключевые требования задач компьютерной графики: точность воспроизведения геометрии, возможность работы в реальном времени и компактность представления данных.

1. **Метрики точности**:

   - **Пиковое отношение сигнала к шуму (PSNR)**

     PSNR служит количественной мерой точности реконструкции поверхности относительно эталонного меша. Для SDF, задающих расстояние до поверхности со знаком, PSNR вычисляется через среднеквадратичную ошибку (MSE) между эталонными и восстановленными значениями расстояний:

     $$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(d_{\text{ref}}^{(i)} - d_{\text{pred}}^{(i)})^2$$

     $$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right)$$

     где MAX - максимальное расстояние в сцене, N - количество пробных точек. Для 8-битных данных $MAX = 255$, но в 3D-контексте значение определяется масштабом сцены. Высокий PSNR (>30 дБ) указывает на близость восстановленной геометрии к эталону, что критично для приложений, требующих точного воспроизведения деталей (медицинская визуализация, инженерное моделирование).

     Ограничение PSNR - слабая корреляция с субъективным восприятием качества при наличии структурных искажений. Однако для задач, где приоритетом является метрическая точность (напр., CAD-модели), PSNR остаётся эталонным инструментом.

   - **FLIP**

     Метрика моделирует три фундаментальных свойства человеческого зрительного анализа: **цветовое восприятие**, **контрастная чувтсвительность**, **обнаружение особенностей**. FLIP учитывает **плотность пикселей** дисплея (PPI), **расстояние до наблюдателя** и **динамический диапозон** изображения. Также эта метрика стандартизирована, есть аппаратная оптимизация и она открыта в виде исходного кода. Таким образом, её можно

считать референсным выбором в области компьютерной графики для замера точности методов рендера.

- **SSIM**

  вычисляется по формуле:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) \cdot (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) \cdot (\sigma_x^2 + \sigma_y^2 + C_2)}$$

  где $\mu_x,\ \mu_y$ - средняя яркость, $\sigma_x,\ \sigma_y$ - дисперсии яркости, σ_{xy} - ковариация между яркостями двух изображений, C_1, C_2 - константы для предотвращения ситуации деления на 0. SSIM принимает значения от 0 до 1, где 1 означает полное совпадение с эталоном.

- **LPIPS**

  Работает на основе нейросети, которая принимает на вход два изображения и выдает оценку качества. LPIPS демонстрирует лучшую корреляцию с человеческим восприятием по сравнению с традиционными метриками, особенно при оценке результатов работы генеративных моделей.

2. **Размер модели**

   Эффективность сжатия оценивается через объём памяти, требуемый для хранения параметров представления. Для нейросетевых методов типа DeepSDF (ссылка на статью) размер определяется числом параметров сети. Традиционные SDF в воксельном представлении требуют $O(n^3)$ памяти.

   Сравнение размеров сконвертированных моделей позволяет оценить выигрыш от использования компактных представлений.

3. **Время рендеринга**

   Производительность рендеринга измеряется временем генерации кадра при использовании:

   - **Ray marching**:

$$t = O(N_{\text{rays}} \cdot t_{\text{SDF}})$$

     где t_{SDF} - время вычисления SDF в точке.

   - **Ray tracing** (аналитический поиск персечения):

$$t = O(N_{rays} \cdot \log M),\ M - \text{количество треугольников}$$

   Нейросетевые SDF увеличивают t_{SDF} из-за вычислений прямого распространения, но компенсируют это сокращением n благодаря непрерывности представления.

   Оптимизации типа иерархических структур ускорения или кэширования значений SDF критичны для достижения частоты кадров >30 FPS.

4. **Синтез требований**

   Выбранные метрики образуют треугольник компромиссов:

   - Точность $\Leftrightarrow$ Размер модели: Повышение точности через увеличение разрешения или сложности модели ведёт к росту объёма данных.
   - Точность $\Leftrightarrow$ Время рендеринга: Точные SDF с высокой детализацией требуют больше вычислений на луч.
   - Размер модели $\Leftrightarrow$ Время рендеринга: Сжатые представления, например, нейросетевые уменьшают объём данных, но могут замедлять рендеринг из-за вычислений параметров.

Предложенный набор метрик позволяет количественно оценить применимость методов SDF для задач, требующих баланса между точностью, производительностью и эффективностью использования ресурсов. Дальнейшие исследования могут дополнить систему другими метриками, но текущий выбор оптимален для инженерного анализа.

4.2 Выбор датасета

Для всесторонней оценки методов представления функций дистанции со знаком (SDF) разработана двухуровневая система тестирования, использующая комплементарные датасеты. Такой подход позволяет анализировать как базовую эффективность методов в штатных условиях, так и их устойчивость к экстремальным нагрузкам, что критично для валидации промышленных решений.

Эти две коллекции обеспечивают:

- Стандартизированную геометрию с размерами, нормализованными в диапазоне $[-0.5, 0.5]^3$, что устраняет артефакты масштабирования
- Водонепроницаемые меши (watertight), гарантирующие корректность расчёта SDF через алгоритм ray marching
- Низкую полигональную сложность (меньше 10^4 треугольников на модель), позволяющую тестировать базовые свойства методов без перегрузки вычислительных ресурсов

1. Базовый датасет: верификация функциональности

   В этом сравнении использовался более простой набор моделей, на которых методы, задействованные в сравнении, не нарушались. Цель состояла в том, чтобы найти перспективный существующий метод, даже если его следует доработать для устранения артефактов в целевых моделях.

   Для генерации SDF используются полигональные сетки из датасета. Это обеспечивает воспроизводимость экспериментов и прямую сопоставимость метрик PSNR между методами.

2. Стресс-тестовый датасет: анализ масштабируемости

   В качестве эталонного набора выбраны модели из ABC-Dataset, содержащий 1 млн CAD-моделей соответственно. Выбранные модели уже сложнее, поэтому для на них тестировались не все методы, а только наиболее эффективные. Также некоторые варианты, например, имеют тонкие поверхности, что является серьезным испытанием для SDF методов.

   Таким образом, собранные модели проверяют методы на способность рендерить большие модели, обладающие сложной для SDF структурой.

Использование двух комплементарных датасетов обеспечивает многоуровневую валидацию методов представления SDF, критически важную для оценки их практической применимости. Базовый датасет (напр., модели двигателей, скелетов) служит контролируемой средой для верификации корректности алгоритмов в идеализированных условиях, где метрики PSNR и время рендеринга отражают фундаментальные свойства методов без искажающего влияния шумов или артефактов. Это позволяет сравнить

методы на стандартизированной базе, обеспечивая воспроизводимость результатов и прямую сопоставимость с предыдущими исследованиями.

Одновременно стресс-тестовый датасет выявляет пределы масштабируемости, демонстрируя, как методы ведут себя при обработке неводонепроницаемых мешей, моделей с $>10^8$ треугольников. Такая комбинация позволяет оценить устойчивость алгоритмов к реальным вызовам.

Синтез результатов двух датасетов формирует треугольник компромиссов "точность-производительность-память позволяя выбрать оптимальный метод для конкретной задачи: нейронные методы для компактного представления стандартных объектов против гибридных сеток для промышленных CAD-моделей. Без многоуровневого тестирования сохраняется риск ложной оптимизации - улучшения метрик на синтетических данных при деградации на реальных сценах.

Первый датасет	
Модель	**Размер, МБ**
Manifold	286.6
Bunny	10.37
Skeleton	196.8
Roadbike	163.3
Buddha	99.82
Crankcase	141.0
Cylinder	71.47
Stone	124.0
Buggy	634.6
Dragon	72.76
Radiator	371.4
WaterPump	72.48

Второй датасет (выборка из ABC)	
Модель	**Размер, МБ**
ABC_80006	154
ABC_83870	199
ABC_88060	84
ABC_88828	20
ABC_515447	58
ABC_687231	108
ABC_701584	45
ABC_702109	122
bulldozer	1511
DEMONSTRATOR	591

Таблица 1: Использованные модели

4.3 Примеры моделей

Рис. 1: Примеры подобранных моделей

4.4 Выбор методов SDF

Выбраны представления, специализирующиеся на форме SDF поверхности, а не на операциях над ней. Следовательно, были предложены следующие реализации:

Авторские методы	Реализованные методы
Decimation [10]	SVO [19]
VDB [17]	SBS [16]
MPR CSG [24]	AASDF [21]
NGLOD [37]	Analytic SVO [20]
Instant NGP [38]	
CSG SComTree(1)	

Таблица 2: Выбранные методы, (1) - SComTree является разработкой лаборатории компьютерной графики и мультимедиа факультета ВМК МГУ

Левая колонка показывает, какие методы были взяты в виде готового авторского кода, который потом дорабатывался для сравнительного вида. Правая колонка отвечает за те методы, которые были реализованы заново по описаниям из их статей.

4.5 Приведение в сравнимый вид

Выбранные методы имеют множество параметров. У некоторых это размер вокселей, в углах которых хранятся значения SDF, у других это параметры энкодеров, нужных для нейронных моделей. В общем, параметров очень много и они разные. Для приведения методов в вид, когда они примерно похожи по памяти, чтобы дальше посчитать время рендера и точность, потребовавлось экспрементально подбирать параметры.

В итоге, получилась таблица параметров, которые позволяют объективно сравнивать разные методы SDF:

Примерный размер	**INGP**				**MESH LOD**	**VDB**		**NGLOD**	**SDF SBS**		**AASDF**	
	n	f	l	r	p	depth	w	nl	depth	br	depth	br
50 kb	3	2	15	16	0.00125	6	2	1	4	3	5	3
300 kb	4	2	15	16	0.0025	7	2	2	5	3	7	3
2 Mb	4	2	18	6	0.01	8	2	3	6	3	8	3
20 Mb	16	2	19	16	0.1	9	2	4	7	3	9	3
40 Mb	16	4	19	16	0.2	10	2	5	8	3	10	3

Таблица 3: Экспрементально подобранные параметры для каждого заданного размера. n - n_levels, f - n_features_per_level, l - log2_hashmap_size, r - base_resolution, p - decimate percentage, nl - lods number, br - brick size.

Эта таблица также открывает возможности изобразить полученные данные в виде графиков, которые были бы невозможными без неё. Вот пример таких графиков:

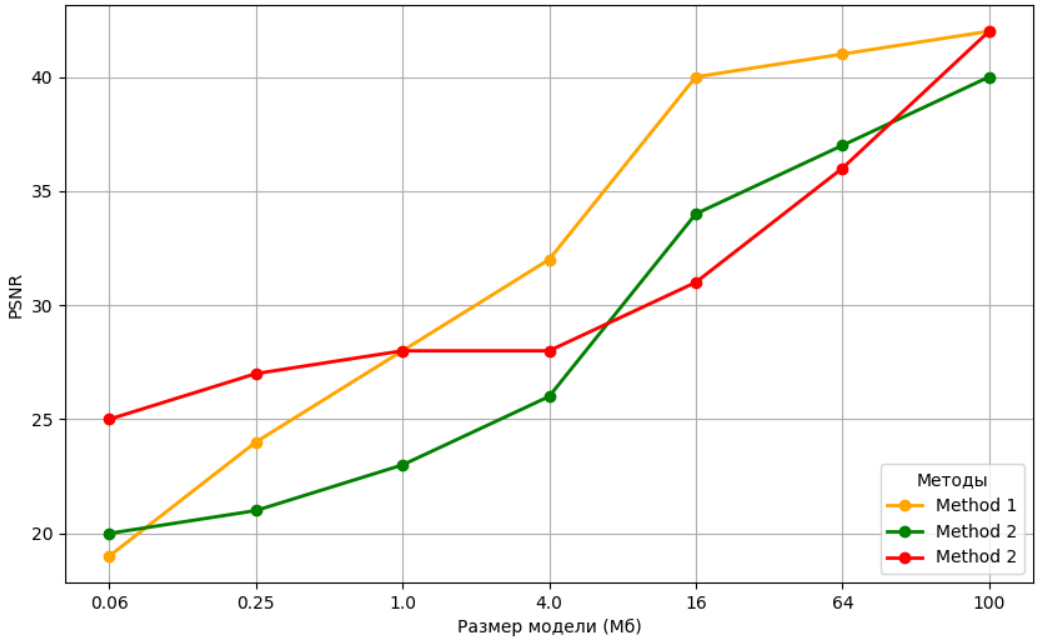


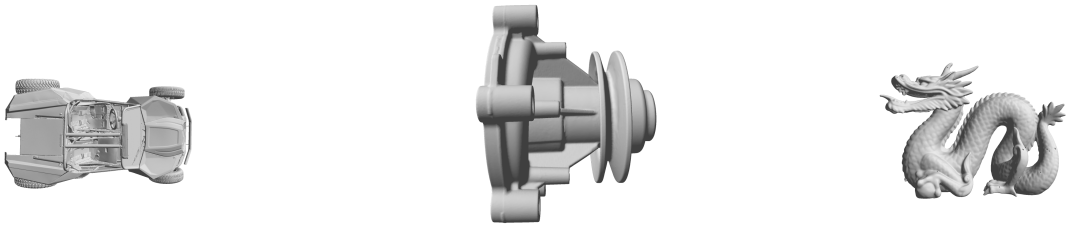
Рис. 2: Пример графиков

4.6 Освещение

В данной работе используется один точечный источник света и простая модель освещения, так как задача - в первую очередь сравнить полученную каждым методом общую форму объекта, не учитывая работу с материалами, множеством разных источников света, отражениями и т.д.

Выбрана модель освещения Ламберта, работающая по следующему принципу:

1. Если SDF(p) = 0, то p - точка на поверхности модели, и нормаль к ней это - $n = \frac{\vec{l}}{\|l\|},\ l = \frac{dSDF(p)}{dp}$.
2. Цвет рассчитывается, как $c = max(0.1, \langle n, d \rangle)$, d - вектор, направленный от точечного источника света к точке p.



(a) Buggy (b) Water Pump (c) Dragon

Рис. 3: Примеры использования реализованного освещения

5 Практическая реализация

5.1 Структура бенчмарка

В ходе проделанной работы был создан бенчмарк, который способен сравнивать, как методы, которые пользователь может реализовать сам, так и готовые решения, предоставляемые в виде готовых проектов. Есть возможность переносить написанные методы на графический ускоритель, используя средства Kernel Slicer [ссылка] для генерации кода для API Vulkan из C++ кода. Для такой трансформации кода написанные методы должны удовлетворять определенным тредованиям этого инструмента.

Сцена может рисоваться как в режиме Мультирендера [ссылка на LiteRT], который использует стандартную трассировку и шагающую трассировку лучей для поиска пересечения с поверхностью, так и рендер систему HydraCore3 [ссылка], разрабатываемую в лаборатории компьютерной графики и мультимедиа факультета ВМК МГУ, которая способна рендерить реалистичные сцены за счет использования трассировки пути. На рисунке 4 показана схема работы бенчмарка.

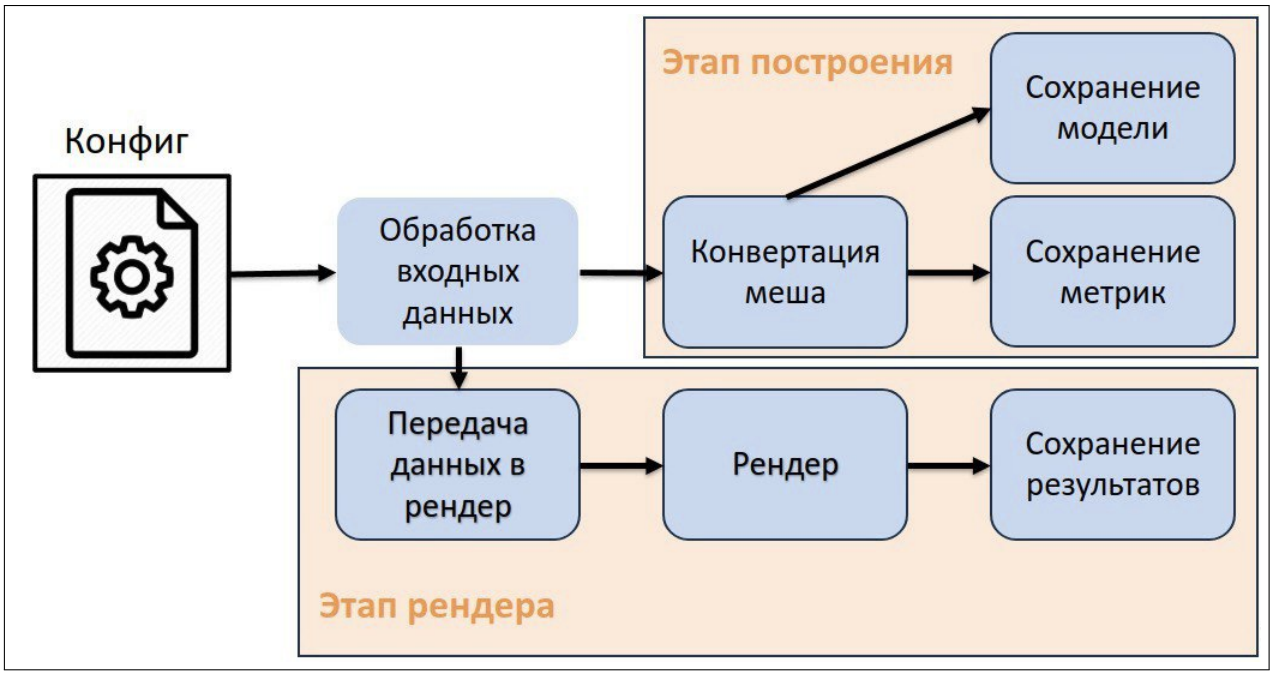


Рис. 4: Схема реализации сравнений методов между собой

Работа бенчмарка идет в два этапа:

1. Сборка каждым методом всех моделей с конкретными параметрами рендера. Полученные данные сохраняются в отдельные папки, откуда будут считываться во время рендера. Также записывается время конвертации, размер и другие данные в специальный .csv файл.

2. Во время рендера из каждой папки берутся данные и рендерятся. Результат сравнивается с референсным методом (по умолчанию это полигональная сетка) и сохраняется в .csv файл.

5.2 Возникшие проблемы

Во время написания программы возниклом множество проблем. Их можно разделить на 4 группы:

1. Методы, имеющие открытую программную реализацию, часто не имели нужных входных параметров. Например, позиция камеры, позиция источника света. В некоторых методах модель нормализуется в куб $[-1, 1]^3$, а не $[-0.5, 0.5]^3$, что давало неверную картинку. Аналогичную проблему вызывали разные настройки камеры, потому что данные о ней могли по-разному храниться и использоваться. Учитывая, что изначально метод мог вообще не принимать на вход информацию о ней, то приходилось это исправлять. Другой проблемой было освещение, потому что, например, в нескольких методах используется модель физического освещения, а в других обходятся обычным рассчетом мест затенения, поэтому приходилось убирать такую сложную модель освещения и заменять на более простую, чтобы полученные картинки совпадали по теням и цвету.

2. Возникла проблема организации хранения данных, потому что очень много параметров, которые меняют выходной результат. Таким образом, на первом этапе бенчмарк сохраняет в отдельную папку бинарный файл с моделью и .xml файл, который используется для её дальнейшей загрузки в выбранную систему рендера. На втором этапе создается другая папка, которая будет содержать отрендеренные изображения. На этом шаге сначала рисуется эталонный метод, чтобы другие реализации могли сразу с ним сравниваться. Все полученные результаты метрик записываются в третью папку. Выбранный подход позволяет получить на каждом этапе простую иерархию данных.

3. При добавлении метода "Массово-параллельный рендеринг сложных неявных поверхностей замкнутой формы"(MPR) пришлось решать проблему, связанную с тем, что его было не с чем сравнивать, потому что он рендерит конструктивную сплошную геометрию, представленную в виде дерева, а не полигональной сеткой. Для этого метода пришлось реализовывать свое синтаксическое дерево,

которое уже могло использоваться, как функция дистанции. Следовательно, появился способ сравнивать SDF методы с методами, реализующими конструктивную сплошную геометрию. Такой подход позволяет использовать бенчмарк для более широкого класса методов, которые изначально могут не иметь возможность сравнения с уже добавленными вариантами.

4. Самая сложная возникшая задача - привести методы в сравнимый вид, потому что они все по-разному программно устроены и зависят от разных параметров, оптимальный подбор которых дал бы объективные графики сравнения. Так как хотелось бы получить модели примерно одинакового веса, чтобы методы могли честно соревноваться в скорости рендера и качестве полученных картинок.

6 Эксперименты и результаты

6.1 Общие результаты

На выбранных датасетах была проведена серия экспериментов. Получены данные о точности, времени рендера и занимаемой памяти каждого метода.

Сцена	INGP					MESH LOD					AASDF					NGLOD					SDF SBS					VDB				
	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t
Budda	24.71	0.57	0.32	0.56	0.99	29.41	0.21	0.34	0.25	0.49	36.41	0.2	0.23	0.26	3.00	29.34	0.17	0.18	0.23	1641.0	32.16	0.24	0.27	0.31	1.98	35.25	0.15	0.16	0.2	1.21
Buggy	25.82	0.35	0.43	0.39	1.58	25.59	0.3	0.62	0.3	0.65	29.00	0.23	0.54	0.73	1.19	19.77	0.33	0.5	0.83	1813.11	23.50	0.31	0.32	0.23	1.58	24.30	0.61	0.81	0.31	2.40
Bunny	24.12	0.41	0.56	0.33	1.02	41.41	0.54	0.13	0.4	0.85	40.30	0.22	0.43	0.21	3.54	35.59	0.63	0.34	0.82	1665.35	40.41	0.23	0.65	0.11	0.10	42.98	0.70	0.11	0.13	3.80
Crankcase	21.31	0.64	0.87	0.73	1.06	34.47	0.15	0.94	0.83	0.08	33.09	0.21	0.33	0.91	3.87	36.69	0.83	0.44	0.87	1280.27	37.65	0.64	0.08	0.55	0.61	35.59	0.14	0.43	0.67	0.97
Cylinder	22.81	0.34	0.03	0.12	1.54	31.25	0.01	0.23	0.81	0.61	25.76	0.71	0.51	0.30	1.66	33.79	0.26	0.51	0.83	1421.02	21.78	0.62	0.29	0.35	0.53	31.49	0.26	0.64	0.20	0.38
Dragon	20.19	0.33	0.63	0.43	1.32	33.91	0.53	0.34	0.93	0.85	27.08	0.12	0.41	0.01	4.80	35.25	0.04	0.83	0.5	1455.56	21.78	0.63	0.33	0.81	1.66	32.91	0.19	0.53	0.41	1.10
Manifold	23.43	0.57	0.32	0.56	1.65	30.32	0.15	0.94	0.83	0.38	29.24	0.22	0.43	0.21	0.90	21.35	0.38	0.44	0.78	1727.27	33.35	0.33	0.5	0.83	0.89	40.31	0.61	0.81	0.31	1.23
Stone	24.11	0.21	0.34	0.25	1.88	31.49	0.3	0.36	0.34	0.45	32.68	0.45	0.31	0.4	4.09	35.88	0.31	0.23	0.12	1479.10	33.60	0.46	0.8	0.55	0.93	30.41	0.40	0.72	0.43	1.26
Pump	22.30	0.35	0.43	0.39	0.47	33.13	0.01	0.23	0.81	0.74	34.95	0.54	0.13	0.4	0.51	31.08	0.24	0.27	0.31	1002.31	31.64	0.83	0.44	0.87	3.88	37.12	0.23	0.65	0.11	1.75

Таблица 4: Сравнение методов на базовом датасете. Все модели приведены к размеру 2 Мб. Рендеринг в разрешении 1400 × 1400 на графическом процессоре Nvidia RTX 4080.

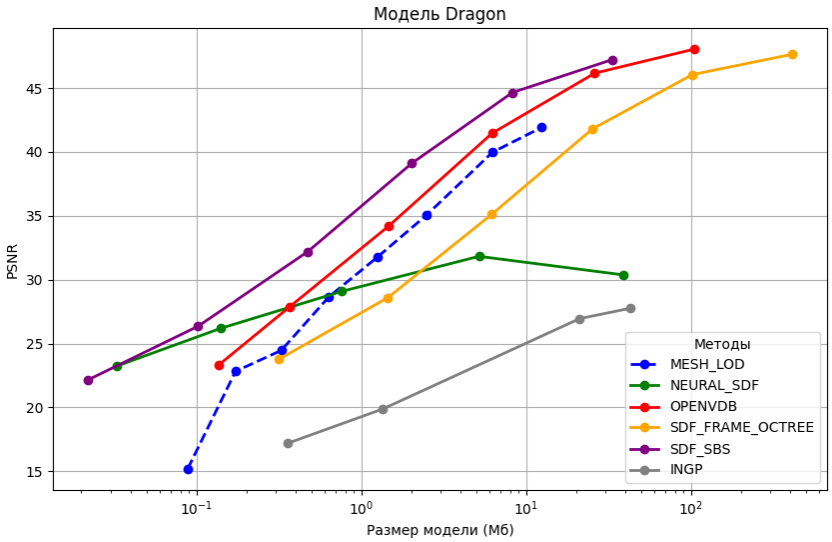
Сжатая сцена в Мб	INGP					MESH LOD					AASDF					NGLOD					SDF SBS					VDB				
	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t	PSNR	FLIP	SSIM	LPIPS	t
ABC_80006 (15.4)	24.4373	0.3	0.36	0.34	2.0208	20.8237	0.4	0.48	0.5	1.25595	30.0934	0.2	0.23	0.26	1.38029	32.9684	0.17	0.18	0.23	729.374	29.6357	0.24	0.27	0.31	1.39803	33.2679	0.15	0.16	0.2	1.39631
ABC_83870 (19.9)	21.9288	0.35	0.43	0.39	1.5898	33.4161	0.21	0.34	0.25	0.7658	27.4981	0.32	0.46	0.37	1.10991	27.1096	0.33	0.5	0.38	1217.91	27.6569	0.28	0.32	0.32	1.92584	30.7536	0.16	0.18	0.13	2.04484
ABC_88060 (8.4)	25.4127	0.43	0.3	0.21	1.6002	31.6253	0.45	0.31	0.4	0.94485	32.1536	0.22	0.34	0.12	3.56534	31.427	0.36	0.43	0.18	1187.38	33.7094	0.32	0.56	0.11	0.810781	36.724	0.07	0.11	0.31	1.08488
ABC_88828 (2.0)	25.3924	0.56	0.78	0.37	1.5706	21.2852	0.51	0.49	0.38	0.2608	30.956	0.12	0.33	0.19	3.84097	26.6941	0.38	0.44	0.78	953.696	22.4191	0.46	0.8	0.55	0.43615	22.7819	0.41	0.34	0.76	0.6985
ABC_515447 (5.8)	25.0376	0.54	0.65	0.33	1.54	37.2475	0.20	0.12	0.18	0.3861	29.8905	0.10	0.35	0.23	1.40661	29.2517	0.32	0.45	0.83	1093.44	26.0431	0.40	0.72	0.43	0.671531	32.6907	0.02	0.16	0.32	0.832969
ABC_687231 (10.8)	24.386	0.57	0.32	0.56	1.6312	26.6885	0.21	0.34	0.25	0.38195	29.5108	0.21	0.34	0.10	4.58807	28.7878	0.4	0.48	0.5	927.968	31.1062	0.36	0.43	0.18	1.63666	32.2423	0.01	0.25	0.14	1.60022
ABC_701584 (4.5)	23.4694	0.41	0.29	0.60	1.613	28.3015	0.23	0.44	0.27	0.3198	22.3917	0.32	0.46	0.37	0.903047	30.4694	0.38	0.44	0.78	995.088	29.347	0.15	0.16	0.2	0.428219	34.3476	0.51	0.49	0.38	1.28916
ABC_702109 (12.2)	24.2761	0.35	0.43	0.39	1.5688	22.9275	0.3	0.36	0.34	0.43865	31.0398	0.45	0.31	0.4	4.03193	31.4955	0.31	0.23	0.12	829.598	24.6738	0.46	0.8	0.55	0.956313	26.5279	0.40	0.72	0.43	1.26706

Таблица 5: Сравнение методов на стресс-тестовом датасете, сжатом примерно в 10 раз. Рендеринг в разрешении 1400 × 1400 на графическом процессоре Nvidia RTX 4080.

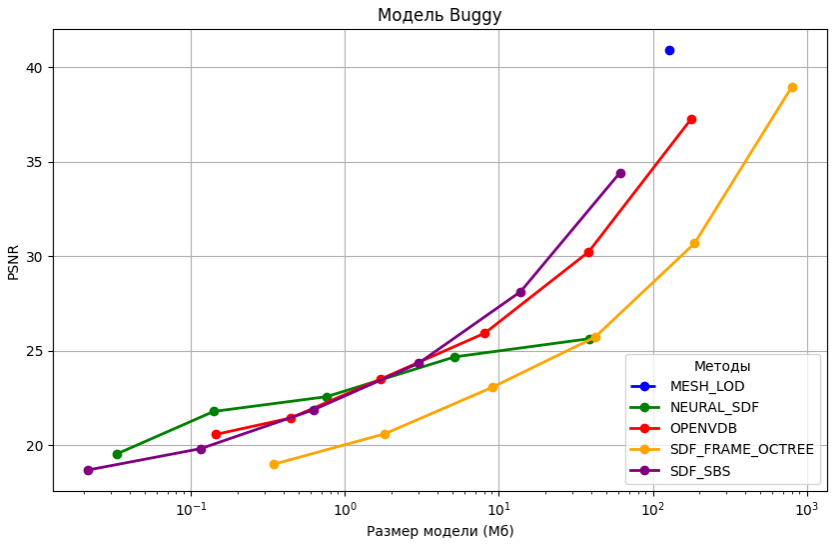
6.2 Компромисс между памятью и качеством

Предложенная методология приведения SDF представлений в сравнимый вид дала возможность графически визуализировать возможности каждой реализации.

График качества памяти всех протестированных методов на моделях «Dragon» и «Buggy». Для каждого метода было создано несколько моделей с разным уровнем детализации, которые представлены в виде точек на этом графике. Каждый метод представлен в виде линии, соединяющей эти точки. Чем ближе линия к верхнему левому углу, тем лучше.



(a) Сравнение методов на модели Dragon



(b) Примеры подобранных моделей

6.3 Классификация

Получив точность каждого метода в зависимости от используемой памяти, можно попробовать сгруппировать их по возможным сценариям использования и составить рейтинг выбранных реализаций. Тогда лучшее разбиение - это:

1. Web - так как в современном мире многое передается через сеть, и некоторые методы могли бы хорошо сжимать, но терять в качестве для быстрой передачи.

2. Графика реального времени - компромисс между размером модели и точностью результата.

3. Кино-фотореализм - в этой области в первую очередь важна точность, поэтому обычно используется много памяти.

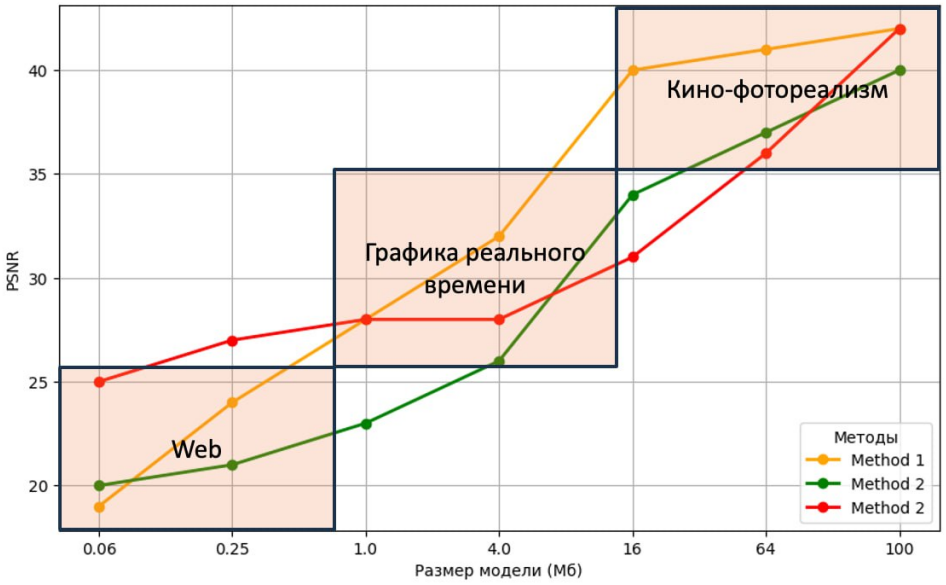


Рис. 6: Пример разделения

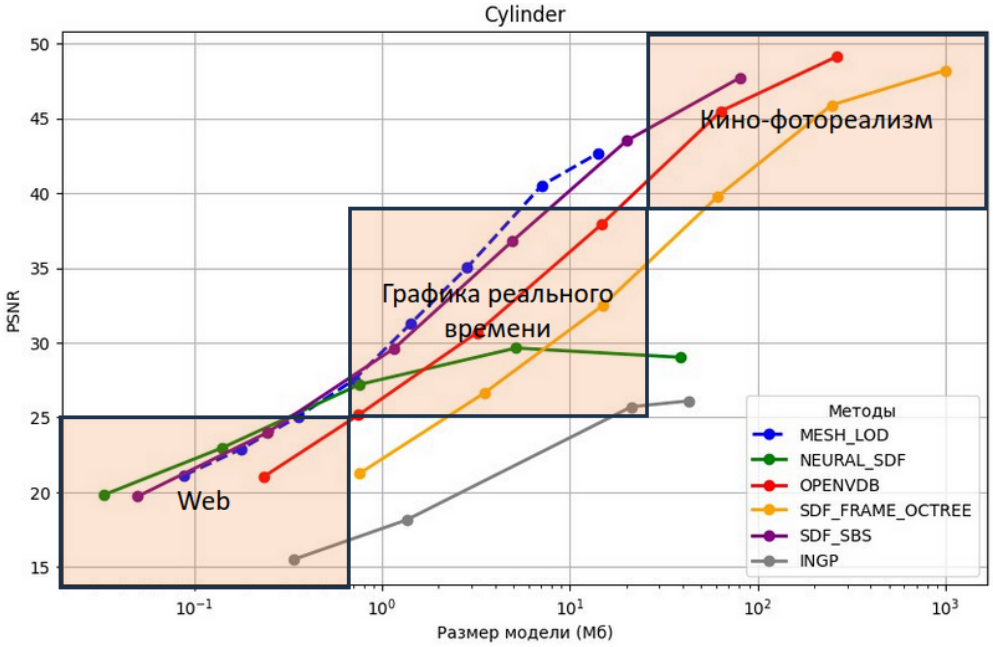


Рис. 7: Распределение методов на группы на модели Cylinder

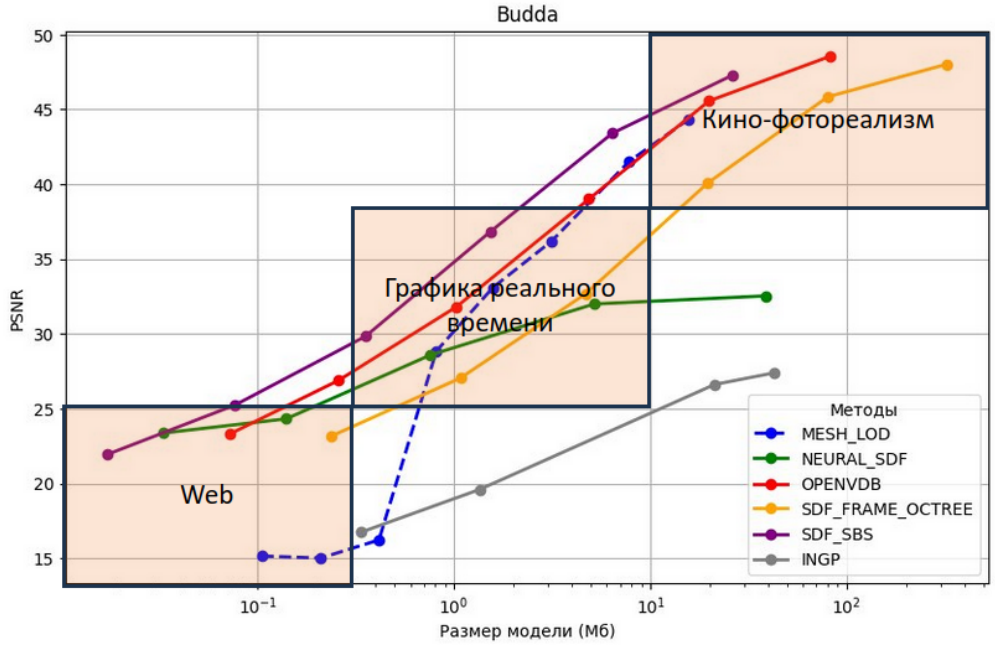


Рис. 8: Распределение методов на группы на модели Budda

Каждому методу в зависимости от его принадлежности к прямоугольнику выставлялся балл:

- 5 - Графика реального времени
- 3 - Кино-фотореализм
- 1 - Web

Получились следующие результаты:

Метод	Баллы
AASDF [21]	52
VDB [17]	50
SBS [16]	48
Decimation [10]	34
NGLOD [37]	28
Instant NGP [38]	15

Таблица 6

7 Заключение

В ходе работы были получены следующие результаты:

- Проведен обзор существующих работ по представлениям функций дистанции со знаком.
- Разработана методология сравнения методов.
- Полученна программная система для сравнительного анализа.
- Проведены экспериментальные сравнения выбранных представлений SDF.
- Проанализированны результаты.

Список литературы

[1] Joshua Barczak, Carsten Benthin, and David McAllister. Dgf: A dense, hardware friendly geometry format for lossily compression meshlets with arbitrary topologies. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 7*, 41, 2024.

[2] Daniel Mlakar, Markus Steinberger, and Dieter Schmalstieg. End-to-end compressed meshlet rendering. *Wiley Online Library, e15002*, 43, 2024.

[3] A. Nikolaev, A. Shcherbakov, and V. Frolov. Mesh compression method with on-the-fly decompression during rasterization and streaming support. *WSCG 2022 Proceedings*, 2022.

[4] Jacob Haydel, Cem Yuksel, and Larry Seiler. Locally-adaptive level-of-detail for hardware-accelerated ray tracing. *ACM Transactions on Graphics*, 42, 2023.

[5] Andrea Maggiordomo, Henry Moreton, and Marco Tarini. Micro-mesh construction. *ACM Trans. Graph*, 42, 2023.

[6] Junjie Xue, Gang Zhao, and Wenlei Xiao. Efficient gpu out-of-core visualization of large-scale cad models with voxel representations. *Advances in Engineering Software*, 99, 2016.

[7] M Wald, GP Johnson, and J Amstutz. Ospray - a cpu ray tracing framework for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23, 2017.

[8] Mark Lee, Brian Green, and Feng Xie. Vectorized production path tracing. *Association for Computing Machinery, New York*, 10, 2017.

[9] Zhen Chen, Zherong Pan, Kui Wu, and Etienne Vouga. Robust low-poly meshing for general 3d models. *ACM Trans. Graph*, 42, 2023.

[10] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 1997.

[11] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Comput. Surv.*, 47, 2015.

[12] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM SIGGRAPH*, 2002.

[13] Venkataram Edavamadathil Sivaram, Tzu-Mao Li, and Ravi Ramamoorthi. Neural geometry fields for meshes. *ACM SIGGRAPH*, 2024.

[14] Gabriel Peyré and Stéphane Mallat. Surface compression with geometric bandelets. *ACM SIGGRAPH*, 2005.

[15] NVIDIA. Rtxmg. NVIDIA, 2025.

[16] Hansson-Söderlund Herman, Evans Alex, and Akenine-Möller Tomas. Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques 11*, 2022.

[17] Ken Museth. High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph. 32*, 2013.

[18] Rama Karl Hoetzlein. Gvdb: raytracing sparse voxel database structures on the gpu. *In Proceedings of High Performance Graphics (Dublin, Ireland)*, 2016.

[19] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. *Association for Computing Machinery*, 2010.

[20] Eduard Pujol and Antonio Chica. Rendering piecewise approximations of sdfs through analytic intersections. *Comput. Graph. 122*, 2024.

[21] Eduard Pujol and Antonio Chica. Adaptive approximation of signed distance fields through piecewise continuous interpolation. *Comput. Graph. 114*, 2023.

[22] Henry A. Boateng and Kyle Bradach. Triquintic interpolation in three dimensions. *J. Comput. Appl. Math. 430*, 2023.

[23] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph. 32*, 2013.

[24] Matthew J. Keeter. Massively parallel rendering of complex closed-form implicit surfaces. *SIGGRAPH 2020*, 2020.

[25] Bogolepov, D.K., D. Ulyanov, V. Turlapov, Sas, and O. Gpu-optimized ray-tracing for constructive solid geometry scenes. *Графикон*, 2016.

[26] Carsten Benthin, Sven Woop, Ingo Wald, and Attila T. Áfra. Improved two-level bvhs using partial re-braiding. *In Proceedings of High Performance Graphics (HPG '17)*, 2017.

[27] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

[28] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. Scaling language-image pre-training via masking. *arXiv:2212.00794v*, 2023.

[29] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR 2018*, 2018.

[30] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, 2012.

[31] Jieji Ren, Feishi Wang, Jiahao Zhang, Qian Zheng, Mingjun Ren, and Boxin Shi. Diligent102: A photometric stereo benchmark dataset with controlled shape and material variation. *IEEE Xplore*, 2022.

[32] Fotios Logothetis, Ignas Budvytis, Stephan Liwicki, and Roberto Cipolla. Luces-mv: A multi-view dataset for near-field point light source photometric stereo. *arXiv*, 2024.

[33] Isabella Liu, Linghao Chen, Ziyang Fu, Liwen Wu, Haian Jin, Zhong Li, Chin Ming Ryan Wong, Yi Xu, Ravi Ramamoorthi, Zexiang Xu, and Hao Su. Openillumination: A multi-illumination dataset for inverse rendering evaluation on real objects. *arXiv*, 2024.

[34] Heng Guo, Jieji Ren, Feishi Wang, Mingjun Ren, and Yasuyuki Matsushita. Diligenrt: A photometric stereo dataset with quantified roughness and translucency. *IEEE Xplore*, 2023.

[35] Chen Guanying, Kai Han, and Kwan-Yee Kenneth Wong. Ps-fcn: A flexible learning framework for photometric stereo. *arXiv*, 2018.

[36] В.А. ФРОЛОВ, Д.С. ПАВЛОВ, М.А. ТРОФИМОВ, П.А. КАЗБЕЕВ, and В.А. ГАЛАКТИОНОВ. ОТКРЫТОЕ СОРЕВНОВАНИЕ РЕНДЕР-СИСТЕМ. *MATHEMATICA MONTISNIGRI*, 2019.

[37] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *In Computer Vision and Pattern Recognition (CVPR), 2021 (Oral)*, 2021.

[38] THOMAS MÜLLER, ALEX EVANS, CHRISTOPH SCHIED, and ALEXANDER KELLER. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (SIGGRAPH 2022)*, 2022.