DOI: 10.15514/ISPRAS-2026-38(3)-34

Сравнение подходов к использованию нейронных моделей в задачах компьютерной графики

Н.А. Милин ORCID: 0009-0004-1627-9445 <nikita.milin@graphics.cs.msu.ru>
Р.О. Родионов, ORCID: 0009-0000-5044-9963 <roman.rodionov@graphics.cs.msu.ru>
В.А. Фролов, ORCID: 0000-0001-8829-9884 <frolov@gin.keldysh.ru>

ИПМ им. М.В. Келдыша РАН,
Россия, 125047, Москва, Миусская пл., д.4.

Аннотация. Данная работа представляет сравнительное исследование производительности различных подходов к применению нейронных моделей в задачах компьютерной графики. Рассмотрены два направления: представление поверхностей с помощью нейронных моделей функции расстояния со знаком (на примере SIREN) и нейронные модели материалов, представленных двунаправленной функцией отражения, (на примере NBRDF). Проанализированы реализации MLP как с помощью кроссплатформенных подходов, так и с использованием специализированного аппаратного ускорения. Результаты экспериментов выявляют существенные различия в производительности между рассматриваемыми подходами. На их основе сформулированы практические рекомендации для разработчиков, планирующих интеграцию нейронных методов в приложения компьютерной графики.

Ключевые слова: компьютерная графика; рендеринг реального времени; нейронные сети; функции дистанции со знаком.

Для цитирования: Милин Н.А., Родионов Р.О., Фролов В.А. Сравнение подходов к исполнению нейронных моделей в задачах компьютерной графики. Труды ИСП РАН, том 38, вып. 3, часть 3, 2026 г., стр. 39–48. DOI: 10.15514/ISPRAS–2026–38(3)–34.

Благодарности: Исследование выполнено при поддержке Российского научного фонда, проект № 25-11-00054.

Comparison of approaches to the use of neural model in computer graphics

N.A. Milin, ORCID: 0009-0004-1627-9445 <nikita.milin@graphics.cs.msu.ru>
R.O. Rodionov, ORCID: 0009-0000-5044-9963 <roman.rodionov@graphics.cs.msu.ru>
V.A. Frolov, ORCID: 0000-0001-8829-9884 <frolov@gin.keldysh.ru>

Keldysh Institute of Applied Mathematics,
Miusskaya sq., 4, Moscow, 125047, Russia.

Abstract. This work provides a comparative performance study for different approaches of neural-based methods in computer graphics. Two directions are considered: representation of surfaces via neural models of the signed distance function (using SIREN as an example) and neural models of materials represented by a bidirectional reflectance function (using NBRDF as an example). This work includes analysis of both cross-platform and hardware accelerated MLP implementations. Experimental results reveal significant performance differences between investigated approaches. Based on that, the paper formulates practical recommendations for developers who plan to integrate neural methods in computer graphics applications.

Keywords: computer graphics; real-time rendering; neural networks; signed distance functions.

For citation: Milin N.A., Rodionov R.O., Frolov V.A. Comparison of approaches to neural model execution in computer graphics. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 3, 2026, pp. 39-48 (in Russian). DOI: 10.15514/ISPRAS-2026-38(3)-34.

Acknowledgements. The study was supported by a grant from the Russian Science Foundation № 25-11-00054.

1. Введение

Технологии Искусственного Интеллекта (ИИ) все чаще находят применение в современной компьютерной графике. Нейронные подходы к представлению геометрии и материалов открывают новые возможности для создания высококачественных визуальных эффектов, такие как моделирование многослойных материалов [1] и геометрии [2]. Кроме того, нейронные модели показали себя эффективными в обратном рендеринге [3-4].

Однако интеграция нейронных методов в графические приложения реального времени может вызвать существенное снижение производительности [1]. Выполнение нейронных сетей на GPU требует эффективного использования архитектурных особенностей современных графических процессоров и тщательной оптимизации алгоритмов.

Первые результаты исследования докладывались на международной конференции по компьютерной графике и зрению Графикон-2025 [5]. В данной статье приводятся новые результаты, а также делаются обобщения, выводы и практические рекомендации для разработчиков, планирующих интеграцию нейронных методов в приложения компьютерной графики.

2. Обзор существующих решений

В современных задачах компьютерной графики применяются разнообразные архитектуры нейронных сетей, каждая из которых адаптирована под специфические требования графических вычислений. Многослойные перцептроны (MLP) широко используются в задачах трассировки лучей и расчета освещения благодаря своей способности аппроксимировать сложные функции излучения и отражения [6]. Сверточные нейронные сети и рекуррентные архитектуры находят широкое применение при решении задачи шумоподавления изображений, полученных методом трассировки путей [7-8]. Архитектуры на основе трансформеров адаптируются для задач синтеза изображений, где важна способность к обработке последовательностей различной длины [9-10].

Значительная часть исследовательских работ в области компьютерной графики опирается на библиотеки глубокого обучения общего назначения, такие как PyTorch, TensorFlow. Это обусловлено простотой их использования и обширным набором инструментов разработки. Работы [2, 8, 10-11] используют PyTorch, однако его применение в контексте графики реального времени сопряжено с существенными накладными расходами, связанными с интерпретацией Python-кода и невозможностью низкоуровневой ручной оптимизации. Решения, основанные на технологии CUDA используемые в работах [6, 12], отличаются высокой степенью оптимизации за счет адаптации к аппаратным возможностям устройств NVIDIA, однако существенным недостатком является привязка к конкретной экосистеме, что ограничивает портируемость решения на другие платформы и создает зависимость от проприетарных технологий.

Отдельно отметим, что традиционные библиотеки глубокого обучения оптимизированы для согласованного выполнения нейронной сети, где входные данные собраны в один набор (батч), и высокая пропускная способность достигается за счет параллельной обработки большого количества образцов. Однако в графическом конвейере может потребоваться выполнить нейронную модель только для части пикселов или выполнить несколько различных моделей с разными данными на входе. Как отмечено в работе [1], инструменты для интеграции нейронных сетей с потенциально расходящимся исполнением в языки программирования шейдеров, такие как GLSL или HLSL, практически отсутствуют. Помимо этого, для повышения производительности предпочтительно использовать аппаратное ускорение матричного умножения, доступное в устройствах NVIDIA, AMD и Intel. Далее обсудим, как решать эти проблемы, и сравним несколько перспективных методов выполнения нейронных сетей в задачах компьютерной графики.

3. Описание исследуемых методов

3.1 Компилируемые веса

Один из прямых подходов к использованию нейронных сетей в рендеринге – сохранить все веса нейронной сети в шейдере. В этом случае они будут доступны компилятору, что позволит поместить их в кэш и провести дополнительные оптимизации, такие как разворачивание циклов и удаление избыточных инструкций. Недостатком этого метода является необходимость создания отдельного шейдера для каждой используемой нейронной сети. Это усложняет их использование при выполнении большого числа моделей в течение рендеринга одной сцены.

3.2 Веса в буфере

Другой подход – загружать параметры нейронной сети в глобальную память GPU динамически, а доступ к ним осуществлять через индексирование внутри шейдера. Это позволяет использовать код одного шейдера для моделей различных размеров, передавая данные об архитектуре сети как константы. Существенным недостатком является то, что каждый отдельный поток многократно загружает параметры модели из видеопамяти. Улучшением могло бы служить использование быстрой разделяемой памяти путем предварительной загрузки небольшого числа параметров в нее и выполнение блочного матричного умножения, но данная возможность доступна только в вычислительных (compute) шейдерах.

3.3 Аппаратное ускорение с использованием Cooperative Vectors

Расширение Cooperative Vectors для современных графических API [13] предоставляет высокоуровневый инструмент для описания матричного умножения в коде шейдера, в котором компилятор отвечает за объединение входных векторов с разных потоков в матрицу

для эффективного перемножения с использованием тензорных ядер. С его появлением стала возможной реализация прямого прохода нейронной сети, совместимая с существующей моделью программирования графических шейдеров. Расширение не требует, чтобы данные в разных потоках были одинаковые, что позволяет обрабатывать несколько нейронных моделей в одном шейдере. Скорость выполнения повышается за счет неявной загрузки параметров модели по необходимости и выполнения умножения матриц и активаций слоев с аппаратным ускорением. Данный метод требует поддержки расширения в графическом драйвере.

Вышеперечисленные методы выбраны для исследования по следующим причинам. Методы 3.1 и 3.2 могут быть реализованы при помощи стандартных операций графических API, поэтому применимы на большинстве платформ, включая мобильные. При этом первый теоретически способен показать лучшую производительность, если компилятору удастся расположить параметры модели в более быстрой памяти. Второй метод не зависит от самих параметров и предоставляет возможность их динамической загрузки. Метод 3.3 легко встраивается в существующий конвейер рендеринга простой заменой аналитических вычислений на оценку нейронной модели. Он предоставляет возможность выполнений нейронных сетей во всех типах шейдеров, в том числе в конвейере трассировки лучей. В методе 3.3 также поддерживается объединение нескольких последовательных операций, таких как умножение матрицы входных данных на матрицу весов, прибавление матрицы сдвига, применение функции активации, что позволяет избежать промежуточных преобразований данных для их непрерывной обработки на тензорных ядрах.

4. Результаты

4.1 Общее описание экспериментов

В ходе экспериментов была проверена работа каждого из методов для разных размеров нейронных моделей. Методы 3.1, 3.2 и 3.3 были реализованы с использованием Vulkan API, в методе 3.3 дополнительно включено расширение VK_NV_cooperative_vector. В качестве языка для написания графических шейдеров использовался язык Slang с открытым исходным кодом [14]. Для измерения производительности применялись временные метки Vulkan API. Все эксперименты проводились на видеокарте NVIDIA RTX 3080 10GB с версией драйвера 572.42 в разрешении 1920×1080, параметры моделей были представлены в числах с плавающей точкой половинной точности (float16). Для оценки и сравнения методов выполнения нейронных моделей были выбраны следующие задачи:

4.2 Представление поверхностей с помощью нейронных SDF

Функция расстояния со знаком в трёхмерном пространстве – это отображение $R^3 \to R$, показывающее расстояние от данной точки до границы некоторого подмножества пространства. Знак функции определяет, находится ли точка внутри или снаружи множества. Такие функции являются универсальным способом представления геометрии, способным аппроксимировать поверхности, заданные в ином виде, например, воксели, сетки треугольников и других. Традиционно SDF задаются аналитически при помощи технологии конструктивной блочной геометрии, где поверхность строится из примитивов и операций над ними. Хотя этим способом можно задать довольно сложные поверхности, его трудно использовать для моделирования произвольных объектов. В то же время нейронные модели способны аппроксимировать сложный сигнал, такой как функция расстояния, и могут являться непрерывным и дифференцируемым представлением [2], что позволяет, использовать их для задач обратного рендеринга [3-4].

Развитие нейронных архитектур для представления SDF прошло путь от простейших моделей с латентными векторами [15-16] до более сложных структур, использующих

октодеревья [17] и хэш-таблицы [6]. В данной работе для сравнения выбран базовый подход: многослойный перцептрон с периодической функцией активации – SIREN [2].

В нашем сравнении для рендеринга SDF использовался стандартный итерационный алгоритм Sphere Tracing [18] с максимальным числом шагов равным 100. Нейронные SDF были обучены на модели Armadillo [19]. Для нашего исследования обучены варианты SIREN на сетке гиперпараметров: число скрытых слоёв сети $L_{SIREN} \in \{2,3,4,5\}$, ширина слоя $H_{SIREN} \in \{16,32,64,128\}$. На рис. 1 изображен рендеринг модели с каждой из рассмотренных конфигураций (первое число – число скрытых слоев, второе – ширина слоя).

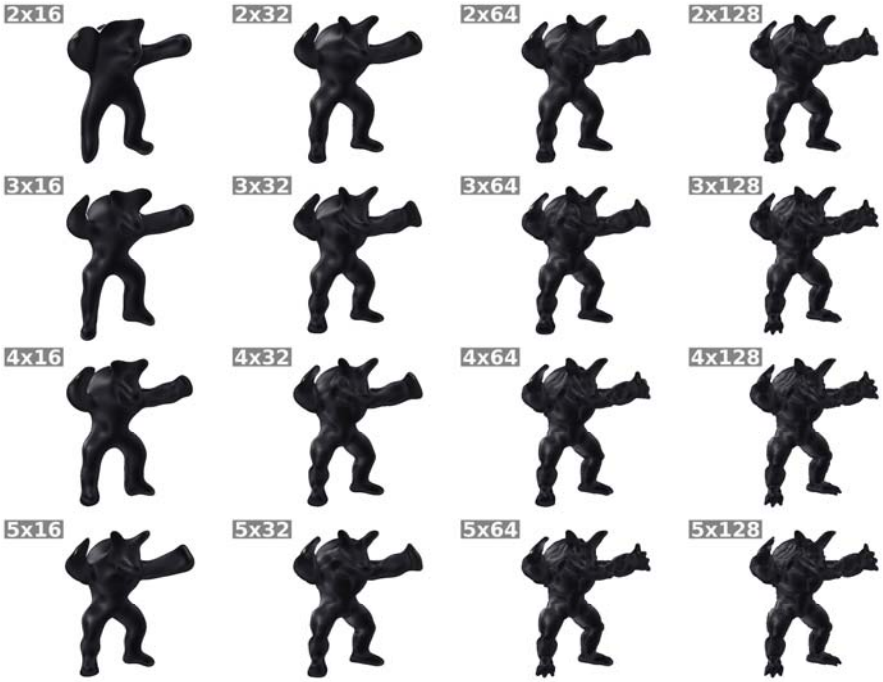

Рис. 1. Рендеринг нейронной SDF с варьируемым количеством скрытых слоёв и шириной сети.
Fig. 1. Neural SDF rendering with varying numbers of hidden layers and network widths.

На рис. 2 и в табл. 1 приведены результаты замеров времени выполнения нейронных SDF методами 3.1–3.3 при разном числе параметров модели. Эксперименты показали, что различия во времени выполнения для одной и той же модели могут достигать трех порядков. Так, метод с компилируемыми весами (названный также «веса в заголовке» на рис. 2) демонстрирует относительно высокую производительность при малом размере внутреннего слоя. Однако при повышении его с 32 до 64 наблюдается резкий рост времени выполнения, либо возникают ошибки создания конвейера в драйвере (обозначены как E в таблицах). Как видно из табл. 1 применение данного метода для большого числа слоев или размера слоя практически невозможно. Метод 3.2 показывает линейный рост времени выполнения при увеличении числа параметров. Для малых моделей он уступает в производительности, но сохраняет применимость при любом размере модели, хотя и с существенными расходами на доступ к памяти. Метод 3.3 (Cooperative Vectors) показывает лучшую производительность и стабильно работает при всех рассмотренных размерах моделей.

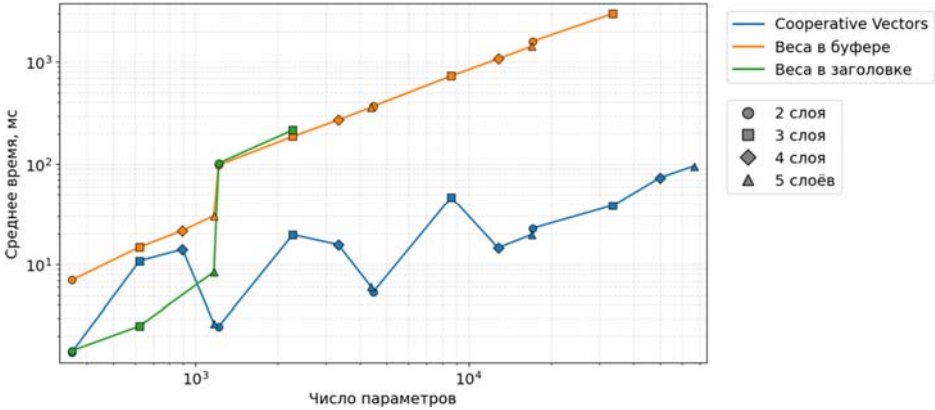

Рис. 2. Время выполнения нейронной SDF при разном числе параметров модели.
Fig. 2. Neural SDF execution time for different numbers of model parameters.

Табл. 1. Время выполнения нейронной SDF.
Table 1. Neural SDF execution time.

Метод реализации	**Параметры модели (число слоёв × ширина слоя)**							
	2×16	2×32	2×64	2×128	3×16	3×32	3×64	3×128
	4×16	4×32	4×64	4×128	5×16	5×32	5×64	5×128
	Среднее время, мс							
Cooperative Vectors	1.372	2.463	5.375	22.823	10.867	19.704	46.396	38.455
	14.102	15.695	14.606	73.173	2.636	6.087	19.630	95.288
Веса в буфере	7.014	97.412	366.468	1600.818	14.836	186.662	728.016	2998.196
	21.616	271.430	1072.837	E	30.232	360.798	1421.615	E
Компилируемые веса	1.425	101.678	E	E	2.492	215.939	E	E
	E	E	E	E	8.448	E	E	E

4.3 Моделирование материалов с помощью нейронных BRDF

Двунаправленная функция отражательной способности (Bidirectional Reflectance Distribution Function, BRDF) является ключевой характеристикой материала, определяющей локальное поведение отражённого света. Она зависит от направлений падающего и отражённого излучения, и, таким образом, представляет собой четырёхмерную функцию в пространстве углов. В отличие от аналитических моделей, способных представлять лишь простейшие материалы, нейронные BRDF (NBRDF) [20] позволяют аппроксимировать сложные и анизотропные материалы. Они обеспечивают более точное воспроизведение реальных материалов на основе измерений, являясь при этом достаточно компактным представлением по сравнению с табличным хранением измеренных значений. И хотя развитие нейронных BRDF также прошло большой путь, в основе большинства методов лежит использование нейронного декодера, непосредственно запоминающего материал [20] или воспроизводящего его из латентного представления [1].

Для экспериментов были обучены модели NBRDF на сетке гиперпараметров: число скрытых слоёв сети $L_{NBRDF} \in \{1,2,3,4\}$, ширина слоя $H_{NBRDF} \in \{16,32,64,128\}$. Для тестирования

выбрано 10 изотропных материалов из набора данных EPFL RGL [21], изображённых на рис. 3 (слева). Для замеров качества производился рендеринг с использованием карты окружения. Значения метрики SSIM для изображений, полученных с различными конфигурациями модели, показаны на рис. 3 (справа).

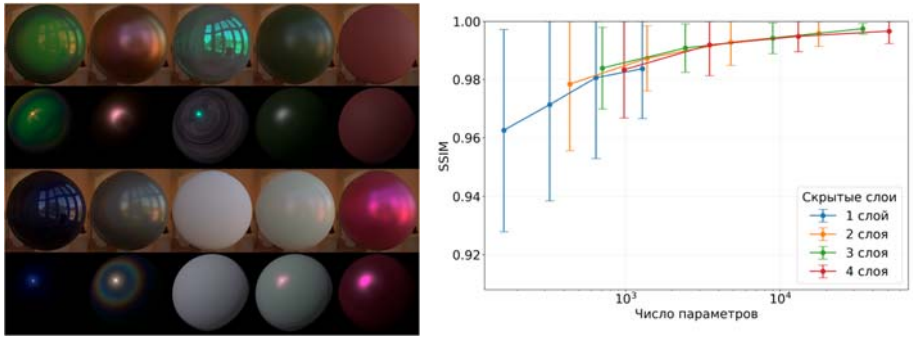

Рис. 3. Используемые в сравнении материалы из набора данных EPFL RGL [21] (слева), средние значения и стандартное отклонение SSIM для различных конфигураций NBRDF (справа).
Fig. 3. Materials from the EPFL RGL [21] dataset used in the comparison (left), standard deviation of SSIM for different NBRDF configuration (right).

На рис. 4 и в табл. 2 приведены результаты замеров времени выполнения нейронных BRDF методами 3.1–3.3 при разном числе параметров модели. Эксперименты показали, что различия во времени выполнения для одной и той же модели могут достигать четырех порядков. Так же, как и в экспериментах с моделями поверхностей, метод с компилируемыми весами демонстрирует относительно высокую производительность при малом размере внутреннего слоя. Но при его повышении наблюдается резкая потеря производительности, либо его применение становится невозможным. Метод 3.2 показывает почти линейный рост времени выполнения при увеличении числа параметров. Для малых моделей он уступает в производительности, но сохраняет применимость при любом размере модели, хотя и с существенными расходами на доступ к памяти. Аналогично эксперименту с нейронноми SDF метод 3.3 (Cooperative Vectors) показывает лучшую производительность и стабильно работает при всех рассмотренных размерах моделей.

5. Выводы и заключение

В рамках исследования был проведен сравнительный анализ различных методов выполнения нейронных моделей в задачах рендеринга реального времени. Эксперименты включали реализацию и измерение производительности различных подходов к выполнению нейронных SDF и BRDF. Полученные данные позволяют сделать обоснованный вывод о том, что эффективное выполнение нейронных моделей в задачах графики реального времени возможно. Однако в каждом конкретном случае необходим тщательный выбор стратегии реализации с учетом специфики задачи и целевой платформы.

Нейросетевое представление поверхностей, как правило, требует применения сложных архитектур. Рассмотренная архитектура SIREN способна качественно реконструировать мелкие детали только при использовании относительно большой сети. В свою очередь, алгоритм Sphere Tracing предполагает многократные вычисления функции расстояния, что влечёт за собой высокие вычислительные затраты. В связи с этим в рассматриваемом сценарии наиболее обоснованным является выбор метода 3.3.

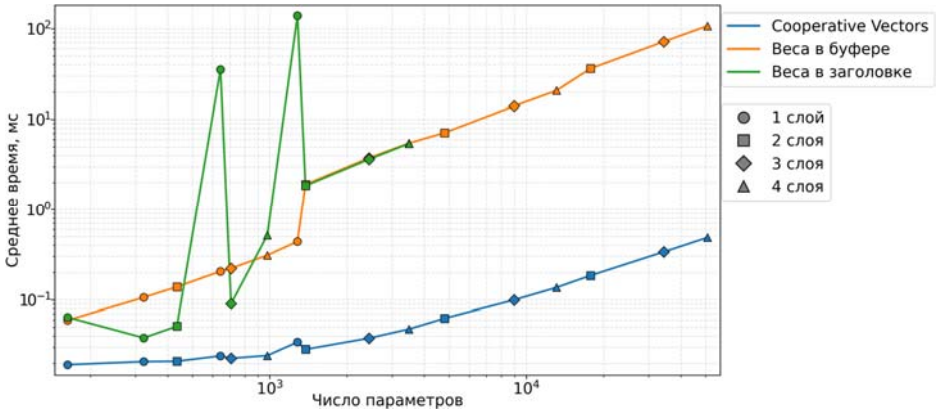

Рис. 4. Время выполнения нейронной BRDF при разном числе параметров модели.
Fig. 4. Neural BRDF execution time for different numbers of model parameters.

Табл. 2. Время выполнения нейронной BRDF.
Table 2. Neural BRDF execution time.

Метод реализации	**Параметры модели (число слоёв × ширина слоя)**							
	1×16	1×32	1×64	1×128	2×16	2×32	2×64	2×128
	3×16	3×32	3×64	3×128	4×16	4×32	4×64	4×128
	Среднее время, мс							
Cooperative Vectors	0.019	0.020	0.024	0.034	0.021	0.028	0.062	0.184
	0.022	0.037	0.099	0.336	0.024	0.047	0.136	0.485
Веса в буфере	0.059	0.106	0.205	0.438	0.138	1.910	7.121	36.166
	0.221	3.758	13.972	71.517	0.308	5.472	20.719	106.933
Компилируемые веса	0.064	0.038	35.360	138.829	0.051	1.850	E	E
	0.090	3.627	E	E	0.516	5.429	E	E
Cook-Torrance	0.027							

В противоположность представлениям поверхности, для представления материалов достаточно компактных моделей. Приемлемого для рендеринга в реальном времени качества можно достичь уже с двухслойной сетью небольшой ширины (в оригинальной работе по NBRDF предлагается использовать два внутренних слоя ширины 21). В таком случае метод 3.1 представляет собой оптимальный баланс между производительностью и кроссплатформенностью. В то же время метод 3.3 обеспечивает скорость рендеринга, сопоставимую с аналитическими моделями (например, с классической моделью Кука-Торренса), что позволяет заменять их нейросетевыми без потери производительности.

Таким образом, можно сделать следующие обобщения. Методом 3.1 можно выполнять только небольшие модели, (до примерно 2000 параметров на видеокарте NVIDIA RTX 3080), но он позволяет получить высокую скорость, и не ограничен конкретной платформой. Метод 3.2 может быть использован, если нужно переносимое решение, работающее с любым числом параметров, и задача не требует большого числа оценок модели. Метод 3.3 является предпочтительным, если целевым оборудованием являются современные графические ускорители, имеющие поддержку расширения Cooperative Vectors.

Список литературы / References

[1]. Zeltner T., Rousselle F., Weidlich A., Clarberg P., Novák J., Bitterli B., Evans A., Davidovič T., Kallweit S., Lefohn A. Real-Time Neural Appearance Models. ACM Transactions on Graphics, 2024, vol. 43, no. 3, Article 33, pp. 1–17. DOI: 10.1145/3659577.

[2]. Sitzmann V., Martel J., Bergman A., Lindell D., Wetzstein G. Implicit Neural Representations with Periodic Activation Functions. Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 7462–7473.

[3]. Zhang D., Wang J., Wang S., Mihajlovic M., Prokudin S., Lensch H.P., Tang S. RISE-SDF: A Relightable Information-Shared Signed Distance Field for Glossy Object Inverse Rendering. The 12th International Conference on 3D Vision (3DV), 2025, pp. 597–606. DOI: 10.1109/3DV66043.2025.00060.

[4]. Zhang K., Luan F., Li Z., Snavely N. IRON: Inverse Rendering by Optimizing Neural SDFs and Materials from Photometric Images. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 5555–5564. DOI: 10.1109/CVPR52688.2022.00548.

[5]. Милин Н.А., Родионов Р.О., Галактионов В.А., Фролов В.А. Сравнительный анализ выполнения нейронных моделей для графики. Материалы 35-й Международной конференции по компьютерной графике и машинному зрению ГрафиКон 2025, 2025, с. 23–29. DOI: 10.25686/978-5-8158-2474-4-2025-23-29.

[6]. Müller T., Evans A., Schied C., Keller A. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. ACM Transactions on Graphics, 2022, vol. 41, no. 4, Article 102, pp. 102:1–102:15. DOI: 10.1145/3528223.3530127.

[7]. Chaitanya C.R.A., Kaplanyan A.S., Schied C., Salvi M., Lefohn A., Nowrouzezahrai D., Aila T. Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder. ACM Transactions on Graphics, 2017, vol. 36, no. 4, Article 98, pp. 98:1–98:12. DOI: 10.1145/3072959.3073601.

[8]. Munkberg J., Hasselgren J. Neural denoising with layer embeddings. Computer Graphics Forum, 2020, vol. 39, no. 4, pp. 1–12. DOI: 10.1111/cgf.14049.

[9]. Kulhánek J., Derner E., Sattler T., Babuška R. ViewFormer: NeRF-free Neural Rendering from Few Images Using Transformers. Computer Vision – ECCV 2022, 2022, pp. 198–216. DOI: 10.1007/978-3-031-19784-0_12.

[10]. Zeng C., Dong Y., Peers P., Wu H., Tong X. RenderFormer: Transformer-based Neural Rendering of Triangle Meshes with Global Illumination. ACM SIGGRAPH Conference Papers, 2025, Article 48, pp. 1–11. DOI: 10.1145/3721238.3730595.

[11]. Mildenhall B., Srinivasan P. P., Tancik M., Barron J. T., Ramamoorthi R., Ng R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Computer Vision – ECCV 2020, 2020, pp. 405–421. DOI: 10.1007/978-3-030-58452-8_24.

[12]. Weier P., Rath A., Michel É., Georgiev I., Slusallek P., Boubekeur T. N-BVH: Neural ray queries with bounding volume hierarchies. ACM SIGGRAPH Conference Papers, 2024, Article 99, pp. 1–11. DOI: 10.1145/3641519.3657464.

[13]. Khronos Group. VK_NV_cooperative_vector. Vulkan Registry. Available at: https://registry.khronos.org/vulkan/specs/latest/man/html/VK_NV_cooperative_vector.html, accessed: 09.02.2026.

[14]. Bangaru S., Wu L., Li T.-M., Munkberg J., Bernstein G., Ragan-Kelley J., Durand F., Lefohn A., He Y. SLANG.D: Fast, Modular and Differentiable Shader Programming. ACM Transactions on Graphics, 2023, vol. 42, no. 6, Article 264, pp. 264:1–264:28. DOI: 10.1145/3618353.

[15]. Park J. J., Florence P., Straub J., Newcombe R., Lovegrove S. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 165–174. DOI: 10.1109/CVPR.2019.00025.

[16]. Chabra R., Lenssen J. E., Ilg E., Schmidt T., Straub J., Lovegrove S., Newcombe R. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. Computer Vision – ECCV 2020, 2020, pp. 608–625. DOI: 10.1007/978-3-030-58526-6_36.

[17]. Takikawa T., Litalien J., Yin K., Kreis K., Loop C., Nowrouzezahrai D., Jacobson A., McGuire M., Fidler S. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 11353–11362. DOI: 10.1109/CVPR46437.2021.01120.

[18]. Hart J.C. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. The Visual Computer, 1996, vol. 12, no. 10, pp. 527–545. DOI: 10.1007/s003710050084.

[19]. The Stanford 3D Scanning Repository. Available at: http://graphics.stanford.edu/data/3Dscanrep, accessed: 09.02.2026.

[20]. Sztrajman A., Rainer G., Ritschel T., Weyrich T. Neural BRDF Representation and Importance Sampling. Computer Graphics Forum, 2021, vol. 40, no. 6, pp. 332–346. DOI: 10.1111/cgf.14335.

[21]. Dupuy J., Jakob W. An adaptive parameterization for efficient material acquisition and rendering. ACM Transactions on Graphics, 2018, vol. 37, no. 6, Article 274, pp. 274:1–274:18. DOI: 10.1145/3272127.3275059.

Информация об авторах / Information about authors

Никита Андреевич МИЛИН – студент кафедры Интеллектуальных информационных технологий факультета ВМК МГУ, программист Института прикладной математики им. М.В. Келдыша РАН. Сфера научных интересов: реалистичная компьютерная графика, рендеринг в реальном времени, параллельные вычисления.

Nikita MILIN – student at the Department of Intelligent Information Technologies, Faculty of Computational Mathematics and Cybernetics, Moscow State University, programmer at the Keldysh Institute of Applied Mathematics RAS. Research interests: realistic computer graphics, real-time rendering, parallel computing.

Роман Олегович РОДИОНОВ – студент кафедры Интеллектуальных информационных технологий факультета ВМК МГУ, программист Института прикладной математики им. М.В. Келдыша РАН. Сфера научных интересов: реалистичная компьютерная графика, спектральный и нейронный рендеринг.

Roman RODIONOV – student at the Department of Intelligent Information Technologies, Faculty of Computational Mathematics and Cybernetics, Moscow State University, programmer at the Keldysh Institute of Applied Mathematics RAS. Research interests: realistic computer graphics, spectral and neural rendering.

Владимир Александрович ФРОЛОВ – кандидат физико-математических наук, старший научный сотрудник Института прикладной математики им. М.В. Келдыша РАН, научный сотрудник факультета ВМК МГУ. Сфера научных интересов: реалистичная компьютерная графика, моделирование освещённости, разработка программных систем оптического моделирования, параллельные и распределённые вычисления.

Vladimir FROLOV – Cand. Sci. (Phys.-Math.) in computer graphics, senior researcher at Keldysh Institute of Applied Mathematics RAS and researcher in computer graphics at Moscow State University. Research interests: realistic computer graphics, light transport simulation, elaboration of optical simulation software systems, GPU computing.