ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 66 за 2019 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Лемешев А.Ф., Жданов Д.Д., Волобой А.Г., Дерябин Н.Б.

Использование методов фильтрации изображений в реалистичном рендеринге реального времени

Рекомендуемая форма библиографической ссылки: Использование методов фильтрации изображений в реалистичном рендеринге реального времени / А.Ф.Лемешев [и др.] // Препринты ИПМ им. М.В.Келдыша. 2019. № 66. 26 с. doi:10.20948/prepr-2019-66
URL: http://library.keldysh.ru/preprint.asp?id=2019-66

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

А.Ф. Лемешев, Д.Д. Жданов, А.Г. Волобой, Н.Б. Дерябин

Использование методов фильтрации изображений в реалистичном рендеринге реального времени

Москва — 2019

Лемешев А.Ф., Жданов Д.Д., Волобой А.Г., Дерябин Н.Б.

Использование методов фильтрации изображений в реалистичном рендеринге реального времени

В работе проведен анализ современных алгоритмов фильтрации изображений, синтезированных методом стохастической трассировки лучей, выполняемых в реальном времени. Вычислительная сложность алгоритмов до недавнего времени не давала возможности реализовать эти алгоритмы для работы в интерактивных приложениях. Тем не менее, качество фильтрации изображения у появившихся алгоритмов фильтрации реального времени уже сравнимо с результатом работы неинтерактивных фильтров. В исследовании особенное внимание было уделено четырем основным алгоритмам фильтрации, для которых приводится краткое описание. Представлен анализ алгоритмов на основе метрик качества изображения. Приведенный анализ позволил выявить основные проблемы рассмотренных алгоритмов и предложить пути их решения. Также были рассмотрены наиболее удачные способы улучшения качества фильтрации изображений в режиме реального времени.

Ключевые слова: реконструкция изображения, рендеринг в реальном времени, трассировка лучей, метод Монте-Карло, фильтрация изображения, подавление шума.

Andrew Fedorovich Lemeshev, Dmitry Dmitrievich Zhdanov, Alexey Gennadievich Voloboy, Nikolay Borisovich Deryabin

Usage of image-filtering techniques in realistic real-time rendering

This article discusses modern real time filtering algorithms for images synthesized by stochastic ray tracing method. The computational complexity of algorithms, until recently, did not make it possible to implement these algorithms for working in interactive applications. However, existing real-time filtering algorithms already show comparable results with non-interactive filters. The text of the article focuses on four algorithms, for which a brief description is given. The paper presents an analysis of algorithms based on image quality metrics. The analysis revealed problems of the algorithms and methods for their solution, and highlighted successful ways to improve the quality of real time filtering.

Keywords: image reconstruction, real-time rendering, ray tracing, Monte-Carlo ray tracing, image filtering, denoising.

Работа частично профинансирована грантами РФФИ № 18-08-01484, 18-01-00569 и 18-31-20032.

1. Введение

Возможность моделирования физически корректного освещения сцены — одна из основных проблем рендеринга в реальном времени. Для решения этой проблемы обычно применяются методы стохастической трассировки лучей. Эти методы основываются на случайном выборе пикселя в каждом кадре. Однако для них характерна высокая вычислительная сложность алгоритмов стохастической трассировки лучей и алгоритмов фильтрации изображения на основе полученных данных. На текущий момент производительность трассировки лучей ограничена приблизительно 100 миллионами лучей в секунду, что для изображения с разрешением 1920x1080 и скоростью обновления экрана, равной 30 кадрам в секунду, позволяет выполнить трассировку одного-двух лучей на пиксель. Для динамических сцен производительность может быть еще ниже.

Это означает, что в обозримом будущем моделирование физически корректного распространения светового излучения будет ограничено скоростью трассировки лучей. Поэтому для синтеза реалистичного изображения необходимо использовать специальную технику, позволяющую снизить шум, вызванный процессом случайного выбора источников видимой яркости пикселя. Во-первых, это может быть техника специального распределения случайных лучей и методы оптимального интегрирования яркости первичного, вторичного и каустического освещения сцены. Поскольку эти методы работают с исходными данными трасс лучей в пространстве сцены, то они в состоянии обеспечить физически корректный расчет видимой яркости. Однако эффективность этих методов недостаточно высока для того, чтобы обеспечить корректный рендеринг в режиме реального времени. Второй подход, позволяющий снизить шумовую составляющую яркости изображения, заключается в его фильтрации. Техника, используемая для фильтрации синтезированного изображения, основывается на специфике процесса его синтеза, и эта техника кардинальным образом отличается от техники фильтрации обычных цифровых изображений, ограниченных диапазоном яркости [0, 255] (или несколько выше для изображений широкого динамического диапазона). Кроме того, фильтрация обычных цифровых изображений не требует сохранения физической корректности, поскольку исходное изображение находится вне пространства реальных яркостей сцены.

С другой стороны, фильтр синтезированных реалистичных изображений обязан сохранять их физическую корректность, например, сохранять среднюю локальную яркость для любой области изображения. Несмотря на то, что

работа в пространстве реальных яркостей вызывает дополнительные сложности для алгоритма фильтрации изображения, наличие информации о структуре сцены в соответствующих точках экрана, например, ориентации нормалей поверхностей или их альбедо, позволяет существенным образом улучшить качество работы данных алгоритмов. Далее будут более детально представлены различные подходы и алгоритмы, применяемые в процессе физически корректной фильтрации синтезированных изображений.

2. История вопроса

На протяжении последнего десятилетия велись интенсивные исследования в области разработки алгоритмических решений физически корректного рендеринга реального времени. Одним из основных трендов в рендеринге реального времени является кэширование предварительно вычисленых трасс световых лучей [1] на объектах сцены. Для того чтобы повысить качество изображения, построенного на основе кэшированного освещения, используются методы, добавляющие такие визуальные эффекты, как ambient occlusion, глянцевые отражения и мягкие тени [2]. Чаще всего данные решения строят красивые и визуально правдоподобные изображения, но их реальная яркость может быть далека от физически корректной. Для достижения реализма в интерактивных приложениях разработчики все чаще используют стохастическую модель освещения, основанную на методе Монте-Карло.

Стохастические методы трассировки лучей позволяют правильно обработать большинство оптически сложных эффектов [3], таких как глубина резкости, размытие при движении, каустика, мягкие тени и глобальное освещение. Помимо большей реалистичности, использование трассировки лучей позволяет легко добавить некоторые сложные оптические эффекты, например, преломление или дисперсию света, что в конечном счете позволяет упростить итоговый графический конвейер. Это видно в приложениях рендеринга, работающих не в реальном времени [4], что привело к появлению многочисленных методов фильтрации, предназначенных для удаления остаточного стохастического шума. Цвикер в работе [5] предоставил обзор данных методов.

Как правило, алгоритмы фильтрации уменьшают дисперсию путем объединения нескольких семплов Монте-Карло на пиксель. Они стараются сгладить выходное изображение при сохранении всех мелких особенностей изображения, например: края, детали поверхности и др. Большинство пространственных фильтров обрабатывают входные изображения, построенные

по нескольким десяткам случайных выборок на пиксель, причем время вычислений измеряется в секундах или минутах.

Подходы на основе регрессии [6, 7] дают хорошие результаты фильтрации на изображениях более высокого качества (> 128 случайных выборок на пиксель), однако в случае низкого качества изображения эти фильтры работают ненадежно. Кроме того, учитывая их высокую вычислительную сложность, они неприменимы для рендеринга в реальном времени. Мункберг [8] демонстрирует возможность использования простых фильтров, работающих в текстурном пространстве, которые выигрывают как от пространственного, так и от временного повторного использования. Однако использование этого подхода для графических приложений, работающих в реальном времени, является нетривиальной задачей. Фильтрация также может происходить в пространстве лучей [9], но при этом используются сложные kNN-алгоритмы, в которых соединяются алгоритмы рендеринга и фильтрации. Пространственные фильтры, работающие по принципу «черный ящик», удобны для приложений реального времени. Хотя поиск соответствий между трассами лучей может улучшить качество фильтрации изображений [10], но расчет этих соответствий – вычислительно сложная задача, и хорошего качества фильтрации можно добиться только после обработки большого количества кадров.

В работе Дьюранда [11] показано, как с помощью частотного анализа засвеченной области можно определить оптимальную область фильтрации. Более поздние расширения алгоритма касаются мягких теней, размытия движения, расфокусировки и непрямого освещения с использованием линейных сдвиговых фильтров [12] и фильтров, выровненных по оси [13]. Несмотря на то, что частотный анализ увеличивает общую вычислительную сложность и, как правило, требует около 4-16 семплов на пиксель для получения хороших результатов, использование разделяемых сдвиговых фильтров приводит к повышению производительности [14]. Также необходимо отметить, что на сложных участках частотно-пространственные фильтры используют меньшую область фильтрации, что приводит к большему разбросу результатов.

Для сохранения мелких деталей нелинейные фильтры адаптируют веса пикселей, что позволяет избежать уменьшения размера обрабатываемой области. Первые исследования по нелинейной реконструкции используют удаление «лишних» пикселей [15], плавное перераспределение энергии [16] и анизотропное рассеивание [17]. Билатеральный фильтр с функцией определения границ объектов также может применяться к реконструкции

изображений [18]. Билатеральный фильтр, представленный в [19], заменяет каждый пиксель средневзвешенным значением соседних пикселей с использованием распределения по Гауссу, которое учитывает расстояние, цвет и другие различия при формировании данных для функции определения границ объектов. Учет геометрической информации об объектах в таких функциях повышает надежность билатерального фильтра.

Определение веса компонент функции определения границ объектов в каждом пикселе дополнительно повышает устойчивость к пространственно изменяющемуся шуму на изображении. Ли [20], Роззелл [21], Калантари [22] и др. используют массив фильтров, среди которых в зависимости от входной дисперсии или ошибки фильтра выбирается подходящий для данного пикселя фильтр. В другой своей работе Калантари [23] предлагает применить небольшую нейронную сеть для расчета весов данных, используемых в билатеральном фильтре. Такие адаптивные фильтры могут применяться даже при относительно низком количестве семплов и обеспечивать требуемое качество фильтрации изображения, но эти фильтры основаны на значительной предварительной обработке или сглаживании аномальных значений дисперсии и в текущем виде не работают в реальном времени.

Существует несколько быстрых приближений для кросс-билатеральных фильтров, таких как: управляемые фильтры [24], вейвлеты с функцией определения границ объекта [25] и другие. Обычно эти приближения создают дополнительные артефакты (например эффект ореола вокруг объекта). Часто бывает, что при низкой дискретизации входных данных (то есть меньше чем 4 семпла на пиксель) невозможно определить мелкую структуру изображения, что в итоге приводит к чрезмерному размытию или к потере деталей изображения при фильтрации.

Использование временной информации между кадрами помогает устранить недостатки пространственных фильтров и улучшить временную стабильность при низкой дискретизации. Делбракио [26] обрабатывает данные лучей из трех кадров, чтобы уменьшить мерцание, а Мейер и Андерсон [27] используют метод главных компонент PCA (principal component analysis) по всем кадрам, отбрасывая незначительные значения для улучшения временной стабильности. Циммер [28] использует декомпозицию в пространстве лучей для оценки движения и фильтрует несколько буферов. Но эти методы требуют множества предварительно вычисленных кадров, а не темпорально отфильтрованные изображения предыдущих кадров, как это требуется для приложений в реальном времени.

Интерактивные фильтры часто репроецируют пиксель из одного кадра в другой на основе векторов движения [29]. Это похоже на интерполяцию видимости [30], которая репроецирует семплы из предварительно представленного множества изображений для генерации новых позиций наблюдения. Сейчас репроекция приобрела еще большую популярность благодаря широкому внедрению временного сглаживания (temporal anti-aliasing) TAA [31]. TAA берет за основу временной амортизированный суперсемплинг [32], но вместо того, чтобы игнорировать «неудачные» семплы, TAA использует соседние цвета пикселей, чтобы рассчитать цвет в текущем. Патни [33] улучшил данный подход с помощью оценки статистического распределения цветов.

Ниже представлены 4 наиболее удачных современных алгоритма фильтрации изображений, синтезированных в реальном времени методом трассировкой лучей.

3. Алгоритм Edge-avoiding à-trous wavelets (EAW)

Данный метод [25] создает на основе зашумленного изображения высокого разрешения, полученного трассировкой лучей с использованием метода Монте-Карло, изображение без видимого шума, используя кроме исходного изображения буферы нормалей и положений объектов сцены в экранном пространстве. Дополнительные буферы данных вместе с исходным изображением позволяют сохранить мелкие детали изображения, например резкие тени, информации о которых нет во входных данных. На изображении остаются острые края вокруг геометрических границ, и сглаживается освещение там, где это необходимо. Использование общего алгоритма трассировки лучей позволяет корректно обрабатывать оптически сложные свойства объектов сцены, такие как глянцевые материалы и источники света сложной формы с диаграммой излучения (например, точечные источники света с асимметричной диаграммой излучения, протяженные источники излучения, освещение, основанное на HDR (high dynamic range) изображении), без специальной обработки.

Рисунок 1 иллюстрирует основные этапы данного алгоритма. На вход подается зашумленное изображение (буфер rt), в простейшем случае в нем содержится как прямое, так и вторичное освещение. Кроме того, функция определения границ объекта À-Trous фильтра учитывает нормали и позиции пикселей. Количество применений фильтра À-Trous определяется желаемым размером фильтра, например, 5 итераций с размером ядра 80 × 80 пикселей.

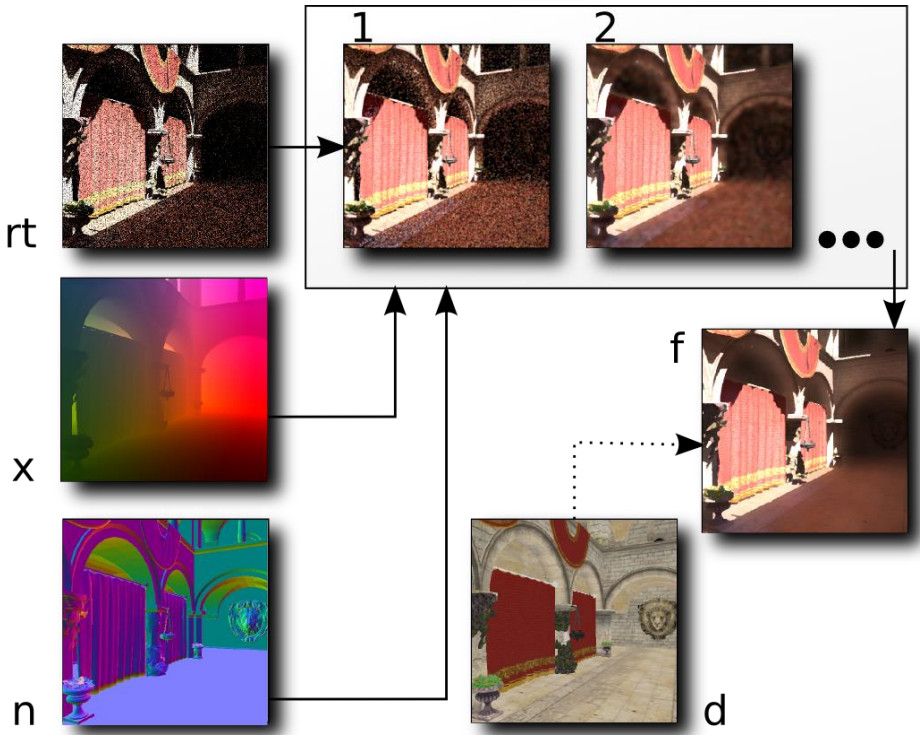

Рис. 1. Этапы алгоритма EAW.

Гладкая поверхность формирует буфер нормалей с плавно меняющимся цветом, и при фильтрации применяется равномерное сглаживание с увеличением размера ядра фильтра за одну итерацию. При наличии ребер влияние соседних точек уменьшается функцией определения границ объекта. Подобный подход тесно связан с билатеральным фильтром и использовался с алгоритмом À-Trous в [34].

Кроме видимых преимуществ, данный алгоритм создает ряд проблем с качеством фильтрации изображения сложных сцен. EAW плохо справляется со слишком сложными сценами, поскольку одной случайной выборки на трассу луча для формирования буфера rt, буфера позиции и буфера нормали пикселя недостаточно для сбора всей необходимой информации, обеспечивающей качественную фильтрацию. Это приводит к потере мелких деталей и чрезмерному размытию, например, теней или границ объектов. Данный недостаток заметен, например, на изображении листвы, представленном на рисунке 10 приложения А. Следующий недостаток состоит в том, что из-за функции определения границ объекта вейвлет-преобразование À-Trous больше не сохраняет энергию. Хотя это приемлемо в некоторых ситуациях, например в игровых приложениях, этот факт следует учитывать для рендеринга высокого разрешения, работающего в неинтерактивных приложениях.

Результат работы алгоритма представлен на рисунке 10 приложения А.

4. Алгоритм Spatiotemporal Variance-Guided Filtering (SVGF)

Данный фильтр [35] реконструкции изображения принимает в качестве входных данных буфер цвета, полученный трассировкой 1 луча на пиксель, вместе с растеризованным G-буфером изображения [36] и буфером истории, сформированным из обработанных данных изображения предыдущего кадра. На выходе фильтр выдает восстановленное изображение и заполняет буферы для последующих кадров. Основные этапы алгоритма изображены на рис. 2.

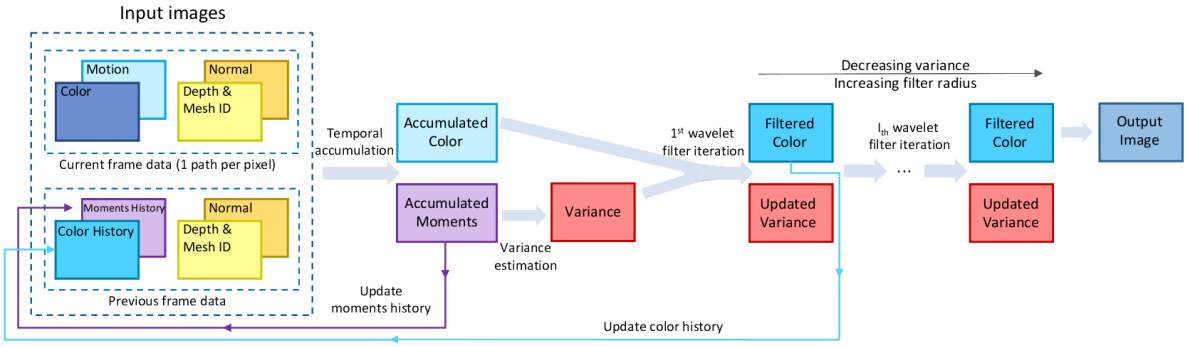

Рис. 2. Этапы алгоритма SVGF.

G-буфер, который подается на вход, содержит глубину до объектов сцены, нормали объекта, ID меша (мелких частей моделей сцены, которые можно рассматривать обособлено) и векторы движения в пространстве экрана, созданные в процессе построения изображения методом растеризации. От предыдущего кадра фильтр получает данные цветов, глубин, нормалей и ID меша. Для того чтобы увеличить надежность, в алгоритме не используется информация, специфичная для сцены, такая как позиция источника света, форма источника или другие параметры сцены. Кроме того, алгоритм не использует информацию о методе трассировки лучей, с помощью которого было построено изображение.

В связи с тем, что временное сглаживание (TAA) [31] в настоящее время широко применяется для создания суперсемплинга в видеоиграх, ту же технику можно применить и для реализации рендеринга, работающего в реальном времени. К сожалению, фильтрация на основе данных о цвете пикселей, полученных от предыдущего кадра, вносит артефакты, когда применяется к зашумленному изображению, полученному методом трассировки лучей. Минимизировать эти артефакты и увеличить эффективное количество семплов можно, применив темпоральный фильтр, основанный на данных о геометрии сцены и базирующийся на идеях Диего Нехаба [29] и Леи Янга [32].

Как и в TAA, для работы фильтра требуется 2D вектор движения, связанный с каждым семплом цвета пикселя C_i для кадра i. Такой вектор описывает геометрическое движение семпла от предыдущего кадра и позволяет нам спроецировать C_i на его место. Получая доступ к буферу цвета пикселей предыдущего кадра, продуцируемому предыдущим проходом фильтрации, постепенно накапливаются семплы цвета нескольких кадров. Для каждого C_i выполняется обратная проекция, чтобы найти семпл C_{i-1} из буфера предыдущего кадра, сравнить глубины двух семплов, нормали и ID меша и определить, являются ли семплы согласованными (то есть находятся на одной поверхности). Эти тесты согласованности используют эмпирические метрики сходства, аналогичные эвристике модуляции фрагментов, предложенных в работе [37, 38]. Эти семплы накапливаются в виде нового интегрированного семпла C'_i с помощью формулы:

$$C'_i = \alpha \cdot C_i + (1 - \alpha) \cdot C'_{i-1}, \qquad (1)$$

где α – коэффициент, который контролирует показатели стабильности по времени и отставанию цвета. Авторы SVGF алгоритма нашли наиболее подходящее значение коэффициента α = 0.2. В буфере векторов движения хранятся перемещения камеры и объектов сцены (как вектора движения в экранных координатах), хотя возможны и более сложные преобразования объектов сцены, требующие хранения большего объема данных в буферах.

Чтобы улучшить качество изображения движущихся объектов, повторно применяется билинейный фильтр 2×2 к каждому семплу C_{i-1}. Каждый шаг фильтра отдельно проверяет глубину, нормали и ID меша. Если на шаге присутствует несогласованная геометрия, семпл отбрасывается, а его вес равномерно перераспределяется по остальным шагам. Если на всех шагах присутствует несогласованная геометрия, в этом случае увеличивается область фильтрации для того, чтобы найти «мелкую» геометрию, например, листву. Если при увеличении области не удается найти согласованную геометрию, то в этом случае семпл представляет собой отклонение, историческое значение пикселя полностью отбрасывается и используется цвет только с текущего кадра.

В качестве основного метода обработки пикселей используется «вейвлет-фильтр», модифицированный EAW [25], который работает со входными данными итерационно, на каждой итерации увеличивая площадь обработки. Чтобы локально настроить EAW на сигнал, оценивается дисперсия яркости пикселя. Для вычисления степени зашумленности области изображения используется темпоральная аккумуляция. Ключевая идея заключается в том,

что реконструкция изображения не должна происходить в областях с небольшим шумом или вообще без шума (например, в областях с полным затенением), основное внимание должно быть уделено наиболее шумным участкам изображения. Необходимо отметить, что пространственная дисперсия — это не самая точная характеристика для оценки шума изображения: шум увеличивает дисперсию, но он не является единственным источником пространственной дисперсии.

Движение камеры и анимация приводят к артефактам, которые влияют на качество оценки дисперсии. Там, где данные накопления сильно ограничены (менее четырех кадров), оценка дисперсии производится с использованием билатерального фильтра 7×7 с весами, вычисленными по значениям глубин и нормалей.

Результат работы алгоритма представлен на рисунке 11 приложения А.

5. Алгоритм SURE-based Optimization for Adaptive Sampling and Reconstruction (SBF)

Данный метод [20] начинается с рендеринга небольшого количества исходных семплов для каждого пикселя, а затем выполняются итерации, на которых применяются различные фильтры с этапами адаптивного семплирования до момента достижения максимального качества изображения. После каждого этапа семплирования восстанавливаются полученные изображения с использованием нескольких фильтров. Каждый пиксель фильтруется несколько раз всеми фильтрами, и ошибка цвета каждого обработанного пикселя оценивается путем вычисления статистической объективной оценки случайных величин Штейна (Stein's unbiased risk estimate, SURE) [39]. Для каждого пикселя выбирается обработанный цвет с наименьшей ошибкой SURE, который попадает в восстановленное изображение. Дополнительно может быть вызван этап адаптивного семплирования, и новые семплы распределяются по пикселям с большими ошибками SURE. На рисунке 3 изображена блок-схема метода.

Прежде всего, небольшое количество начальных семплов (обычно 8 или 16 семплов на пиксель) берется, чтобы исследовать сцену. Семплы генерируются последовательностями с малой разностью и распределяются равномерно по каждому пикселю. После рендеринга с этими семплами данный метод выполняет первый этап реконструкции с собранной информацией.

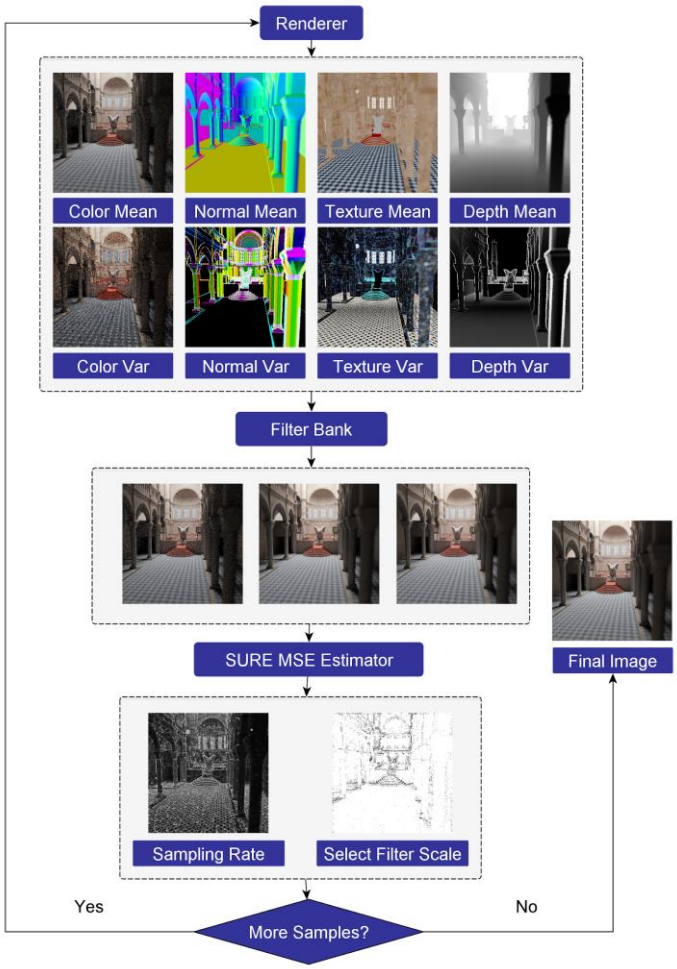

Рис. 3. Этапы алгоритма SBF.

Основное преимущество этой техники перед техникой жадной минимизации ошибок [40] — это возможность использовать произвольные фильтры. В данной работе в качестве примера были использованы кросс-билатеральные фильтры с различными коэффициентами, поскольку они позволяют добиться наилучшего компромисса между производительностью и качеством изображения.

В других исследованиях показано, что кросс-билатеральные фильтры эффективны для удаления шума после метода Монте-Карло [25, 41]. Как и в предыдущих исследованиях, вспомогательные данные, включая нормали поверхности, глубины и цвета текстуры, собираются путем кэширования информации после трассировки лучей. В процессе фильтрации для каждого пикселя вычисляются среднее значение и дисперсия и сохраняются как массив значений для кросс-билатерального фильтра.

Сен и Дараби [41] отмечают, что при рендеринге таких эффектов, как глубина резкости линзовой системы и размытие движущихся геометрических объектов, их свойства (нормаль и глубина до наблюдателя) могут быть

неверными из-за шума, вызванного методом Монте-Карло. В этих ситуациях, поскольку весовая функция не является точной, использование геометрических свойств в том виде, в котором они были синтезированы, может увеличить шум. Исследование, проведенное Сен и Дараби, решает эту проблему путем вычисления функциональной зависимости случайных параметров Монте-Карло и особенностей сцены. Они используют полученные параметры для уменьшения веса семплов, если свойства этих семплов имеют сильную зависимость от случайных параметров. Этот метод успешно обрабатывает изображения с дефокусировкой и с размытием движущихся объектов. Однако то, что метод применяется для каждого пикселя, может привести к снижению производительности и к значительному потреблению памяти, поскольку вычисление попарной взаимной информации между семплами и случайными параметрами Монте-Карло не только занимает много времени, но также требует значительного количества памяти для хранения результатов.

В итоге, для каждого пикселя используется найденная минимальная ошибка SURE для того, чтобы определить оптимальный масштаб кросс-билатеральных фильтров. Однако вычисление SURE с использованием семплов, полученных методом Монте-Карло, обычно приводит к неверному выбору фильтра и, таким образом, дает зашумленный результат. Причина заключается в том, что SURE является оценочной функцией средней квадратичной ошибки (mean squared error, MSE) и имеет свою собственную дисперсию. Чтобы уменьшить дисперсию SURE, необходимо либо увеличить количество семплов, либо выполнить дополнительную фильтрацию. В данной реализации решено было выполнить фильтрацию, чтобы уменьшить дисперсию SURE.

Поскольку человеческий глаз более чувствителен к ошибкам в темных областях изображения, необходимо обрабатывать большее количество семплов в темных областях и дополнительно делить функцию семплирования на квадрат яркости отфильтрованного цвета. Эта стратегия также была использована в [42].

Результат работы алгоритма представлен на рисунке 12 приложения А.

6. Алгоритм LBF – Machine Learning Approach for Filtering Monte Carlo Noise

В этом методе [23] авторы используют нейронную сеть, формально многослойный перцептрон (multilayer perceptron, MLP), в качестве модели регрессии, поскольку она представляет собой простую, но мощную систему для

обнаружения сложных нелинейных отношений между ее входными и выходными данными. Более того, MLP по своей природе параллельна и может быть эффективно реализована на GPU. Однако предлагаемый подход отличается от стандартного MLP тем, что в процесс обучения добавлен фильтр, который требует «обратного распространения» (backpropagation) данных для обновления весов сети во время обучения. Это подразумевает то, что фильтр должен быть дифференцируемым по отношению к параметрам фильтра. Общие фильтры, такие как фильтры Гаусса, кросс-билатеральные и кросс-нелокальные средние, являются дифференцируемыми и могут применяться в данной системе. Этапы алгоритма показаны на рис. 4.

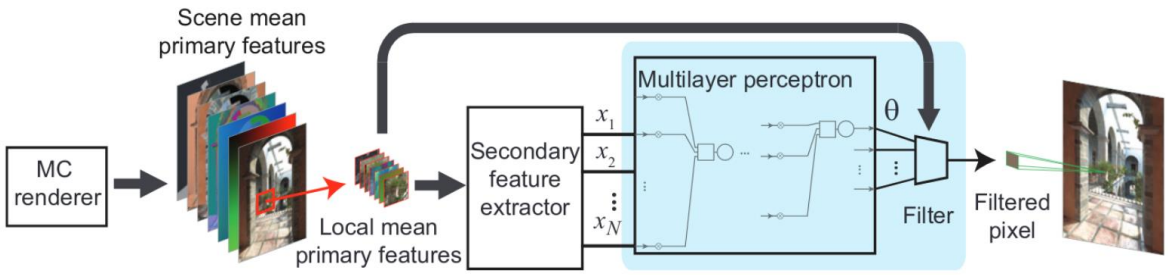

Рис. 4. Этапы алгоритма LBF.

Идея метрики для измерения погрешности между значениями отфильтрованного и референсного пикселя заключается в следующем: вместо стандартного MSE используется показатель относительной среднеквадратичной ошибки (RelMSE), предложенный Роуззеллом [40], который лучше подходит для данного приложения, поскольку зрительная система человека более чувствительна к изменениям цвета в более темных областях изображения.

Для оптимизации обучения нейронной сети авторы используют большое количество входных данных и соответствующих референсных изображений, которые должны быть сгенерированы до начала процесса обучения. Для каждого зашумленного изображения извлекается множество дополнительных данных, связанных с каждым пикселем. Далее необходимо обучить сеть с помощью итеративного трехэтапного процесса, называемого обратным распространением (backpropagation) [43]. Цель этого процесса состоит в том, чтобы найти оптимальные весовые коэффициенты для всех узлов сети, которые минимизируют ошибку между вычисленными и желаемыми выходными данными (то есть значениями референсных изображений) для всех пикселей в рассматриваемых изображениях.

Перед началом процесса обратного распространения веса случайным образом инициализируются небольшими значениями около нуля (от -0.5 до 0.5

в реализации авторов). Затем на первом этапе, называемом проходом с прямой связью, выходные данные сети вычисляются с использованием всех входных данных. На втором этапе ошибки между вычисленными и желаемыми выходными данными используются для определения влияния весов на выходную ошибку. Для этого необходимо взять производную изменения ошибки по весам. Таким образом, функции активации (activation functions) (в данном случае также и фильтр) должны быть дифференцируемыми. Эти два шага выполняются для всех обучающих данных, и накапливается ошибка каждого веса. Наконец, на третьем этапе все веса обновляются в соответствии с их накопленной ошибкой. Это завершает одну итерацию обучения, известную как эпоха, обычно для правильной тренировки системы и получения конвергентного набора весов требуется много эпох.

Результат работы алгоритма представлен на рисунке 13 приложения А.

7. Анализ

В данной статье сравниваются 4 современных алгоритма, которые в реальном времени и на входных данных низкого качества дают сравнимые результаты. Два из них (алгоритмы SBF и LBF) изначально не являлись алгоритмами реального времени, но на современном графическом оборудовании они обрабатывают изображение в пределах требований, предъявляемых к интерактивному приложению. Выбор алгоритмов, представленных в статье, продиктован прежде всего тем, что работ в области фильтрации изображений, синтезированных трассировкой лучей, выполняемой в реальном времени, практически нет. Среди существующих алгоритмов фильтрации в реальном времени в этой статье были рассмотрены два алгоритма: SVGF и EAW. Высокий интерес к данной тематике вызван прежде всего технологией RTX [44], предложенной корпорацией NVIDIA. На конференции GDC 2018 NVIDIA продемонстрировала свой алгоритм фильтрации в реальном времени, но, к сожалению, материалы по этому алгоритму на момент написания данной работы так и не были опубликованы.

Все алгоритмы, описанные и проанализированные в данной работе, получали на вход одинаковые входные параметры, состоящие из данных, синтезированных трассировкой лучей и G-буфера стадии растеризации, в котором находились: цвет, нормали, позиции объектов сцены в пикселе (вычисляемые через их глубину) и ID растеризованных объектов. Все рассмотренные алгоритмы никак не опирались на метод трассировки лучей и рассматривали его как черный ящик.

В алгоритме SVGF реконструкция изображения основана на темпоральной фильтрации, таким образом, фильтрация происходит с большим количеством эффективных семплов, хранящихся в буфере истории. Чаще всего темпоральная фильтрация позволяет SVGF получить лучший результат по сравнению с другими алгоритмами. Кроме того, применение алгоритма SVGF обеспечивает меньшую изменчивость от кадра к кадру (то есть обеспечивает более высокую темпоральную стабильность). Темпоральное накопление увеличивает эффективное количество семплов фильтрации, таким образом позволяя частично обойти ограничение в 1 семпл на пиксель. За счет этой техники алгоритму SVGF удается добиться лучшего качества изображения по сравнению с остальными алгоритмами.

Рис. 5. Эффект призрачности при использовании темпоральной аккумуляции.

С другой стороны, темпоральная аккумуляция влечет за собой ряд проблем. Одна из таких проблем — это эффект призрачности (см. рисунок 5), след от объекта из прошлого кадра. В общем случае достаточно использовать метод темпоральной репроекции [29], но в некоторых случаях (например, обработка зеркальных отражений) этого недостаточно, и призрачность может приводить к неприятным артефактам. Кристоф Шид [45] предложил способ, как в большинстве случаев можно избавиться от эффекта призрачности.

Другая серьезная проблема при использовании темпоральной аккумуляции заключается в отставании теней от объектов при движении, пример такого отставания представлен на рисунке 6. Проблема отставания теней менее заметна, но гораздо серьезнее. В ее основе лежит тот факт, что алгоритм использует цвет предыдущего кадра для модуляции цвета в текущем. В зависимости от коэффициента α из уравнения (1) можно добиться либо

большей темпоральной стабильности, но и большего отставания тени, либо, наоборот, – меньшего отставания, но и меньшей стабильности изображения на текущем кадре. Частично это также исправляется методом, предложенным Шидом [45].

Рис. 6. Отставание теней от объекта при использовании темпоральной аккумуляции.

Алгоритм SBF проводит серьезную обработку данных перед фильтрацией. С помощью этого алгоритма вычисляется оптимальный размер области обработки для кросс-билатерального фильтра. Более детально фильтруются сложные обрасти изображения. Нечто подобное было реализовано и в алгоритме SVGF, но там была использована яркость пикселя в качестве метрики. Так или иначе с помощью некоторых косвенных метрик или вычисления специального коэффициента соответствующий участок изображения может быть оценен, и на основе этой оценки может быть принято решение о глубокой обработке пикселей в данном участке.

Алгоритм LBF использует искусственную нейросеть для синтеза изображения. Авторам данного метода удалось добиться хорошего результата с использованием нейросетей для задач фильтрации. В условиях интерактивного приложения LBF показывает средний результат среди рассмотренных алгоритмов. Изображения, синтезированные рассмотренными алгоритмами, можно увидеть на рисунке 7. Визуально можно сравнить качество изображения в выделенных областях. Сравнение по метрикам RMSE и индексам

структурного сходства (Structural Similarity Index, SSIM), предложенное в [46], можно увидеть в таблице 1.

Таблица 1.

RMSE и SSIM метрики, полученные при сравнении с референсным изображением

Filter	Sponza (RMSE)	Sponza (SSIM)	SanMiguel (RMSE)	SanMiguel (SSIM)
EAW	0.040	0.909	0.070	0.818
SVGF	0.022	0.947	0.059	0.863
LBF	0.022	0.924	0.058	0.833
SBF	0.031	0.858	0.061	0.815

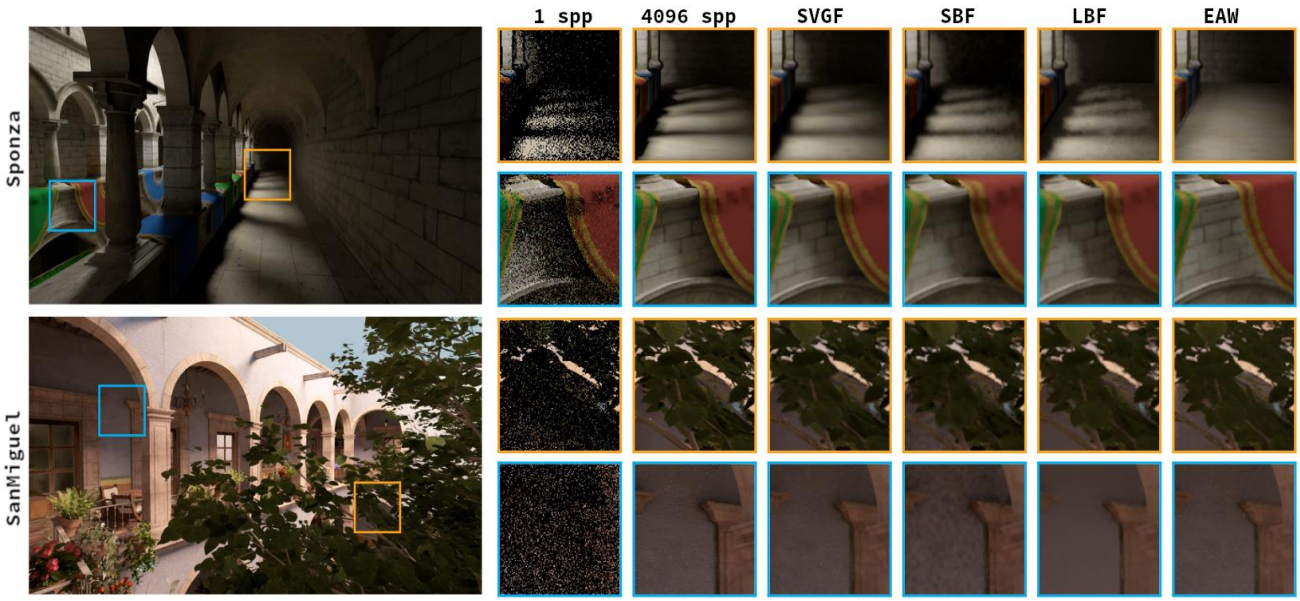

Рис. 7. Сравнение качества синтезируемых алгоритмами изображений.

8. Заключение

Были проанализированы четыре современных алгоритма фильтрации изображений, синтезированных трассировкой лучей, выполняемые в реальном времени. Вычислительная сложность алгоритмов фильтрации изображений до недавнего времени не давала возможность реализовать эти алгоритмы для работы в режиме реального времени. Тем не менее фильтрация изображений существующими алгоритмами реального времени уже достигает качества, сравнимого с работой неинтерактивных фильтров. Существующие проблемы фильтров, работающих в реальном времени, в основном заключаются в фильтрации динамических сцен, что является новой задачей для рендеринга в реальном времени. Ряд проблем с трассировкой лучей и процессом последующей фильтрации связан с производительностью всего конвейера рендеринга. Для увеличения производительности трассировки NVIDIA анонсировала новую технологию RTX. Дальнейшие исследования в данной теме будут направлены на увеличение производительности фильтров в целом и на увеличение временной стабильности между кадрами.

Литература

[1] Kajiya J. T. The Rendering Equation. // Computer Graphics (SIGGRAPH 86 Proceedings). 1986. Vol. 20. 143–150.

[2] Ritschel T., Dachsbacher C., Grosch T., Kautz J. The State of the Art in Interactive Global Illumination. // Computer Graphics Forum 31, 1 (2012), 160–188.

[3] Cook R. L., Porter T., Carpenter L. Distributed Ray Tracing. // Computer Graphics, Vol. 18. 1984. 137–145.

[4] Keller A., Fascione L., Fajardo M., Georgiev I., Christensen P., Hanika J., Eisenacher C., Nichols G. The Path Tracing Revolution in the Movie Industry. // SIGGRAPH Courses. 2015.

[5] Zwicker M., Jarosz W., Lehtinen J., Moon B., Ramamoorthi R., Rousselle F., Sen P., Soler C., Yoon S-E. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. // Computer Graphics Forum 34, 2 (2015), 667–681.

[6] Bierli B., Rousselle F., Moon B., Iglesias-Guitian J., Adler D., Mitchell K., Jarosz W., and Novak J.. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. // Computer Graphics Forum. 2016. Vol. 35. 107–117.

[7] Moon B., McDonagh S., Mitchell K., and Gross M. Adaptive Polynomial Rendering. // ACM Transactions on Graphics 35. 2016. 4, 40:1–40:10.

[8] Munkberg J., Hasselgren J., Clarberg P., Andersson M., Akenine-Moller T. // Texture Space Caching and Reconstruction for Ray Tracing. ACM Transactions on Graphics 35. 2016. 6, 249:1–249:13

[9] Keller A., Dahm K., Binder N. Path Space Filtering. // In Monte Carlo and Quasi-Monte Carlo Methods 2014. 2016. 423–436.

[10] Gautron P., Droske M., Wachter C., Kener L., Keller A., Binder N., Dahm K. Path Space Similarity Determined by Fourier Histogram Descriptors. // In ACM SIGGRAPH Talks. 2014. 39:1–39:1

[11] Durand F., Holzschuch N., Soler C., Chan E., Sillion F. A Frequency Analysis of Light Transport. // ACM Transactions on Graphics 24. 2005. 3, 1115–1126.

[12] Egan K., Durand F., Ramamoorthi R. Practical Filtering for Ecient Ray-traced Directional Occlusion. // ACM Transactions on Graphics 30. 2011. 6, 180:1–180:10.

[13] Mehta S., Yao J., Ramamoorthi R., Durand F. Factored Axis-Aligned Filtering for Rendering Multiple Distribution Effects. // ACM Transactions on Graphics 33. 2014. 4, 57:1–57:12.

[14] Hasselgren J., Munkberg J., Vaidyanathan K. Practical Layered Reconstruction for Defocus and Motion Blur. // Journal of Computer Graphics Techniques 4. 2015. 2, 45–58.

[15] Lee M. E., Redner R. A. A Note on the Use of Nonlinear Filtering in Computer Graphics. // IEEE Computer Graphics and Applications 10. 1990. 3, 23–29.

[16] Rushmeier H., Ward G. Energy Preserving Non-linear Filters. // InProceedings of SIGGRAPH 94. 1994. 131–138.

[17] McCool M. Anisotropic Diusion for Monte Carlo Noise Reduction. // ACM Transactions on Graphics 18. 1999. 2, 171–194.

[18] Xu R., Paanaik S. A Novel Monte Carlo Noise Reduction Operator. // IEEE Computer Graphics and Applications 25. 2005. 2, 31–35.

[19] Eisemann E., Durand D. Flash Photography Enhancement via Intrinsic Relighting. // ACM Transactions on Graphics 23. 2004. 3, 673–678.

[20] Li T., Wu Y., Chuang Y. SURE-based Optimization for Adaptive Sampling and Reconstruction. // ACM Transactions on Graphics 31. 2012. 6,194:1–194:9.

[21] Rousselle F., Manzi M., Zwicker M. Robust Denoising UsingFeature and Color Information. // Computer Graphics Forum 32. 2013. 7, 121–130.

[22] Kalantari N., Sen P. Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms. // Computer Graphics Forum 32. 2013. 2, 93–102.

[23] Kalantari N., Bako S., Sen P. A Machine Learning Approach for Filtering Monte Carlo Noise. // ACM Transactions on Graphics 34. 2015. 4, 122:1–122:12.

[24] He K., Sun J., Tang X. Guided Image Filtering. // IEEE Transactions on Paern Analysis and Machine Intelligence 35. 2013. 6, 1397–1409.

[25] Dammertz H., Sewtz D., Hanika J., Lensch H. Edge-Avoiding A-Trous Wavelet Transform for fast Global Illumination Filtering. // In High Performance Graphics. 2010. 67–75.

[26] Delbracio M., Muse P., Buades A., Chauvier J., Phelps N., Morel J. Boosting Monte Carlo Rendering by Ray Histogram Fusion. // ACM Transactions on Graphics 33. 2014. 1, 8:1–8:15.

[27] Meyer M., Anderson J. Statistical Acceleration for Animated Global Illumination. // ACM Transactions on Graphics 25. 2006. 3, 1075–1080.

[28] Zimmer H., Rousselle F., Jakob W., Wang O., Adler D., Jarosz W., Sorkine-Hornung O., Sorkine-Hornung A. Path-space Motion Estimation and Decomposition for Robust Animation Filtering. // Computer Graphics Forum 34. 2015. 4, 131–142.

[29] Nehab D., Sander P., Lawrence J., Tatarchuk N., Isidoro J. Accelerating Real-time Shading with Reverse Reprojection Caching. // In Graphics Hardware. 2007. 25–35.

[30] Chen S., Williams L. View Interpolation for Image Synthesis. // In Proceedings of SIGGRAPH 93. 1993. 279–288.

[31] Karis B. High-Qality Temporal Supersampling. In SIGGRAPH Courses: Advances in Real-Time Rendering in Games. 2014.

[32] Yang L., Nehab D., Sander P., Sitthi-amorn P., Lawrence J., Hoppe H. Amortized Supersampling. // ACM Transactions on Graphics 28. 2009. 5, 135:1–135:12.

[33] Patney A., Salvi M., Kim J., Kaplanyan A., Wyman C., Benty N., Luebke D., Lefohn A. Towards Foveated Rendering for Gaze-Tracked Virtual Reality. // ACM Transactions on Graphics. 2016. 35, 6, 179:1–179:12.

[34] Fattal R., Agrawala M., Rusinkiewicz S.: Multiscale shape and detail enhancement from multi-light image collections. // In SIGGRAPH’07: ACM SIGGRAPH 2007 papers (2007), p. 51.

[35] Schied C., Kaplanyan A., Wyman C., Patney A., Chaitanya C., Burgess J., Liu S., Dachsbacher C., Lefohn A., Salvi M. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. // In Proceedings of HPG’17. 2017. 12 pages.

[36] Saito T., Takahashi T. Comprehensible Rendering of 3-D Shapes. // Computer Graphics. 1990. 197–206.

[37] Jouppi N., Chang C. Z3: An Economical Hardware Technique for High-Qality Antialiasing and Transparency. // In SIGGRAPH/Eurographics Workshop on Graphics Hardware. 1999.

[38] Kerzner E., Salvi M. Streaming G-buffer Compression for Multi-sample Anti-aliasing. // In Proceedings of High Performance Graphics. 2014. 1–7.

[39] Stein C. Estimation of the Mean of a Multivariate Normal Distribution. // The Annals of Statistics. November 1981. 9 (6): 1135–1151.

[40] Rousselle F., Knaus C., Zwicker M. Adaptive sampling and reconstruction using greedy error minimization. // ACM Trans. Graph. (Proceedings of SIGGRAPH Asia 2011). 2011. 30,6, Article 159.

[41] Sen P., Darabi S. On filtering the noise from the random parameters in Monte Carlo rendering. // ACM Trans. Graph. 2012. 31, 3, Article 18.

[42] Overbeck R., Donner C., Ramamoorthi R. Adaptive wavelet rendering. // ACM Trans. Graph. (Proceedings of SIGGRAPH Asia 2009). 2009. 28, 5, Article 140.

[43] Rumelhart D., Hinton G., Williams R. Learning representations by back-propagating errors. // Nature. 1986. 323(Oct.), 533–536.

[44] NVIDIA RTX™ platform. [URL]: https://developer.nvidia.com/rtx

[45] Schied C., Peters C., Dachsbacher C. Gradient Estimation for Real-Time Adaptive Temporal Filtering. // Proceedings ACM Computer Graphics Interact. Tech. 2018. 1, 2, Article 24 (August 2018), 16 pages.

[46] Wang Z., Bovik A., Sheikh H., Simoncelli E. Image quality assessment: from error visibility to structural similarity. // IEEE transactions on image processing. 2004. 13, 4, 600–612.

Приложение А

Рис. 8. Исходное необработанное изображение.

Рис. 9. Эталонное изображение.

Рис. 10. Изображение, синтезированное алгоритмом EAW.

Рис. 11. Изображение, синтезированное алгоритмом SVGF.

Рис. 12. Изображение, синтезированное алгоритмом SBF.

Рис. 13. Изображение, синтезированное алгоритмом LBF.

Оглавление

1. Введение 3
2. История вопроса 4
3. Алгоритм Edge-avoiding à-trous wavelets (EAW) 7
4. Алгоритм Spatiotemporal Variance-Guided Filtering (SVGF) 9
5. Алгоритм SURE-based Optimization for Adaptive Sampling and Reconstruction (SBF) 11
6. Алгоритм LBF – Machine Learning Approach for Filtering Monte Carlo Noise 13
7. Анализ 15
8. Заключение 19
Литература 19
Приложение А 23
Оглавление 26