Опыт реализации Gradient Domain Metropolis Light Transport

Павлов Д.С.: НИУ Высшая школа экономики
Фролов В.А.: ИПМ им. М.В. Келдыша РАН,
МГУ имени М.В, Ломоносова
dspavlov_1@edu.hse.ru, vfrolov@graphics.cs.msu.ru

Аннотация

С недавнего времени в области рендеринга изображений с помощью Монте-карло методов появились алгоритмы, основанные на рендеринге в так называемом пространстве градиентов. В настоящее время эти методы не используются широко на практике из-за большого количества нюансов и высокой сложности их реализации и, как следствие, недостаточно большом количестве исследований их преимуществ и недостатков. Наша работа направлена на заполнение этого пробела. В работе рассматривается применение рендеринга в пространстве градиентов к алгоритму переноса света Метрополиса, описываются детали реализации и приводятся сравнения с существующими схожими методами.

1 Введение

Рендеринг фотореалистичных изображений является трудоемким процессом. В общем виде, задача состоит в том, чтобы вычислить интеграл освещенности или, что тоже самое, решить *уравнение рендеринга:*

$$I(\varphi_r,\theta_r)=\int_{\varphi_i}\int_{\theta_i} L(\varphi_i,\theta_i)R(\varphi_i,\theta_i,\varphi_r,\theta_r)\cos(n,l_{\varphi_i,\theta_i})d\varphi_i d\theta_i$$

где L – функция падающего освещения, а R – функция, описывающая отражательные свойства материала.

Поскольку аналитическое решение возможно только для частных случаев, в основном, уравнение рендеринга решается численно, с помощью методов интегрирования Монте-Карло. В качестве метода интегрирования обычно используется алгоритм трассировки путей [Kajiya 1986].

Несмотря на то, что трассировка путей (Path Tracing, PT) позволяет корректно рассчитывать освещение и учитывать множество физических эффектов, для того, чтобы получить изображение без шума, требуется большое количество выборок (сэмплов) и, соответственно, времени.

2 Обзор существующих методов

В оригинальном алгоритме трассировки путей [Kajiya 1986], для каждого пикселя изображения считается некоторое фиксированное количество сэмплов, после чего они записываются в изображение. Один из методов более оптимального распределения сэмплов в пространстве изображения - это Адаптивная Трассировка Путей [Pharr 2016; Mitchell 1991]. Основная идея заключается в том, чтобы на некоторых этапах рендеринга оценивать дисперсию между вкладами в пиксель / соседними пикселями / блоками пикселей и при достижения некоторого порогового значения, прекращать вычисление данных пикселей. Этот метод даёт выигрыш на простых сценах, но имеет ряд недостатков. Общая проблема у алгоритмов адаптивной трассировки путей – это выбор подходящей метрики для оценки сходимости изображений. Метод описанный в [Pharr 2016] для оценки сходимости конкретного пикселя использует лишь информацию о первичных вкладах в изображение и среднем значением самого пикселя и не использует информацию о прилежащих пикселях. Это позволяет избегать излишнего сэмплирования относительно-простых областей изображения (например, фона с картой окружения), но не улучшает сходимость в целом: на сложных областях адаптивное сэмплирование, используя тот же базовый метод (например, PT), по-прежнему работает не эффективно.

Метод, описанный в [Mitchell 1991] использует блоки пикселей и вычисляет дисперсию по всему блоку, что позволяет минимизировать кол-во затраченных сэмплов на гладкие области, но метод плохо работает при рассчёте таких эффектов как *размытие в движении* и *глубина резкости*. Кроме того, адаптивная трассировка путей, точно так же, как и обычная трассировка путей, не способна эффективно рассчитывать сцены со сложным освещением (не прямым освещением от узких, ярко освещённых участков поверхности).

Одним из хорошо зарекомендовавших себя способов рассчёта сложного освещения является алгоритм переноса света Метрополиса [Veach 1997; Kelemen 2002]. Он позволяет находить пути, которые вносят существенный вклад в изображение и сосредотачивать большее количество сэмплов в окрестности этих путей. Однако, в отличии от трассировки путей, этот алгоритм работает не в пространстве изображения, а в пространстве путей. Это означает, что невозможно выбрать и вычислить какой-то отдельно взятый пиксель --- можно вычислять только всё изображение целиком.

Рассмотрим алгоритм Метрополиса-Гастингса на примере вычисления интеграла от функции *g(x)*.

$$\int g(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{g(x_i)}{p^*(x_i)} = \frac{C}{N}\sum_{i=1}^{N}\frac{g(x_i)}{f(x_i)}, \quad (1)$$

где *g(x)* – некоторая интегрируемая функция; $x_i \sim p^*(x_i) = \frac{f(x)}{C}$; *C* – интеграл от *f* (называемый также константой нормализации), посчитанный какой-то другой техникой, обычно методом Монте-Карло; *f(x) – целевая функция* (является скалярной величиной, несмотря на то, что *g(x)* может быть векторной величиной). В алгоритме переноса света Метрополиса [Veach 1997, Kelemen 2002] *g(x)* представляет собой вклад некоторого пути в пиксел и *f(x)* представляет собой скалярную функцию, пропорционально которой MLT строит распределение (целевая функция). В оригинальной статье [Veach 1997, Kelemen 2002], *f(x)* – представляет собой линейную комбинацию компонентов спектра.

Однако, функция, которую необходимо проинтегрировать не обязательно должна быть линейно пропорциональна целевой функции [MacKay 2003]. Например, в работе [Hoberock 2010] в зависимости от типа поверхностей целевая функция взвешивалась с разными весами, тем самым вынуждая алгоритм чаще выбирать пути с конкретным сценарием переноса света.

В работе [Lehtinen 2013] был предложен метод, названный *Gradient-domain Metropolis Light Transport (GDMLT)*. В GDMLT используется похожая идея на рассмотренную выше, но распределение строится пропорционально изображениям градиентов. Для этого на ряду с обычным путём дополнительно проводится смещенный путь через соседний пиксел и считается их разница. Затем строится целевая функция, которая состоит из взвешенных яркостей вклада обычного пути и разницы между обычным путём и смещенным. Это позволяет алгоритму чаще сэмплировать пути, которые находятся на границах в пространстве изображения. Далее мы более подробно рассмотрим работу [Lehtinen 2013] и аналоги.

2.1 Рендеринг в пространстве градиентов

Градиент изображения имеет свойство разреженности, но при этом хранит точно такую же информацию, что и оригинальное изображения (если считать, что информация сконцентрирована на границах).

Идея рендеринга в пространстве градиентов состоит в том, чтобы одновременно с обычным изображением вычислять ещё и градиенты. Полагая далее, что градиент будет менее шумным (т.к. он разреженный) в сравнение с обычным изображением, восстанавливать итоговое изображение по градиентам.

В работе [Kettunen 2015] был представлен алгоритм, названный Gradient-domain Path Tracing (GDPT). Он представляет собой модификацию оригинальной трассировки путей. Для некоторого пикселя с помощью обычного PT проводится базовый путь и вычисляется вклад в изображение:

$$I_j = h_j(x) * \int_{\Omega} f(x)d\mu(x),$$

где $h_j(x)$ - фильтр для *j*-того пикселя; Ω - пространство всех путей конечной длины; *f(x)* кол-во света, достигающее сенсор через путь *x*; $d\mu(x)$ - произведение площадных мер $\prod_{i=0}^{k} dA(x_i)$, где $x_0,...,x_k$ - вершины пути и $A(x_i)$ - дифференциальная площадь у соответствующей вершины x_i [Pharr 2016, ch. 5.4; Veach 1997, ch. 4.1].

Для каждого такого пути проводится дополнительный луч через каждый соседний пиксель и соединяется с базовым путем, как только встречаются 2 “незеркальные” вершины подряд на обоих путях. Получается новый путь, имеющий такие же вершины, что и базовый путь, кроме первых. Назовём его смещенным путём (рис. 1).

Далее считается разница между базовым и смещенным путём и записывается вклад в изображение с градиентом.

$$\Delta_{i,j} = \left(h(x) * \int_{\Omega} f(x) d\mu(x) \right)(x_i) - \left(h(x) * \int_{\Omega} f(x) d\mu(x) \right)(x_j)$$

В итоге получается 3 изображения: (1) оригинальное изображения, (2) изображение с градиентом по оси X, (3) изображение с градиентом по оси Y.

Далее решается задача оптимизации:

$$\arg\min_{I} \left(\left\| \begin{pmatrix} H^{dx} I \\ H^{dy} I \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\| + \left\| \alpha (I - I^{g}) \right\| \right)$$

Здесь I^{dx}, I^{dy} - градиенты изображения; I^{g} - оригинальное изображение, посчитанное вместе с градиентами; α - коэффициент, регулирующий степень влияния оригинального изображения на реконструкцию (при $\alpha = 0$, используются только градиенты); I – искомое изображение; H^{dx}, H^{dy} - матрицы, вычисляющие *x, y* градиенты из изображения I (при реализации, их можно просто представить в виде функции, которая в цикле считает конечные разницы *(j+1) – (j)* из изображения); $\| \ \|$ - L1 или L2 норма.

Другими словами, необходимо найти такое изображение, чьи градиенты максимально приближают градиенты, полученные трассировкой путей.

Такой метод позволяет значительно уменьшить шум, но имеет всё те же недостатки, что и оригинальная трассировка путей.

Что касается алгоритма GDMLT из работы [Lehtinen 2013], в ней был предложен метод, который представляет собой модификацию алгоритма переноса света Метрополиса. Аналогично GDPT, одновременно с обычным изображением вычисляются градиенты изображения и затем, решая уравнение Пуассона, востанавливается финальное изображение. Дополнительно, GDMLT использует разницу между базовым и смещенным путём в своей целевой функции, что позволяет передвигать сэмплер используя информацию о соседних пикселях.

3 Реализация метода

В оригинальной статье [Lehtinen 2013] алгоритм GDMLT строится поверх так называемого “Veach-style” или “Path space” Metropolis Light Transport, который работает в пространстве путей. Данная статья описывает реализацию GDMLT поверх Primary Sample Space MLT [Kelemen 2002] (далее PSSMLT), который производит мутации не в пространстве путей, а в пространстве случайных чисел. А именно, изменяет значения в единичном гиберкубе, по которым затем пути строятся.

Мы будем называть реализованный нами метод *Gradient Domain Primary Sample Space Light Transport* или *GDPSSMLT* сокращённо.

3.1 Алгоритм переноса света Метрополиса

Для удобства запишем уравнение рендеринга в следующем виде:

$$I_j = \int_{\Omega} h_j(\bar{x}) f^*(\bar{x}) d\mu(\bar{x}), \qquad (2)$$

где $h_j(\bar{x})$ - фильтр для *j*-того пикселя; Ω - пространство всех путей конечной длины; $f^*(\bar{x})$ - кол-во света, которое переносит путь x; $d\mu(\bar{x})$ - произведение мер $\prod_{i=0}^{k} dA(x_i)$.

Metropolis Light Transport использует алгоритм Метрополиса-Гастингса (1) для решения уравнения рендеринга (2):

$$I_j \approx \frac{C}{N} \sum_{i=1}^{N} \frac{h_j(\bar{x}_i) f^*(\bar{x}_i)}{f(\bar{x}_i)},$$

где $f(\bar{x}_i)$ - скалярная функция, представляющая собой яркость вклада $f^*(\bar{x})$; C – константа нормализации, посчитанная с помощью алгоритма трассировки путей.

3.2 Naïve PT как базовый метод

В оригинальной статье [Kelemen 2002, Veach 1997] для построения путей используется Двунаправленная Трассировка Путей [Veach 1997, ch. 10]. В нашей реализации алгоритма мы строим PSSMLT поверх наивной трассировки путей (т.е. без теневых лучей). Такой подход менее эффективен на практике, но он позволяет нам лучше исследовать свойства собственно MLT и GDMLT т.к. в принципе исключает

эффективные стратегии генерации выборок для источников небольшого размера.

Также следует отметить, что в большинстве реализаций MLT первичное и вторичное освещение вычисляется раздельно (первичное освещение считается, например, при помощи PT а вторичное при помощи MLT). Мы намеренно отказались от такого разделения чтобы исследовать свойства GDMLT т.к. одним из позиционируемых преимуществ GDMLT над MLT является лучшая сходимость на сценах с сильным перепадом уровней освещённости [Lehtinen 2013].

Поясним, что одним из недостатков обыкновенного (с линейно-зависимой функцией вклада) MLT является то, что в пространстве изображения он сосредотачивает сэмплы пропорционально яркости. Поэтому, если разделения на первичное и вторичное освещение нет, алгоритм будет распределять большую часть вычислительных ресурсов на пути, ответсвтенные за вклад только лишь первичного освещения. А это освещение не является сложным с точки зрения современных алгоритмов с многократной выборкой по значимостью (например, PT или BPT [Veach 1997]), и не должно оттягивать на себя вычислительные ресурсы MLT.

3.3 Вычисление градиентов

В первую очередь нам необходимо посчитать градиенты изображения. То-есть, попиксельную разницу по оси x и по оси y. Обозначим обычный вклад в изображение I_i, разницу по оси x: $I_i^{dx} = I_{i+1} - I_i$, разницу по оси y: $I_i^{dy} = I_{i+1} - I_i$.

Обозначим за $T(\bar{x})$ *функцию сдвига,* которая принимает на вход некоторый путь $\bar{x}$, проходящий через пиксель (s_x, s_y) и изменяет его так, чтобы он проходил через пиксель $(s_{x+xOffset}, s_{y+yOffset})$

Тогда вычисление попиксельной разницы выглядит следующим образом:

$$I_j^{dx} = I_{j+1} - I_j = \tag{3}$$

$$\int_{\Omega} h_j(\bar{x}) \left[f^*(T_{1,0}(\bar{x})) \frac{d\{\mu \circ T_{1,0}\}}{d\mu}(\bar{x}) - f^*(\bar{x}) \right] d\mu(\bar{x})$$

где $\frac{d\{\mu \circ T_{1,0}\}}{d\mu}$ - якобиан смещенного пути, возникающий при переходе из пространства сдвинутых путей в обычное пространство путей Ω. Якобиан нужен для того, чтобы учесть изменение в плотности меры. Если при вычисление вклада от смещенного пути проигнорировать умножение на Якобиан, то это может привести к видимым артефактам (рис. 1).

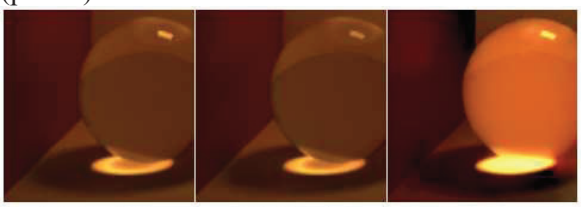

Рис 1. Артефакты, возникающие при игнорировании изменения меры пространства путей [Lehtinen 2013].

Можно сказать что Якобиан представляет из себя регулязатор, который взвешивает часть смещенного пути до соединения, как если бы это был такой-же путь, но полученный случайно в соседнем пикселе. Регуляризатор необходим, поскольку смещённый путь строится не стохастическим способом, а детерминировано. Идея похожа на то, как при построении теневого луча до источника света, когда теневой луч строится детерминировано, но его вклад должен быть учтён, как если бы он был получен случайно.

Таким образом, каждый пиксель должен дополнительно хранить градиент по оси x и градиент по оси y.

В итоге имеется 4 потенциальных смещенных пути для каждого построенного базового пути: (-1, 0); (1, 0); (0, -1); (0, 1).

В отличии от GDPT, где для каждого базового пути проводятся все 4 смещенных пути и вычисляются 4 попиксельных разницы, в GDMLT проводится лишь один смещенный путь. Направление сдвига выбирается случайно. Можно представить это, как произведение пространства путей и множества сдвигов.

$$\Omega' = \Omega \times \{(-1,0),(+1,0),(0,-1),(0,+1)\}$$

Мутации в этом пространстве путей происходят аналогично PSSMLT. Направление сдвига выбирается равновероятно, как при обычной мутации, так и при Large Step мутации.

Так как сдвиг может быть как положительным, так и отрицательным, то и пиксель, в которой записывается вклад - варьируется.

Для положительных сдвигов, вклад считается, как $I_i^{dx} = I_{i+1} - I_i$ и записывается в *i*-й пиксель. Для отрицательных сдвигов, вклад считается, как : $I_{i-1}^{dx} = I_i - I_{i-1}$ и записывается в *i-1*-й пиксель.

3.4 Построение смещенных путей

После того, как проводится первый луч через соседний пиксель и находится первая вершина, смещенный путь необходимо соединить с базовым путём. Это можно сделать, только если текущая вершина смещенного пути и базового пути являются незеркальными и следующая вершина базового пути является незеркальная. То-есть, смещенный путь проводится до тех пор, пока не сможет соединиться с базовым (рис. 2).

Условно, каждый смещенный путь можно разделить на 3 части:

1) Начальная вершина (камера).
2) Путь, состоящий из зеркальных вершин и заканчивающийся на незеркальной вершине.
3) Оставшаяся не измененная часть, скопированная из базового пути.

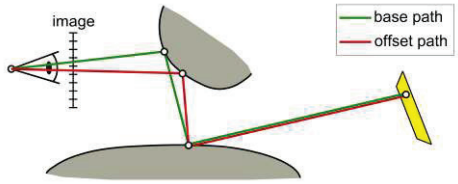

Рис 2. Построение смещённого пути аналогично работе [Lehtinen2013]

При трассировке смещенного пути, исходящие направления выбираются не стохастически, а при помощи соответствующих вершин базового пути. Для этого вычисляется half-vector в вершине базового пути, проецируется на вершину смещенного пути, и относительно него выбирается исходящее направление.

Данный подход был представлен в статье [Kettunen 2015] (GDPT) и подходит для реализации GDMLT. В оригинальной работе [Lehtinen 2013] смещенный путь строится с помощью специальной техники построения мутаций, называемой Manifold Exploration [Jacob 2012]. Manifold Exploration может быть использована только для MLT в пространстве путей (Path Space MLT)[Veach 1997], поэтому для построения смещённого пути мы выбрали альтернативный вариант, реализованный в [Kettunen 2015].

Теперь рассмотрим вычисление Якобиана для различных типов поверхностей.

Пусть, $\bar{x}$, $\bar{y}$ – базовый и смещенный путь, w_i^x, w_i^y – входящее направление луча для вершины базового и смещенного пути, h^x, h^y – half-vector базового и смещенного пути, n_1^x, n_2^x - индексы преломления для базового пути, n_1^y, n_2^y - аналогично для смещенного пути, x_1^x, x_2^x – вершины базового пути, x_1^y, x_2^y – вершины смещенного пути, между которыми происходит соединение, θ – угол между нормалью вершины и сегментом $x_1 x_2$.

Тогда, Якобиан будет вычисляться следующим образом:

Для зеркального отражения:

$$\left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| = \left|\frac{\partial \omega_i^y}{\partial h^y}\right| \left|\frac{\partial h^x}{\partial \omega_i^x}\right| = \frac{\omega_o^y \cdot h^y}{\omega_o^x \cdot h^x}$$

Для преломления:

$$\left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| = \left|\frac{\partial \omega_i^y}{\partial h^y}\right| \left|\frac{\partial h^x}{\partial \omega_i^x}\right| = \frac{\left|\omega_i^y + \frac{n_2^y}{n_1^y}\omega_o^y\right| \omega_i^x \cdot h^x}{\left|\omega_i^x + \frac{n_2^x}{n_1^x}\omega_o^x\right| \omega_i^y \cdot h^y}$$

Для незеркального отражения (соединение смещенного пути с базовым):

$$\left|\frac{\partial \omega_i^y}{\partial \omega_i^x}\right| = \left|\frac{\partial \omega_i^y}{\partial x_2^y}\right| \left|\frac{\partial x_2^x}{\partial \omega_i^x}\right| = \frac{\cos\theta_2^y \left|x_1^x - x_2^x\right|^2}{\cos\theta_2^x \left|x_1^y - x_2^y\right|^2}$$

Для оставшейся части смещенного пути, в каждой вершине Якобиан равен единице.

Конечный Якобиан, используемый в формуле (3) представляет собой произведение Якобианов на каждой вершине пути.

3.5 Построение целевой функции

В качестве целевой функции используется взвешенная сумма вклада базового пути и градиента.

$$f(\bar{z}) = \left\| f^*(\bar{z}) \right\| + \alpha \left(\frac{1}{4} \left\| f^*(\bar{x}) \right\| \right),$$

где $f^*(\bar{z})$ - абсолютное значение градиента, $f^*(\bar{x})$ - вклад базового пути, $\| \ \|$ - некоторая скалярная функция, обычно это яркость (*luminance*), представляющая собой линейную комбинацию всех спектров вклада; $\alpha \in [0;1]$- регуляризатор. Стоит заметить, что он не обязан совпадать с параметром альфа при реконструкции и может варьироваться в зависимости от сцены.

Так как алгоритм Метрополиса-Гастингса требует, чтобы целевая функция была скалярной величиной, мы используем яркость вклада, точно так же, как и в оригинальном PSSMLT. Так же, стоит заметить, что градиент $f^*(\bar{z})$ может принимать отрицательные значения. Чтобы избежать случаев, когда при подсчётё яркости вклада (*luminance*) один или несколько отрицательных спектров значительно уменьшают яркость вклада, используются абсолютные значения спектров (а не абсолютное значение уже посчитанной яркости).

3.6 Реконструкция изображения

На последнем шаге необходимо восстановить изображение по градиентам и обычному изображению.

Для этого нужно решить уравнение Пуассона, что можно записать в виде следующей задачи оптимизации

$$\arg\min_I \left(\left\| \begin{pmatrix} H^{dx} I \\ H^{dy} I \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\| + \left\| \alpha \left(I - I^g \right) \right\| \right),$$

где I^{dx}, I^{dy} - градиенты изображения; I^g - оригинальное изображение, посчитанное вместе с градиентами; α - коэффициент, регулирующий степень влияния оригинального изображения на реконструкцию; I – искомое изображение; H^{dx}, H^{dy} - матрицы, вычисляющие *x, y* градиенты из изображения *I,* $\| \ \|$ - L1 или L2 норма.

Данную задачу можно решать различными способами. Авторы статьи [Lehtinen2013] используют для этого методы наименьших квадратов. При использовании L2 нормы, задача решается с помощью *сопряженных градиентов* [Shewchuk 1994], при использовании L1 нормы, используется метод, называющийся *Iteratively Reweighted Least Squares* [Burrus 2012], который сводится к решению серии задач L2 оптимизации, так же с помощью сопряженных градиентов.

Стоит упомянуть, что существует популярный способ решения уравнения Пуассона в gradient-domain задачах на основе дискретного косунусного преобразования [Bhat 2008].

4 Сравнения и результаты

Мы сравниваем реализованный алгоритм (GDPSSMLT) со следующими методами: Трассировка путей (PT), Трассировка путей в пространстве градиентов (GDPT) и Перенос света Метрополиса с мутациями в пространстве случайных чисел (PSSMLT).

В качестве тестовой сцены используется модель Sponza с маленьким и ярким источником света, расположенным сверху над моделью. Ещё раз подчёркнём, что источник света сэмплируется неявно (т.е. только тогда когда луч в него случайно попадёт), а теневые лучи мы намеренно не использовали. Все изображения были посчитаны в разрешении 800x800 пикселей за фиксированное время ~3 часа. (Рис 3, 4, 5, 6, 7). Для реализации трассировки лучей мы использовали библиотеку трассировки лучей Intel Embree [Wald et. All 2014].

На рисунке 3 показано изображение, полученное с помощью наивной Трассировки Путей. После 3 часов рендеринга оно всё еще является чрезвычайно шумным, не позволяя оценить облик сцены даже приблизительно.

На рисунке 4 показано изображение, полученное с помощью трассровки путей в просранстве градиентов (GDPT). Основное изображение, как и градиенты всё ещё сильно зашумлено, что приводит к видимым артефактам при реконструкции.

На рисунке 5 показано изображение, полученное с помощью алгоритма переноса света Метрополиса с мутациями в пространстве случайных чисел (PSSMLT). В сравнение с трассировкой путей, данный метод работает достаточно хорошо. В светлых областях наблюдается небольшой шум. Однако в темных областях шум всё еще неприемлимо-высок. На рис. 3-6 показано изображение, полученное с помощью нашей реализации GDMLT, названной нами GDPSSMLT и сравнение с аналогами.

5 Выводы

Вывод №1: для обыкновенного Монте-Карло (PT против GDPT) рендеринг в пространстве градиентов несмотря на артефакты существенно улучшает изображение для сложных сценариев, позволяя оценить внешний вид сцены и освещения. Это происходит благодаря тому, что градиенты являются разреженными и поэтому являются менее шумными, чем обычное изображение. После реконструкции эти гладкие области сходятся достаточно быстро.

Вывод №2: Среди всех тестируемых методов PSSMLT даёт наиболее точную оценку освещения.

Вывод №3: GDMLT не полностью решает проблему сэмплирвоания алгоритмом Метрополиса преимущественно ярких областей, поскольку лишь оттягивает часть сэмплов из ярких областей в области границ на изображении. Однако, т. к. для целевой функции используется взвешенная сумма (раздел 3.5) в которую яркость всё-равно входит с большим весом, проблема рассчёта преимущественно ярких областей сохраняется.

Вывод №4: изображения, полученные при помощи GDPSSMLT полностью лишены шума в отличие от PSSMLT. Однако, они получаются размытыми (в том числе размываются текстуры). Это позволяет позволяет нам поставить следующий научный вопрос: действительно ли GDMLT имеет преимущество над обычым MLT с последующим применением денойзинга (например, на основе метода Non Local Means). На наш взгляд ответ на данный вопрос в настоящий момент не известен, и это является предметом наших будущих исследований.

Список литературы

James T. Kajiya. 1986. *The rendering equation.* In Proceedings of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH '86), David C. Evans and Russell J. Athay (Eds.). ACM, NY, USA, 143-150.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Don P. Mitchell. 1991. *Spectrally optimal sampling for distribution ray tracing.* In Proceedings of the 18th annual conference on Computer graphics and interactive techniques (SIGGRAPH '91). ACM, New York, NY, USA, 157-164.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. Ph.D. Dissertation. Stanford University, Stanford, CA, USA. Advisor (s) Leonidas J. Guibas. AAI9837162.

Csaba Kelemen, László Szirmay-Kalos, György Antal and Ferenc Csonka. 2002. *Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm.* EUROGRAPHICS 2002 / G. Drettakis and H.-P. Seidel. 2002, v.21, № 3.

David J. C. MacKay. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

Jared Hoberock and John C. Hart. 2010. "Arbitrary importance functions for metropolis light transport." *Computer Graphics Forum*. Blackwell Publishing Ltd. v.29, № 6.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand and Timo Aila. 2013. *Gradient-domain metropolis light transport*. ACM Trans. Graph. 32, 4, Article 95 (July 2013), 12 pages.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand and Matthias Zwicker. 2015. *Gradient-domain path tracing*. ACM Trans. Graph. 34, 4, Article 123 (July 2015), 13 pages. DOI: https://doi.org/10.1145/2766997

Wenzel Jakob and Steve Marschner. 2012. *Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport.* ACM Trans. Graph. 31, 4, Article 58 (July 2012), 13 pages.

Jonathan R Shewchuk. 1994. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Technical Report. Carnegie Mellon Univ., Pittsburgh, PA, USA.

C.S. Burrus. 2012. *Iterative reweighted least squares.* OpenStax-CNX Web Publication. http://cnx.org/content/m45285/1.12

Pravin Bhat, et al. 2008. *Fourier analysis of the 2D screened Poisson equation for gradient domain problems*. In European Conference on Computer Vision. Springer, Berlin, Heidelberg. 114-128.

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: a kernel framework for efficient CPU ray tracing. *ACM Trans. Graph.* 33, 4, Article 143 (July 2014), 8 pages.

Благодарности

Работа поддержана грантом РФФИ №16-31-60048 мол_а_дк.

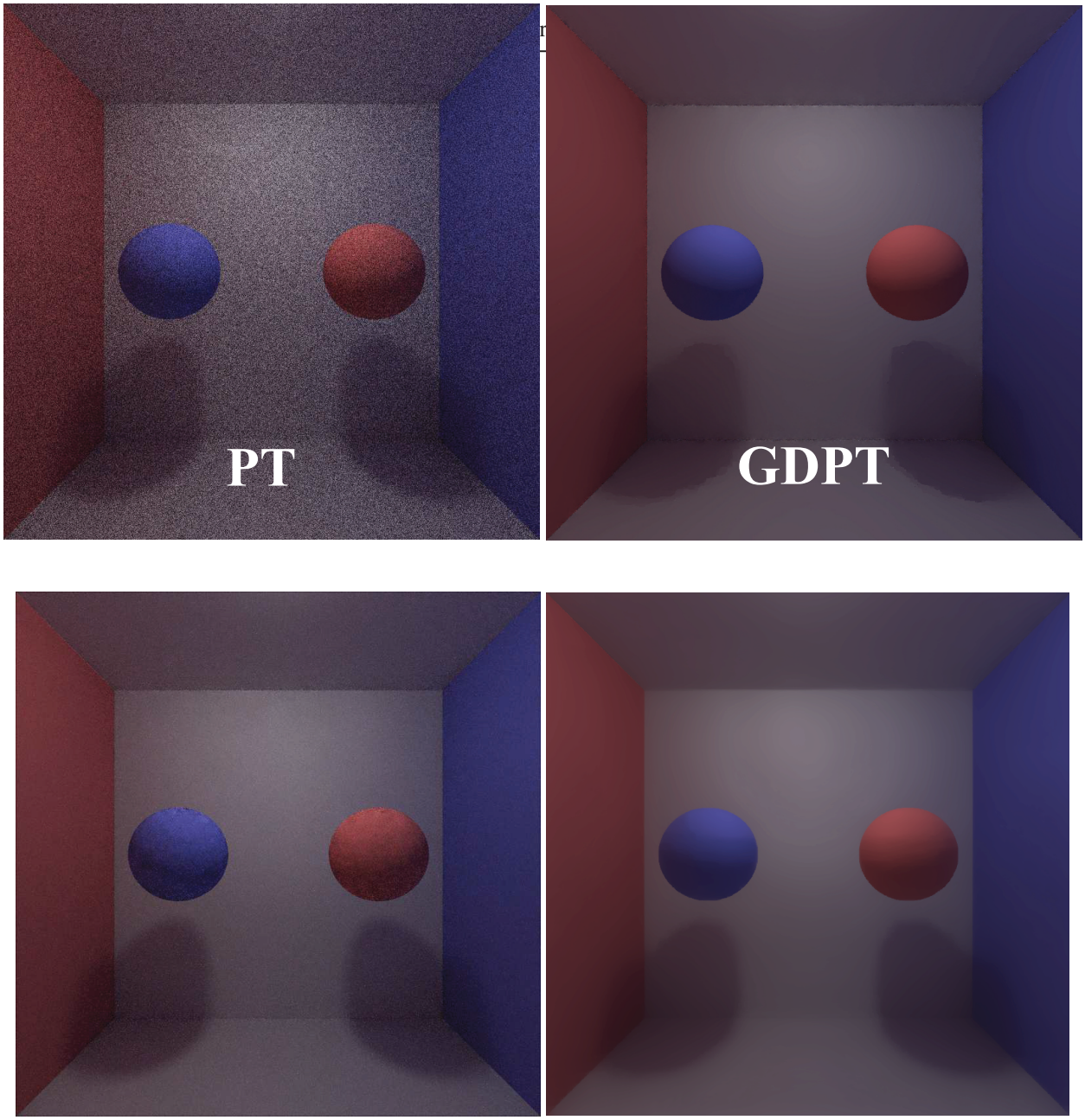

Рис. 3. Сравнение разработанного метода (GDPSSMLT) и аналогов

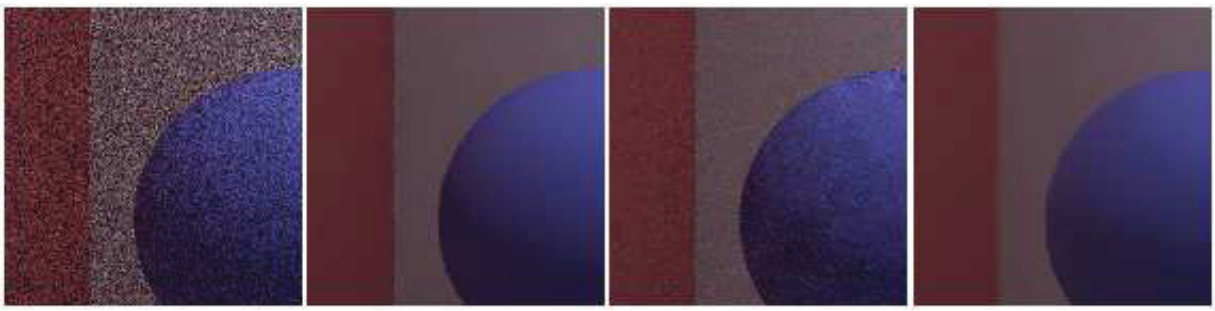

Рис. 4. Увеличенный фрагмент сравнения рис. 3.
Слева направо: PT, GDPT, PSSMLT, GDPSSMLT

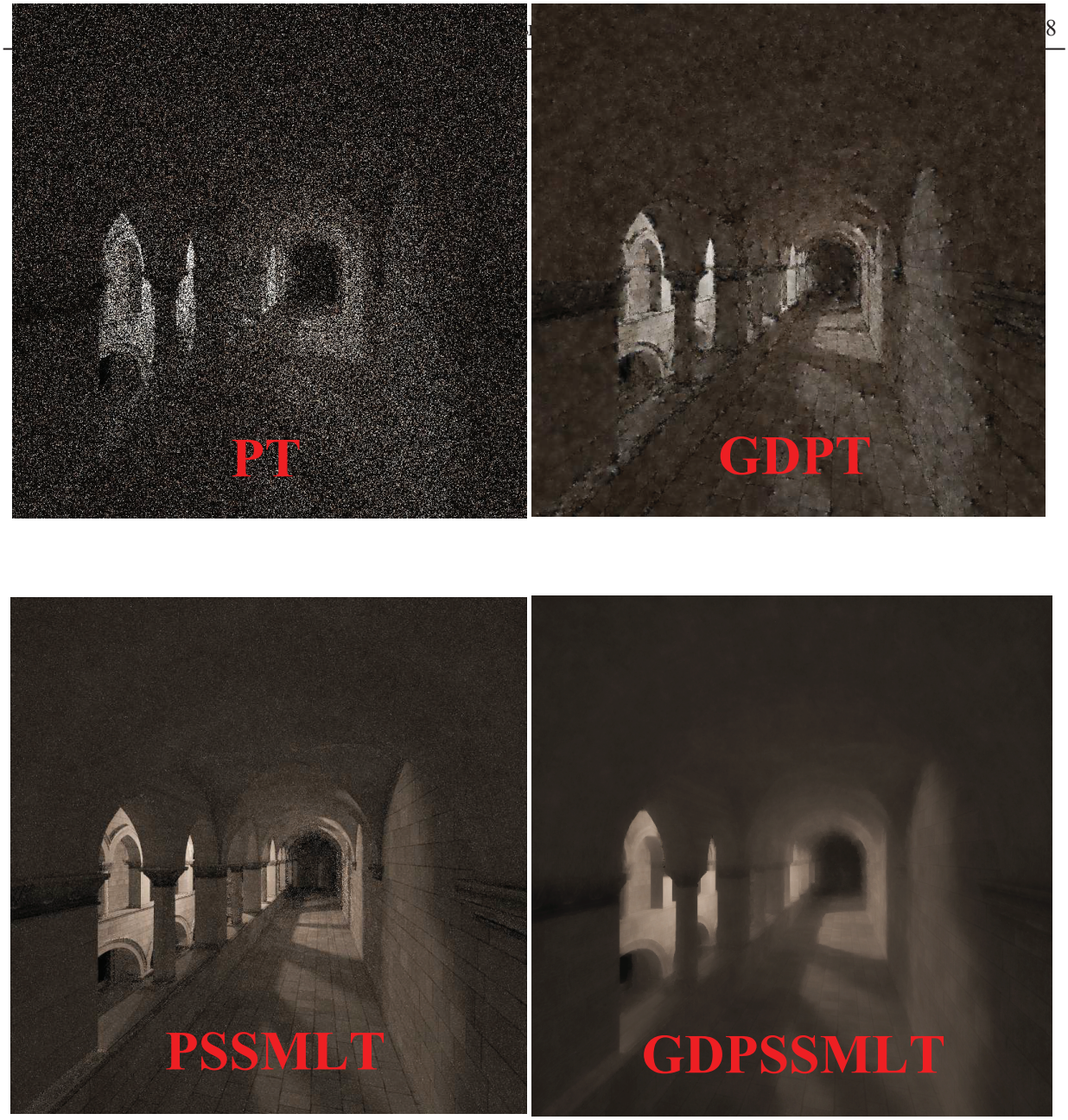

Рис. 5. Сравнение разработанного метода (GDPSSMLT) и аналогов

Рис. 6. Увеличенный фрагмент сравнения рис. 3.
Слева направо: PT, GDPT, PSSMLT, GDPSSMLT