Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Симиютин Борис Петрович

Удаление отброшенных теней с 3D моделей

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
Ген. директор ООО"Живой софт"
м. ф.-м. н. Семенов Д. А.

Рецензент:
Исследователь в области машинного обучения, Google
м. ф.-м. н. Мордвинцев А. С.

Санкт-Петербург
2018

Реферат

стр. 37, рис. 21, табл. 3

Данная работа посвящена изучению задачи удаления отбрасываемых теней с трехмерных компьютерных моделей. Эта задача часто возникает при подготовке моделей, полученных с помощью фотограмметрии, к использованию в виртуальных сценах с искусственным освещением. Целью работы является создание практического подхода для решения данной задачи. В результате анализа за базовый алгоритм был выбран подход для удаления теней с фотографий и выполнен переход к обработке 3D моделей. Для достижения стабильных результатов обработки предложен интерактивный инструмент для окончательной доводки результата, получанного на первом шаге. Численное и количественное сравнение показало превосходящие результаты по сравнению с алгоритмами, запущенными на скриншотах моделей.

Оглавление

Реферат **2**

Введение **4**

1. Используемые понятия и определения **5**
1.1. 3D модели 5
1.2. Тени 6

2. Обзор существующих подходов **8**
2.1. Практические подходы 8
2.1.1. 3D 8
2.1.2. 2D 10
2.2. Теоретические подходы 11
2.2.1. 3D 11
2.2.2. 2D 11
2.3. Сравнение и выводы 16

3. Предлагаемый подход **19**
3.1. Предлагаемый алгоритм 19
3.2. Гладкость множителя внутри умбры 20
3.2.1. Экстраполяция в связном случае 20
3.2.2. Экстраполяция в несвязном случае 21
3.3. Постпроцессинг 25
3.3.1. Недостаток базового алгоритма 25
3.3.2. Предлагаемое решение 25
3.3.3. Алгоритм 27

4. Реализация **28**

5. Сравнение **29**
5.1. Регрессионное сравнение 29
5.2. Сравнение в 3D 30
5.2.1. Датасет 30
5.2.2. Количественная оценка 30
5.2.3. Качественная оценка 32
5.3. Скорость работы 34

Заключение **35**

Список литературы **36**

Введение

В компьютерной графике объекты из трехмерного пространства представляются в виде моделей, состоящих из геометрии и текстуры. Одним из способов получения моделей является фотограмметрия, которая позволяет восстановить объект по множеству его фотографий с разных ракурсов. При этом получается реалистичный результат, но на текстуре остается информация об освещении, имевшем место при съемке. Для использования модели в виртуальном пространстве необходимо иметь текстуру без этой информации. Наиболее заметным и определяющим возможность использования модели эффектом являются тени. Для их минимизации советуют проводить съемку в пасмурную погоду, что не всегда возможно в солнечных странах. Таким образом, встает вопрос постобработки модели для компенсирования теней и получения текстуры без этой компоненты информации об освещении.

Целью настоящей работы является проанализировать существующие методы для удаления теней с 3D моделей и фотографий, и на их основе разработать практический подход для удаления теней с 3D моделей.

1. Используемые понятия и определения

1.1. 3D модели

3D модели представляются как множество вершин в трехмерном пространстве, сгруппированных в геометричекие примитивы, чаще всего треугольники. У каждой вершины есть атрибуты, такие как цвет, нормаль или текстурные координаты.

Текстура модели обычно содержится в изображении, называемом текстурным атласом. Каждая вершина треугольника имеет отображение на атлас в виде атрибута текстурной координаты. При отрисовке объекта для каждой точки берется цвет с текстурного атласа, соответствующий текстурной координате в этой точке.

Уложить текстуру всего объекта на атлас без разрезов в общем случае может быть невозможно по топологическим причинам. Таким образом, обычно на атласе есть больше одной связной области, соответствующей связной области на поверхности объекта. Такие области называются чартами.

Рис. 1: Геометрия и текстурный атлас

1.2. Тени

Тень делится на следующие регионы:

- **Собственная тень:** часть тени, которая обеспечивается только углом между нормалью к поверхности и направлением на источник света
- **Отброшенная тень:** часть тени, которая обеспечивается геометрией на пути световых лучей
- **Умбра:** области отброшенной тени, в которых источник света заслонен полностью
- **Пенумбра:** области отброшенной тени, в которых источник света заслонен частично

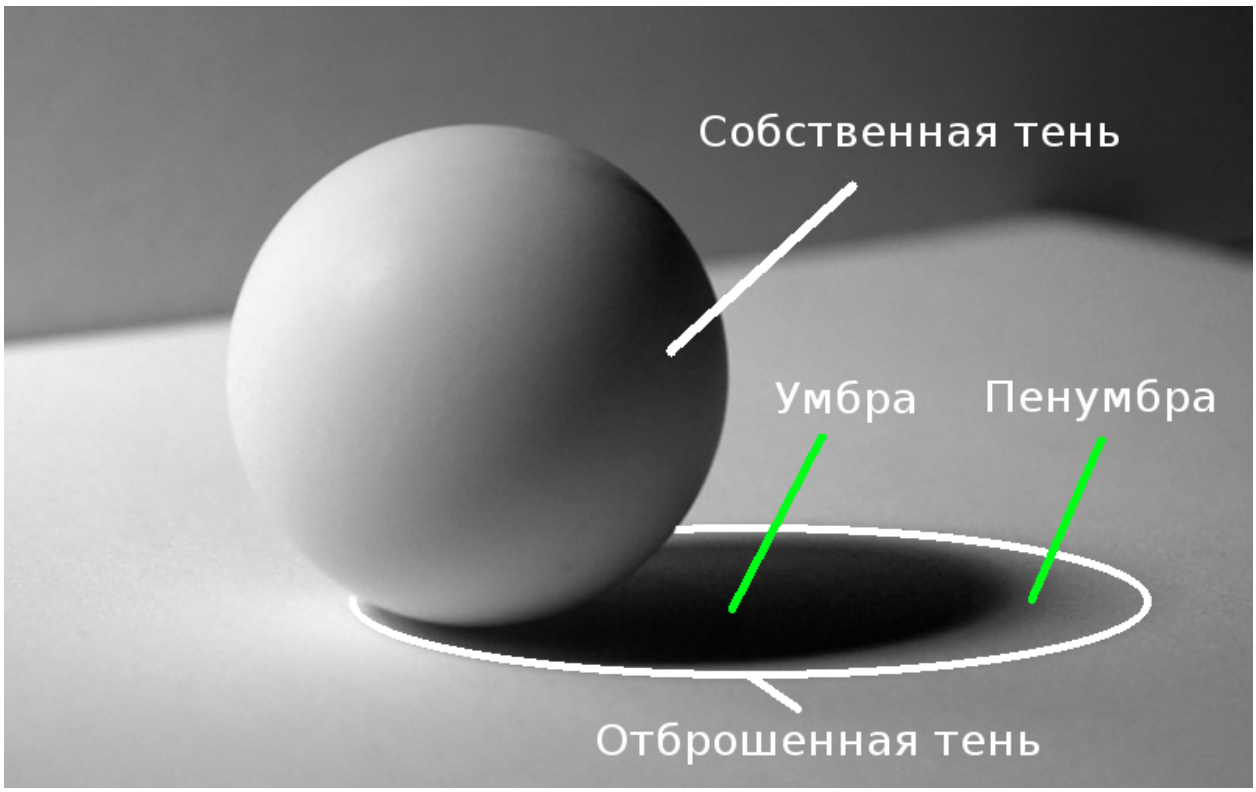

Рис. 2: Регионы тени

Для представления отображаемых цветов используется следующая модель [4]:

$$I_i = S_i \circ R_i \tag{1}$$

где i - индекс пикселя I_i - итоговая RGB интенсивность, S_i - множитель затененности материала, R_i - собственный цвет объекта (альбедо).

На полностью освещенной области виден собственный цвет материала, на области, куда свет не падает совсем, интенсивность равна нулю. Таким образом, множитель S варьируется в диапазоне $[0, 1]$. Оценка множителя S является основной задачей при удалении теней, так как зная его, можно получить искомую карту альбедо поэлементным делением на S исходного атласа с тенями.

2. Обзор существующих подходов

Задача удаления теней с моделей не нова, есть некоторое количество практических и теоретических подходов, ориентированных на эту задачу, как полностью автоматических, так и с пользовательским вводом. Среди них есть методы как для 3D моделей, так и для фотографий. Для решения поставленной задачи интересны оба этих направления, так как промежуточные шаги и идеи могут быть универсальны.

2.1. Практические подходы

2.1.1. 3D

Unity De-Lighting Tool [22], Agisoft Photoscan "remove lighting" feature[3], полностью автоматические решения, работающие с 3D моделями.

Рис. 3: Пример выравнивания освещенности в Agisoft Photoscan. Слева: исходная модель, справа - обработанная

Эти подходы ориентированы на удаление собственных теней и не поддерживают отброшенные тени. Это обусловлено тем, что в контексте моделей с известной геометрией удаление собственных теней представляет из себя более простую задачу, чем удаление отброшенных. Рассмотрим диффузную модель освещения Ламберта:

$$S = (\vec{n}, \vec{d})L \tag{2}$$

, где $\vec{n}$ - нормаль к поверхности в точке, $\vec{d}$ - направление на источник света, L - цвет

световых лучей. Если источник света достаточно далеко, то вектор направления на него для точек со схожей нормалью можно считать постоянным, то есть, множитель S для точек с похожей нормалью есть константа. Если сделать предположение, что модель в целом однотонна, что часто верно для типичных для фотограмметрии моделей (например, скал или статуй), то множитель S можно оценить как

$$S = I_n / R_a \tag{3}$$

, где I_n - средний цвет модели по всем похожим нормалям, R_a - средний тон модели без теней, который можно оценить как 70 персентиль цветов по яркости.

Для отброшенных теней подобный подход не работает, потому что нельзя сделать предположение о константности множителя S для точек с похожей нормалью. Так же, могут возникнуть трудности, если на модели нет доминирующего цвета, или если есть материал, уникальный по направлению (например, собранный кубик рубика. В этом случае все стороны приведутся к среднему цвету и потеряется информация о цвете).

Unreal Engine: Imperfection for perfection, part 2 [23]. Подход, основанный на сохранении дополнительной информации в момент съемки фотографий и воссоздании условий освещения в программе для 3D рендеринга. Для этого используется две сферы: рассеивающая, с известным альбедо, для определения интенсивности, и отражающая, для более точного позиционирования источника.

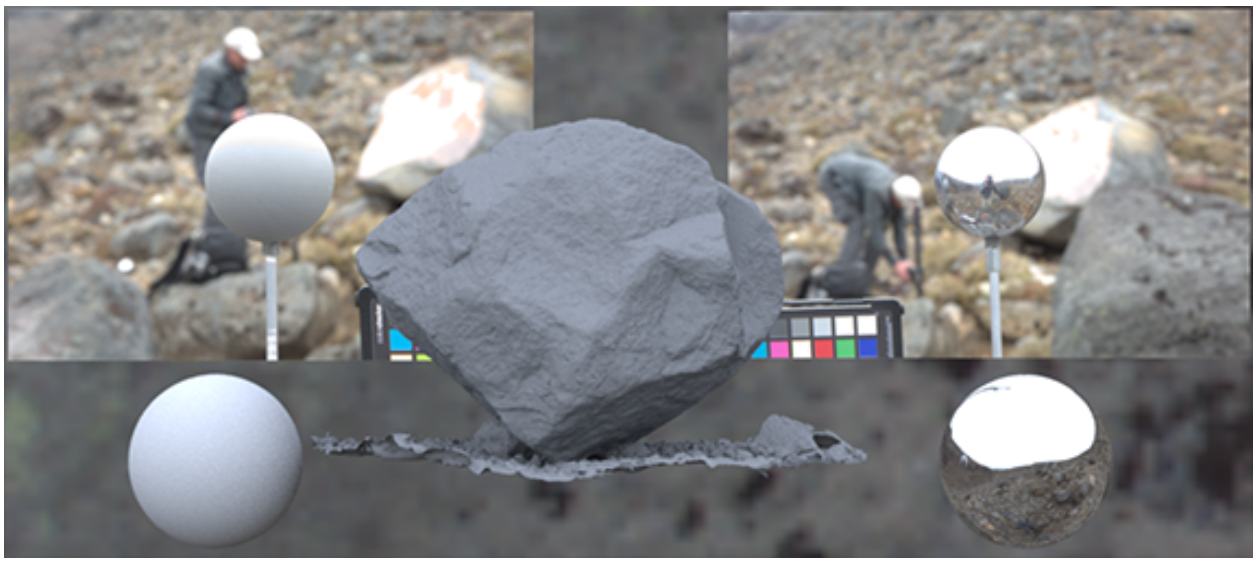

Рис. 4: Рассеивающая и отражающая сферы

Данный подход не использует предположения об однородности цвета объекта и ввиду этого более широко применим, нежели [22], [3]. Однако, он неприменим к уже отснятым моделям, и требует дополнительные действия и аппаратуру, что не всегда

доступно.

2.1.2. 2D

На практике с фотографий тени удаляются вручную с помощью редакторов изображений общего назначения таких как Photoshop. Кроме того существовала специализированная программа Lightbrush [20], позволяющая удалить тени в полуавтоматическом режиме, но она недоступна с 2012 года. Так же существует программа с закрытым исходным кодом Inpaint[21], по-видимому, работающая по принципу замены частей текстуры в затененной области на части из освещенной (см. [5], [14]). Этот подход позволяет обработать регионы с регулярной текстурой, но не подходит для случаев, когда в тени находится уникальный объект.

2.2. Теоретические подходы

Подходы по удалению отброшенных теней чаще всего делятся на две стадии: выделение тени и ее удаление. В какой-то мере эти стадии независимы, но алгоритмы удаления имеют разные требования к качеству получаемой маски, более устойчивы к одним типам проблем и менее устойчивы к другим.

2.2.1. 3D

Одним из методов вляется метод [13]. Это метод для удаления теней с трехмерных моделей в условиях солнечного освещения. Выделение тени делается с помощью проекции тени на модель. Модель отрисовывается в ортографической проекции по направлению солнечных лучей, и все точки модели, что оказываются заслонены геометрией, помечаются как тень. Пенумбра оценивается полосой заданной ширины (глобальный параметр алгоритма) и отбрасывается, оставшаяся часть принимается за область умбры. По границе умбры ищется множитель S как отношение самой светлой части к самой темной части в локальной окрестности. Далее, множитель гладко интерполируется внутрь умбры. Текстура в области пенумбры генерируется pull-push[10] алгоритмом, что обеспечиает гладкий переход, но не сохраняет исходную текстуру в этой области. Вторым недостатком этого подхода помимо потери текстуры в пенумбре является нечувствительность к сложным теням. Это происходит из-за неточностей геометрии модели и из-за погрешности в оценке направления солнечных лучей.

Рис. 5: Пример потери текстуры в области пенумбры

2.2.2. 2D

Область удаления теней с двухмерных изображений исследована очень обширно, существует большое количество разнообразных методов. При удалении тени с фотографии разница в удалении отброшенных и собственных теней исчезает из-за отсутствия данных о геометрии, поэтому методы не различаются по этому признаку.

Одним из методов является Guo[15]. Это полностью автоматический подход для удаления теней с двухмерных изображений. Выделение тени осуществляется в два этапа: на первом этапе производится сегментация изображения, на втором шаге - классификация сегментов по признаку тень/не тень с дополнительным поиском пар сегментов с одинаковым материалом, но разными классами затененности.

Удаление тени осуществляется с помощью найденных пар сегментов. Имея такие пары, оценивается требуемый множитель S : (формула для оценки множителя упрощена для простоты изложения по сравнению с формулой в источнике, отброшен учет плавного перехода в пенумбре)

$$I_{shadow} = S \circ R$$

$$I_{light} = 1 \circ R$$

$$S = \frac{I_{shadow}}{I_{light}}$$

Гладкость переходов в пенумбре обеспечивается с помощью алгоритма, решающего задачу мэттинга[1]: разделения изображения на передний и задний план с плавным переходом.

$$I = \alpha L + (1 - \alpha)S, \quad \alpha \in [0, 1] \tag{4}$$

, где L - передний план, S - задний план. В нашем случае, передний план - область изображения на свету, задний - в тени. Основным недостатком такого метода выделения тени является нечувствительность классификатора к сегментам, одинаково выглядящим, но имеющим разную освещенность.

Рис. 6: Пример ошибки классификатора.

Основным недостатком такого метода удаления тени является то, что оценка константным множителем работает плохо для неоднородных теней.

Еще одним методом является [12]. Это метод для удаления теней с изображений. Выделение тени производится с помощью мэттинга [1]: на вход пользователь передает изображение со штрихами в областях тени и света. Главным недостатком такого метода выделения тени является то, что требуемая сложность пользовательского ввода сильно коррелирует с количеством замкнутых областей тени и света, что затрудняет работу со сценами с большим количеством мелких единообразных регионов тени.

Удаление тени производится с помощью подразбиения всей области изображения на перекрывающиеся патчи, и для каждого затемненного патча находится ближайший сосед по текстуре из осветленных патчей. После этого пользуются подходом локального переноса цвета [24] для переноса тона светлого патча на затененный патч с сохранением текстуры:

$$I'_k = \frac{\sigma(L)}{\sigma(S)}(I_k - \mu(S)) + \mu(L) \tag{5}$$

, где $\sigma(L), \sigma(S)$ - стандартное отклонение в освещенном и затененном патчах соответственно, $\mu(L), \mu(S)$ - среднее значение в освещенном и затененном патчах.

Описанный подход поддерживает произвольные неоднородности интенсивности тени, что делает его весьма мощным методом, но поиск ближайших соседей по текстуре плохо работает на примерах с похожей текстурированностью, но разным тоном материала.

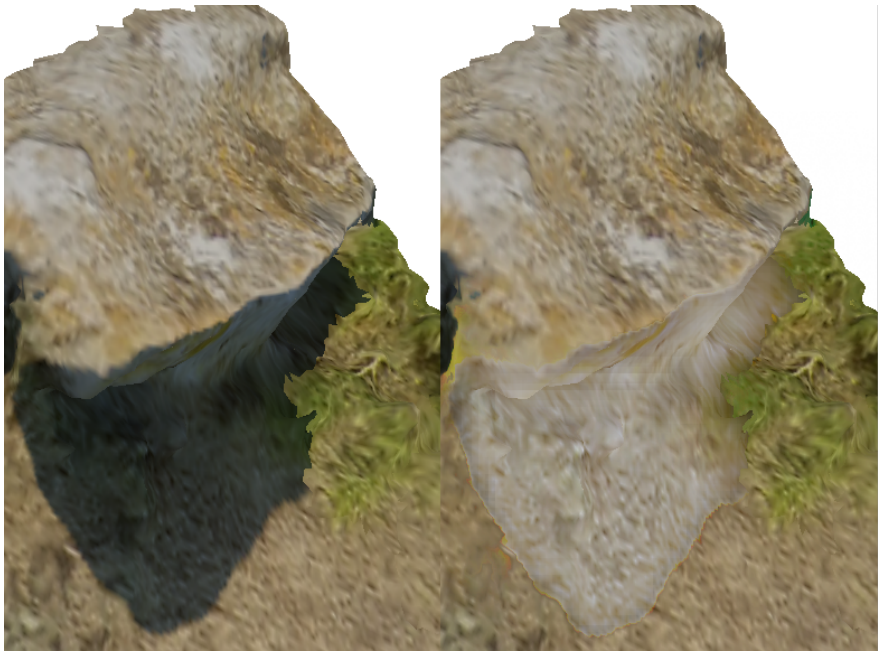

Рис. 7: Плохая работа алгоритма [12] на материалах со слабо различающейся текстурированностью

Еще одним методом является [11]. Это метод для удаления теней с фотографий использующий пользовательский ввод на стадии выделения тени.

Способ выделения тени основывается на пользовательском вводе, схожем с [12]. После этого запускается KNN для классификации каждого пикселя изображения по признаку тень/не тень, основываясь на размеченных пользователем пикселях. При этом, данный метод выделения тени требует меньшее количество пользовательского вода, чем [12], потому что достаточно выделить все уникальные экземпляры материалов в тени и на свету, и нет зависимости от количества регионов тени с одинаковым материалом.

Удаление: В пенумбре семплируются полоски, ортогональные границе тени, вдоль полосок находится множитель S. Далее множитель интерполируется в пределах пенумбры между полосками с сохранением текстуры, и экстраполируется в область умбры.

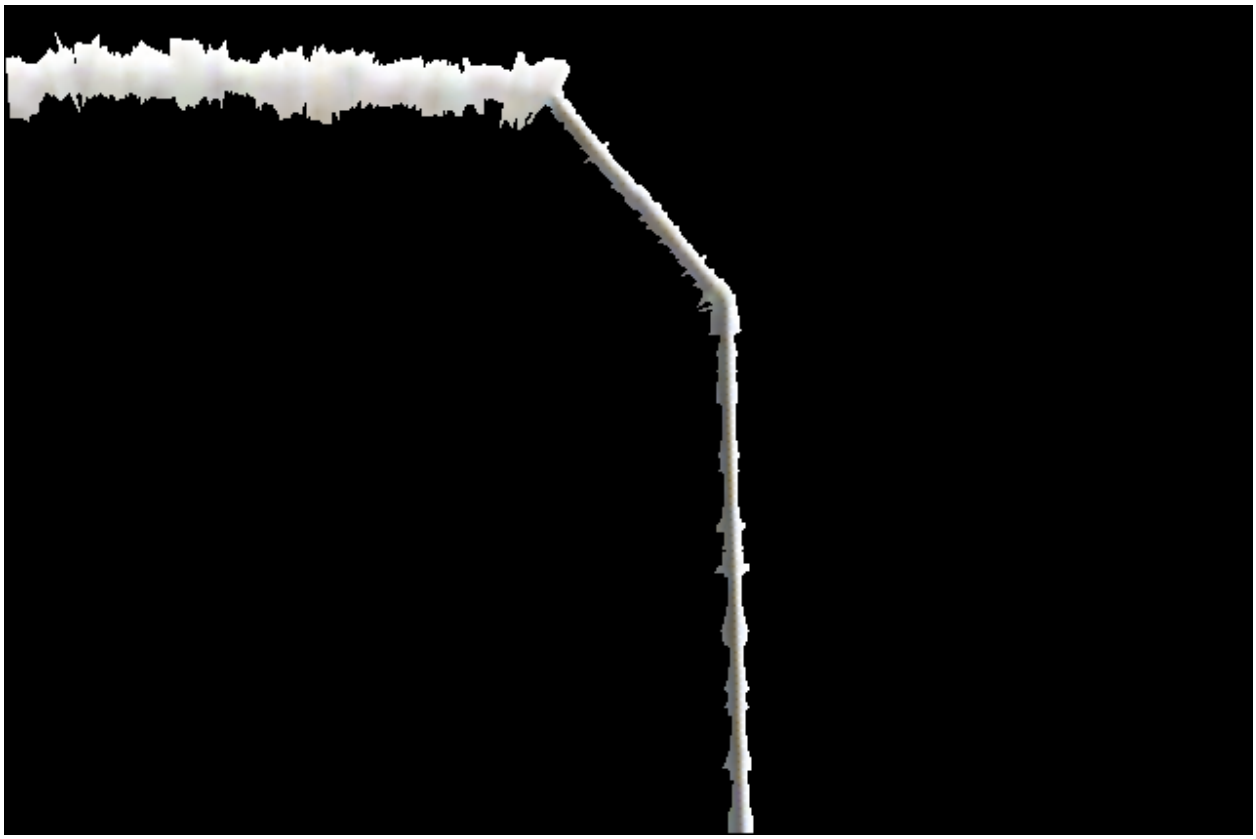

Рис. 8: Найденный в области пенумбры множитель S для изображения на рис. 7

Рис. 9: Экстраполированный в область умбры множитель

Главным недостатком данного подхода являются артефакты, связанные с сильной неоднородностью тени внутри умбры, не достигающей пенумбры.

Еще одна проблема состоит в том, что предположение, положенное в основу оценки множителя в пенумбре, а именно, что на границе тени не происходит смены материала, часто нарушается, когда один объект заслоняет другой.

Рис. 10: Пример провала предположения о константности материала на границе тени. Соседние к границе тени регионы имеют разный материал, что привело к порче текстуры в пограничном регионе

2.3. Сравнение и выводы

Из-за совокупности численных результатов и теоретических преимуществ, приведенных ниже, алгоритм [11] был выбран как базовый для 3D моделей.

Преимущества метода выделения тени, используемом в [11]:

- KNN показывает себя предсказуемым и устойчивым методом (см. рис. 6, 11)
- При сравнимом с мэттингом [12] качестве получаемой маски для ее получения нужен меньший пользовательский ввод, зависящий только от количества материалов в сцене, но не от количества регионов тени [11]

Рис. 11: Пример работы алгоритма [11]. Слева: оригинальное изображение с наложенным пользовательским вводом. Справа: обработанное изображение. Благодаря явной разметке, темные клетки доски остались нетронутыми, в отличие от результата, полученного с помощью метода [15]

Преимущества метода удаления тени, используемом в [11]

- Гладкая экстраполяция множителя внутрь умбры, что похоже на подход [13], обеспечивает поддержку неоднородных вблизи границы теней.
- Сохранение текстуры в области пенумбры (в отличие от [13])
- Проблема, присущая фотографиям и связанная с порчей текстуры в пограничном регионе (см. рис. 10), менее присуща трехмерным моделям, так как большое количество проблем при работе с фотографиями возникает из-за перспективы и перекрытия осветленных объектов затененными и наоборот, чего не наблюдается на 3D моделях, так как там соседние участки на текстуре соседние и в пространстве. Это не избавляет от проблемы целиком (например, кубик рубика, затененный с одной стороны), но сильно сужает множество проблемных случаев.
- Нет особых требований к подаваемой на вход текстуре, будь то отсутствие постобработки и сжатия, глубина цвета или характеристики снимающей камеры (который свойственны методам, использующим знания о физических свойствах системы, например, [8])

Приведем результаты количественного сравнения вышеперечисленных методов на датасете ([15], [11]). Это датасет из 214 изображений с ground truth для сравнения методов удаления теней.

Используемая метрика [11] основана на метрике Root Mean Square Error

$$RMSE(A, B) = \sqrt{E((A - B)^2)} \tag{6}$$

Применять $RMSE$ напрямую для оценки качества не очень показательно, так как процент площади тени варьируется от фотографии к фотографии, и в случае малого этого процента ошибка будет мала, даже если алгоритм ничего не сделает и вернет исходное изображение. Считать ошибку только для пикселей в тени тоже не очень правильно, потому что интересно также учесть то, насколько алгоритм затронул области, не являющиеся тенью. Для решения вышеупомянутых проблем в [12] используется следующая метрика:

$$RMSE_{fair} = \frac{RMSE(delit, gt)}{RMSE(orig, gt)} \tag{7}$$

, где gt - ground truth, $orig$ - исходное изображение с тенями, $delit$ - обработанное изображение.

Эта метрика позволяет судить, улучшился или ухудшился результат, по тому, меньше ее значение единицы или больше.

Группа	Все пиксели			Только в тени		
	Guo[15]	Zhang[12]	Gong[11]	Guo[15]	Zhang[12]	Gong[11]
текстура	0.61 (0.73)	0.44 (0.27)	**0.34 (0.22)**	0.51 (0.92)	0.27 (0.38)	**0.19 (0.21)**
мягкость	0.77 (0.73)	0.48 (0.31)	**0.39 (0.18)**	0.68 (0.84)	0.33 (0.43)	**0.22 (0.15)**
рваность	0.81 (0.78)	0.63 (0.31)	**0.41 (0.19)**	0.76 (1.08)	0.46 (0.46)	**0.26 (0.17)**
цветность	1.12 (1.31)	0.58 (0.41)	**0.43 (0.20)**	1.09 (1.73)	0.50 (0.77)	**0.27 (0.23)**

Таблица 1: Сравнение алгоритмов, данные взяты из [11]. Данные отображаются в формате: *среднее значение метрики(стандартное отклонение)*. Подходы сравниваются сначала по среднему значению, потом по стандартному отклонению. Жирным шрифтом выделены лучшие значения по каждой категории. Датасет поделен на 4 группы, по наибольшей степени принадлежности одной из четырех категорий: выраженная текстура материала, размер пенумбры, сильная вариация формы тени или ширины пенумбры, цвет тени. Тень может быть цветной если отбрасывается сквозь полупрозрачный предмет, например, пластиковую бутылку

3. Предлагаемый подход

3.1. Предлагаемый алгоритм

1. Препроцессинг.

Грубая разметка пользователем затененных и освещенных областей модели. Вместе с этим, генерируются:

- Карта чартов, представляющая собой бинарное изображение с единицами в тех пикселях, где есть информация о текстуре материала и нулями в промежутках между чартами
- Карта позиций, где вместо цвета изображения сохраняются координаты точек в трехмерном пространстве

2. Получение карты тени, нахождение множителя внутри полосок.

Аналогично методу [11], текстурный атлас классифицируется по признаку тень/не тень с помощью KNN. Ортогонально границе семплируются полоски пикселей, формирующие пенумбру, и находится множитель. На этом шаге к алгоритму распространения полоски добавляется условие остановки при достижении границы чарта, для чего используется карта чартов, полученная на стадии препроцессинга. При этом делается предположение, что пенумбра редко пересекается с границами чартов. Это предположение верно для алгоритмов раскладки текстуры по атласу, применяющихся в Agisoft Photoscan[3]. Но некоторые алгоритмы, например в Reality Capture[16], склонны к учету текстуры и созданию большого количества мелких чартов, без сохранения пропорций оригинальной модели где сделанное предположение начинает плохо работать. Это более сложный случай, которые не поддержан в рамках данной работы.

3. Распространение множителя

На этом шаге нужна принципиальная модификация алгоритма относительно подхода для изображений. Дело в том, что все предыдущие шаги локальны и выполняются в пределах каждого чарта независимо. Но при этом, область умбры зачастую весьма обширна и покрывет более одного чарта, что делает невозможной почартовую экстраполяцию множителя. Для поддержки этого случая нужна модификация алгоритма экстраполяции. Она описана подробно в подразделе (3.2).

4. Постобработка

Оригинальный алгоритм обеспечивает гладкость на границе тени из-за высокой локальности при обработке пенумбры. Экстраполяция множителя хорошо работает для теней без сильной неоднородости в области умбры. Однако, она дает неверные результаты, когда тень сильно неоднородна в области умбры, либо отражающая способность материала сильно меняется. Для решения этой проблемы применяется по-

стобработка с применением ручного ввода, которая подробно описана в подразделе (3.3).

3.2. Гладкость множителя внутри умбры

3.2.1. Экстраполяция в связном случае

В алгоритме [11] экстраполяция множителя в области умбры проводится с помощью метода [6]. Его идея состоит в том, связать каждый эстраполируемый пиксель с четырьмя своими соседями, для обеспечения гладкости. Это формулируется в виде системы уравнений:

$$\begin{cases} A(i_1, j_1) - A(i_1 - 1, j_1) = 0 \\ A(i_1, j_1) - A(i_1 + 1, j_1) = 0 \\ A(i_1, j_1) - A(i_1, j_1 - 1) = 0 \\ A(i_1, j_1) - A(i_1, j_1 + 1) = 0 \\ \dots \\ A(i_k, j_k) - A(i_k, j_k + 1) = 0 \end{cases} \tag{8}$$

, где A - изображение, k - количество неизвестных пикселей для экстраполяции.

В матричном виде данная система записывается следующим образом:э

$$Bx = 0 \tag{9}$$

, где B - матрица размера (количество связей) на (количество пикселей в экстраполируемом изображении). На k-ой строке матрицы B содержится ровно два ненулевых элемента: единица и минус единица в столбцах, соответствующих пикселям в связи. Часть связей проходит между неизвестным и известным пикселем. В связи с этим систему можно переписать в виде

$$B_{known}x_{known} + B_{unknown}x_{unknown} = 0 \tag{10}$$

$$B_{known}x_{known} = -B_{unknown}x_{unknown} \tag{11}$$

, где B_{known} - столбцы матрицы B, соответствующие известным пикселям, $B_{unknown}$ - столцы, соответствующие неизвестным.

Матрица B имеет большие размеры (порядка 10^6 x 10^6), и при этом очень разрежена, так как имеет всего по два ненулевых элемента в каждой строке. Для решения таких систем существуют эффективные способы, такие как [19], [18]

3.2.2. Экстраполяция в несвязном случае

Данный способ естественным образом обобщается на случай чартов - нужно исключить из системы связи с пикселями, находящимися вне чартов, и добавить связи между пикселями, находящимися на границах чартов и являющимися соседями на модели.

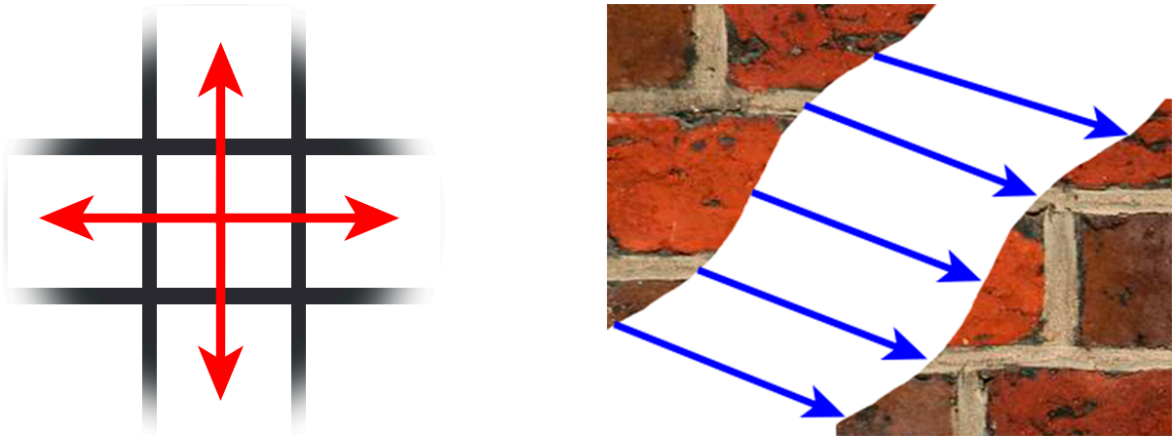

Рис. 12: Слева: связи между пикселем и соседями. Справа: связи между граничными точками чартов

При нахождении телепортов необходимо позаботиться о двух деталях:

- Телепорт может находиться на том же чарте. Такой случай возможен, если, например, обрабатывается модель конусовидного стаканчика, который решили разрезать от центра к краю, чтобы уложить в один чарт и при этом сохранить пропорции
- Телепорта может не быть. Это ситуация, когда у поверхности, образующей модель, есть край

Предлагается следующее решение:

1. Для точки, для которой ищется телепорт, находится локальная окрестность: множество точек, принадлежащих тому же чарту, и путь по чарту с учетом разрезов до которых меньше наперед заданного радиуса
2. Среди всех точек границ чартов находится ближайшая точка, которая не принадлежит локальной окрестности и лежит в трехмерном пространстве ближе, чем самая далекая точка из окрестности

Локальная окрестность вводится из-за того, что ввиду численных погрешностей телепорт может не оказаться ближайшей точкой кроме ближайших трех соседей на своем чарте.

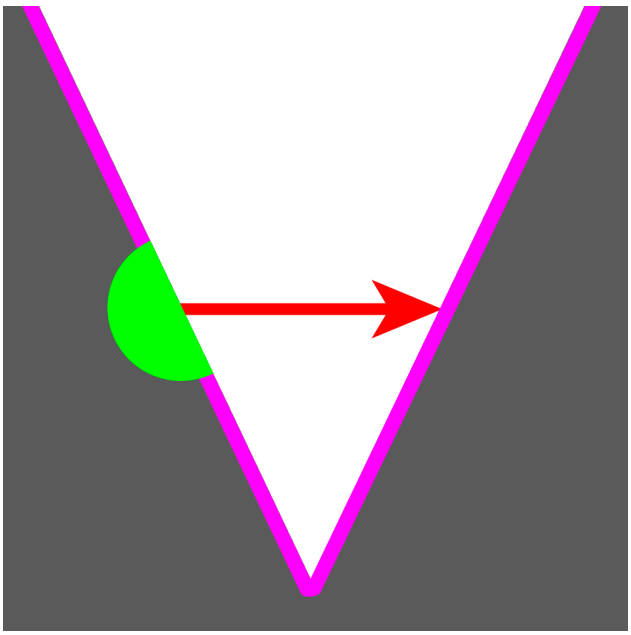

Рис. 13: Серым цветом отмечен чарт, розовым - граница разреза чарта, зеленым - локальная окрестность точки, для которой ищется телепорт, стрелкой указан найденный телепорт

Algorithm 1: Нахождение карты телепортов

Вход : Карта чартов C, карта позиций P, радиус локальной окрестности R в пикселях, количество ближайших соседей для поиска кандидатов K

Выход: Карта T где в каждом пикселе записан индекс телепорта в линеаризованном изображении, либо -1, если это не граничный пиксель, или сосед не найден из-за того, что граница чарта совпадает с границей модели

$T \leftarrow MatrixFilledWith(-1, size(C))$
$contours \leftarrow FindContours(C)$
$knnIndex \leftarrow BuildKDTree(C, P)$
foreach $point \in contours$ **do**
    $chartLabel \leftarrow C(point.x, point.y)$
    $locals \leftarrow FindLocalNeighborhood(point, chartLabel, C, R)$
    $neighbors \leftarrow FindNeighbors(knnIndex, point, K)$
    $minDist \leftarrow GetDistToFarthest(locals, point, P)$
    for $neighbor \in neighbors$ **do**
        if $neighbor \notin locals$ ***and*** $dist(point, neighbor) < minDist$ **then**
            $T(point.x, point.y) \leftarrow PointToIndex(neighbor)$
            $minDist \leftarrow dist(point, neighbor)$
        end
    end
end
return T

Время работы: Инициализация и поиск контуров линейны по размеру карты C, построение индекса: $O(N \log(N))$, где N - суммарное количество точек в контурах, вызов функции $FindLocalNeighborhood$ представляет из себя поиск в ширину, линейный по размеру ответа: $O(NR^2)$, поиск ближайших соседей: $O(kN \log(N))$. K, R - константы, итоговое время работы: $O(C.width * C.height + N \log(N))$

Algorithm 2: FindLocalNeighborhood Поиск локальной окрестности

Вход : Пиксель P, индекс чарта $CIND$, карта чартов C, радиус в пикселях R

Выход: Множество пикселей - локальная область L

```
1  L ← ∅
2  Q ← GetEmptyQueue()
3  Q.push(P)
4  while not Q.empty() do
5  |  p ← Q.pop()
6  |  L.insert(p)
7  |  for shift ∈ {left, right, up, down} do
8  |  |  shifted ← p + shift
9  |  |  if C(shifted.x, shifted.y) == CIND and dist(P, shifted) < R and
   |  |  shifted ∉ L then
10 |  |  |  Q.push(shifted);
11 |  |  end
12 |  end
13 end
14 return L
```

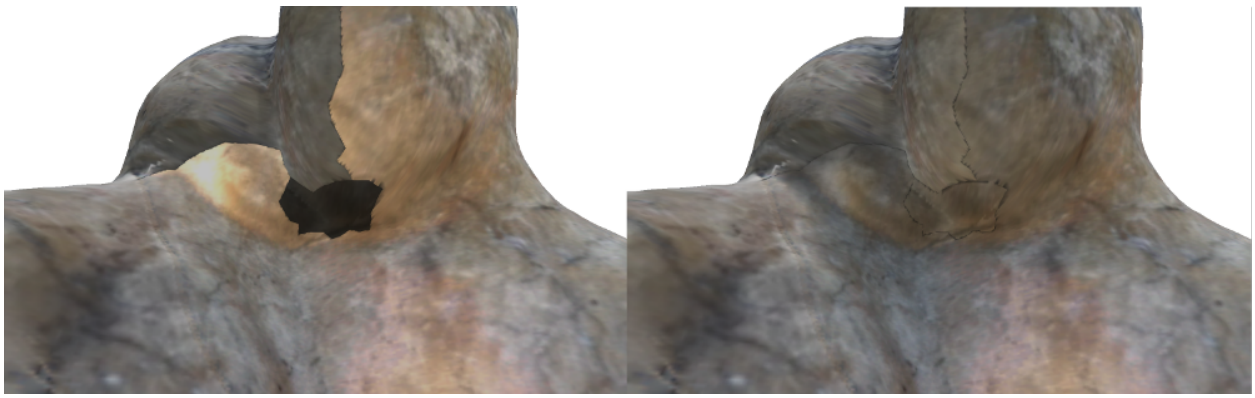

Рис. 14: До и после экстраполяции между чартами

3.3. Постпроцессинг

3.3.1. Недостаток базового алгоритма

Оригинальный алгоритм обеспечивает гладкость на границе тени благодаря целенаправленной обработке пенумбры. Однако, экстраполяция множителя внутрь умбры не учитывает изменения интенсивности тени или смену материала, что не создает проблем в случае, близком к линейному изменению интенсивности тени относительно ее границ, и в случае, когда материалы имеют близкую отражающую способность. Однако, в случае сильной неравномерности тени внутри умбры, экстраполяции может быть недостаточно.

Рис. 15: Пример, когда экстраполяции недостаточно для полного удаления тени. Красным цветом обведена проблемная область, возникшая из-за резкого перехода с земли на стену, где тень слабее

3.3.2. Предлагаемое решение

Среди рассмотренных алгоритмов есть те, что поддерживают неоднородности тени в умбре ([24], [7], [12]). Однако, результат их работы зависит от сложности формы тени и параметров материала, таких как текстурированность и однородность. Для практического применения разрабатываемого подхода важно максимально достижимое качество, поэтому было решено использовать пользовательский ввод для коррекции результата, полученного на первой стадии алгоритма (подраздел 3.1).

Пользовательский ввод можно было бы реализовать в виде разбиения изображения на патчи (как в методе [12]) и ручного выделения пар патчей донора - акцептора, но возникает две сложности:

- Тень может быть высокодетализированной и сложной, например, тень от листвы. В таком случае возникает трудность определения размера патча. Если выбрать слишком большой размер, то вместе с областью тени захватится часть

освещенной области, что приведет к артефактам. Если выбрать слишком маленький, то потеряются детали текстуры

- Большое количество ручной работы, которое тем больше, чем меньший размер патчей мы рассматриваем

Авторы [2] предлагают метод для интерактивного глобального переноса цвета между изображениями. Для этого выделяется две области: область - донор на изображении, откуда переносится цвет, и область на целевом изображении, содержащая пример материала, который нужно скорректировать. На целевом изображении считается маска похожести на средний цвет выделенного региона. Затем, к целевому изображению применяется преобразование переноса цвета 5, пропорционально маске похожести.

Маска похожести рассчитывается по следующей формуле:

$$f_k = F(\rho(\mu(T), C_k) \tag{12}$$

, где f_k - вес маски похожести в пикселе k, C_k - цвет пикселя на целевом изображении, $\mu(T)$ - средний цвет в отмеченном целевом регионе, ρ - евклидово расстояние в цветовом пространстве LAB, F - функция отображающая расстоние между цветами в похожесть между ними и обладающая свойствами $F(0) = 0, \lim_{x\to\inf} F(x) \to 0$. В оригинальной статье авторы используют $F(x) = e^{-3x}$.

Скорректированное изображение рассчитывается по следующей формуле:

$$C'_k = C_k(1 - f_k) + C_k^{new} f_k \tag{13}$$

, где C_k^{new} - значение пикселя, полученное по формуле переноса цвета (5)

Данный подход позволяет обработать все изображение, выделив всего два региона. Его можно применить и к задаче удаления теней, где целевым материалом будет часть затененной области с похожей интенсивностью тени.

Применять преобразование глобально в случае трехмерных моделей достаточно сложно, так как часто все ее части недоступны взгляду одновременно, и после каждого этапа выделения примеров и применения преобразования приходится проверять всю модель, что весьма трудоемко. В итоге был выбран компромиссный подход: преобразование проводится внутри радиуса, задаваемого пользователем, и внутри этого радиуса считается маска похожести цвета на целевой цвет, указанный пользователем.

3.3.3. Алгоритм

Пользовательский ввод реализован в виде клика по материалу - донору и штриха по материалу - акцептору, который требуется сравнять с материалом донора.

Алгоритм постпроцессинга итеративный:

1. Пользователь выбирает радиус региона, который будет использоваться для оценки среднего цвета
2. Кликает на материал - донор
3. Выбирает радиус области применения преобразования
4. Кликает на материал, который следует скорректировать, или делает по нему штрих, если радиуса области применения клика недостаточно
5. Если желаемый результат не достигнут, то переход к шагу 1

Рис. 16: Слева: модель после применения основной стадии алгоритма. Справа: модель после постобработки

4. Реализация

Предлагаемый подход реализован на C++ и компилируется в три исполняемых файла, работающих независимо. Каждый из файлов отвечает за одну из ступеней подхода: препроцессинг, основная часть, и постобработка. Основной алгоритм реализован с поддержкой как 3D моделей, так и фотографий для возможности сравнения с алгоритмами удаления теней с фотографий.

Для оптимальной производительности использованы следующие библиотеки:

- Flann для эффективного поиска ближайших соседей
- UMFPACK для эффективного решения разреженных систем
- CGAL для интерполяции в области пенумбры
- OpenMP для распараллеливания на CPU
- OpenGL для шагов препроцессинга и постобработки

5. Сравнение

5.1. Регрессионное сравнение

Для тестирования качества реализации алгоритма, он был запущен на датасете, используемом при сравнении алгоритмов для изображений (см. таблицу 1 в разделе 2.3). Запуск проведен без использования шага постобработки.

Приведем здесь эту таблицу еще раз вместе с результатами работы реализованного алгоритма:

Группа	Все пиксели			
	Guo[15]	Zhang[12]	Gong[11]	1 шаг
текстура	0.61 (0.73)	0.44 (0.27)	**0.34 (0.22)**	**0.37 (0.22)**
мягкость	0.77 (0.73)	0.48 (0.31)	**0.39 (0.18)**	**0.44 (0.18)**
рваность	0.81 (0.78)	0.63 (0.31)	**0.41 (0.19)**	0.47 (0.23)
цветность	1.12 (1.31)	0.58 (0.41)	**0.43 (0.20)**	0.46 (0.21)
Группа	Только в тени			
	Guo[15]	Zhang[12]	Gong[11]	1 шаг
текстура	0.51 (0.92)	0.27 (0.38)	**0.19 (0.21)**	0.24 (0.23)
мягкость	0.68 (0.84)	0.33 (0.43)	**0.22 (0.15)**	0.29 (0.19)
рваность	0.76 (1.08)	0.46 (0.46)	**0.26 (0.17)**	0.33 (0.26)
цветность	1.09 (1.73)	0.50 (0.77)	**0.27 (0.23)**	0.35 (0.25)

Таблица 2: Регрессионное сравнение подхода для 3D моделей с алгоритмами для фотографий. Реализованный подход озаглавлен "1 шаг". Данные отображаются в формате: *среднее значение метрики(стандартное отклонение)*. Подходы сравниваются сначала по среднему значению, потом по стандартному отклонению. Лучший подход выделен жирным шрифтом, в случае отсутствия лучшего подхода выделяется первая группа.

Предлагаемый метод превосходит все методы из сравниваемых, за исключением опорного алгоритма, которому проигрывает на некоторых категориях данных. Этот факт был посчитан приемлемым при условии существенного упрощения некоторых шагов оригинальной реализации, что упрощает разработку и поддержку кода, а также обеспечивает более предсказуемый результат.

5.2. Сравнение в 3D

5.2.1. Датасет

Для сравнения качества удаления теней с моделей необходимо иметь датасет с ground truth. При этом, получить такой датасет с помощью фотограмметрии представляется трудновыполнимой задачей так как нужно обеспечить полную идентичность внешних условий в незатененных областях. Это практически невозможно сделать для моделей в естественном освещении из-за переменчивости погодных условий. В помещении сделать такую модель тоже сложно из-за смены интенсивности рассеянного освещения при включении и выключении источника направленного освещения и из-за эффекта Ambient Occlusion, который проявляется даже при идеально равномерном мнешнем освещении. По этой причине для сравнения используются синтетически сгенерированные данные.

За основу было взято четыре модели, на текстуре которых нет теней. Часть примеров была получена путем отделения освещенных сегментов от более крупных моделей, часть представляет собой модели, отсканированные при подходящих погодных условиях с последующей ручной обработкой для получения карты альбедо. Для каждой модели в программе для 3D рендеринга были синтезированы карты освещения с различной жесткостью и формой отбрасываемой тени.

Из-за отсутствия доступных подходов для удаления отбрасываемых теней с трехмерных моделей, сравнение производится с методами обработки фотографий. Для этого сопоставляются скриншоты обработанных моделей трехмерным подходом, и обработанные двумерными подходами скриншоты затененных моделей. Получившийся датасет состоит из 21 изображения.

5.2.2. Количественная оценка

Все пиксели				
Guo[15]	Zhang[12]	Gong[11]	1 шаг	2 шаг
0.61 (0.20)	0.45 (0.11)	0.47 (0.20)	0.47 (0.18)	**0.34 (0.11)**
Только в тени				
Guo[15]	Zhang[12]	Gong[11]	1 шаг	2 шаг
0.54 (0.20)	0.45 (0.11)	0.47(0.20)	0.46(0.18)	**0.33(0.11)**

Таблица 3: Количественное сравнение алгоритмов на скриншотах трехмерных моделей. Подход состоящий только из базового алгоритма обозначен ”шаг 1”, подход с последующей постобработкой: ”шаг 2”

Предлагаемый подход, состоящий из двух шагов, выигрывает у остальных методов. При этом, алгоритм без пользовательского ввода проигрывает алгоритму Zhang[12], но при качественной оценке оказывается более приемлемым, так как лучше сохраняет текстуру объекта и восстанавливает гладкость на границе тени. Характерными артефактами, присущими базовому подходу, являются засветы и неудаленные тени в областях с неоднородным освещением внутри умбры, с чем хорошо справляется алгоритм постпроцессинга.

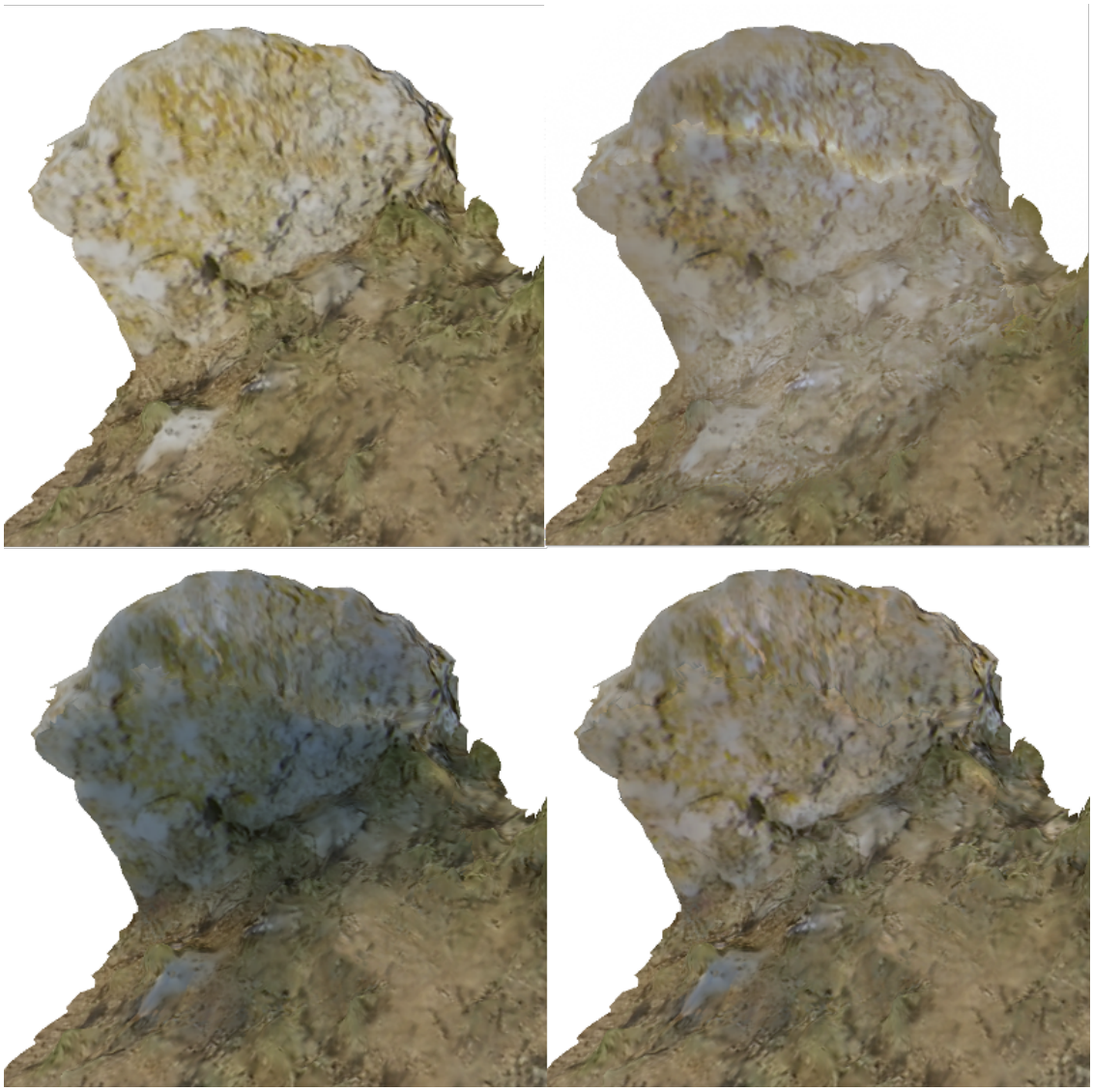

Рис. 17: Слева направо, сверху вниз: ground truth, результат работы Zhang[12], результат работы первого шага разрабатываемого алгоритма, результат второго шага. Несмотря на меньшую степень удаления тени, последний подход сохранил различия между материалами на земле, что позволило улучшить результат с помощью постобработки

5.2.3. Качественная оценка

Алгоритм был протестирован на качественном уровне с использованием моделей, находящихся в открытом доступе на ресурсе SketchFab[17] и в базе данных Agisoft.

Рис. 18: Пример работы: оригинальная модель. Автор: GlobalDigitalHeritage[9]

Рис. 19: Пример работы: модель после первого шага. Есть засветы и области, где тень не удалилась до конца

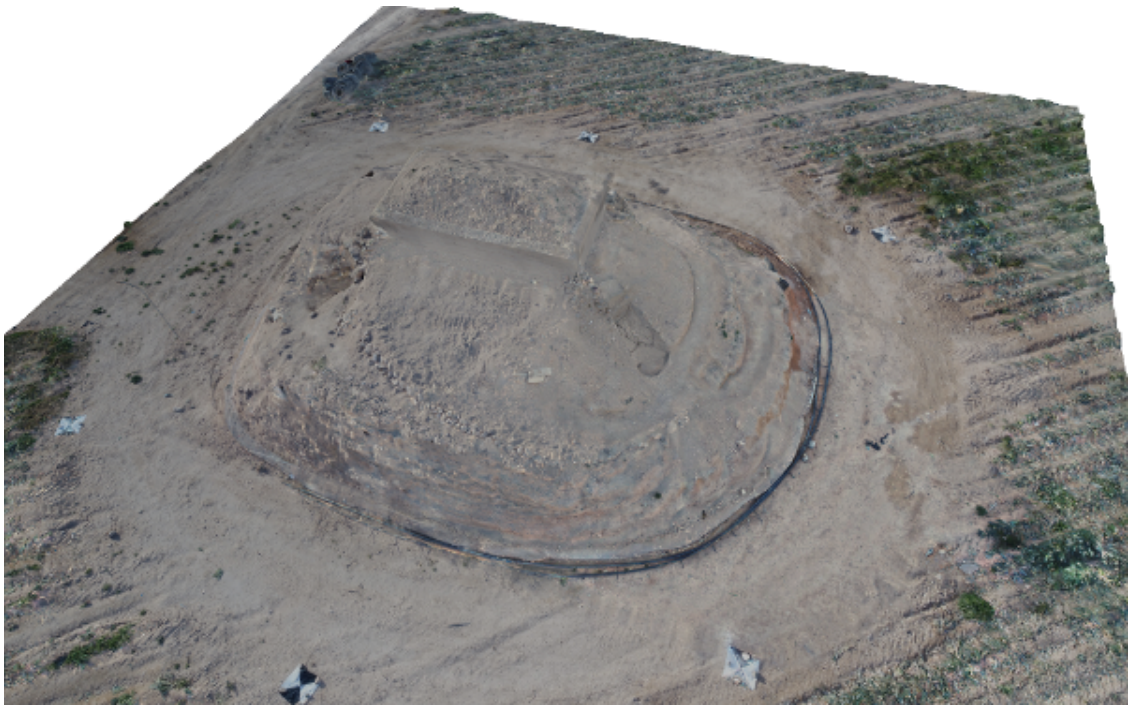

Рис. 20: Пример работы: модель после этапа постобработки

У предлагаемого алгоритма на есть принципиальное ограничение, приводящее к некорректной работе на моделях, где разбиение на чарты сильно коррелирует с текстурой или геометрией модели. В таком случае область пенумбры часто располагается близко к границе чарта, что не дает распространить множитель S в ее пределах. Данный недостаток является ожидаемым следствием сделанных в процессе реализации предположений и будет исправлен в дальнейшей работе.

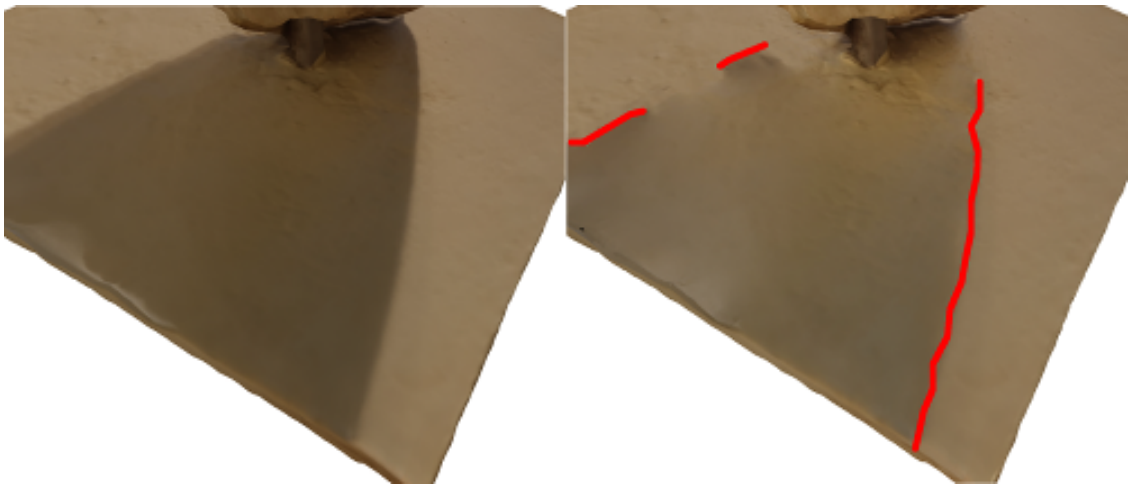

Рис. 21: Пример случая, не поддерживаемого алгоритмом. Красными линиями отмечены границы чартов, попавшие в область пенумбры.

5.3. Скорость работы

На тестовом датасете из 214 фотографий размером 640x480 пикселей реализованный метод отработал за 612 секунд, тогда как реализация Gong[11] за 2272 секунды. Обработка текстуры 2048x2048 как фотографии заняла 1 минуту против 25 минут у оригинальной реализации.

Заключение

Были проанализированы существующие подходы для выделения и удаления теней с фотографий и 3D моделей и на основе них создан алгоритм, работающий с трехмерными моделями. Для обеспечения устойчивого качества получаемых результатов, был разработан полуавтоматический метод для доводки результатов основного алгоритма в сложных случаях обширных неоднородных теней. Итоговый алгоритм был протестирован на синтетическом датасете и показал лучшие результаты среди сравниваемых методов при высокой скорости работы относительно опорной реализации.

Список литературы

[1] A. Levin D. Lischinski Y Weiss. A Closed Form Solution to Natural Image Matting // IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 30, Issue: 2). — 2008.

[2] A. Maslennikova V. Vezhnevets. Interactive Local Color Transfer Between Images. — 2007.

[3] Agisoft Photoscan. — URL: http://www.agisoft.com/.

[4] Barrow H., Tenenbaum J. Recovering intrinsic scene characteristics // Computer Vision System, A Hanson, E. Riseman (Eds.) стр. 3–26. — 1978.

[5] Criminisi Antonio, Pérez Patrick, Toyama Kentaro. Object Removal by Exemplar-Based Inpainting // CVPR. — 2003.

[6] D'Errico J. — URL: https://www.mathworks.com/matlabcentral/fileexchange/4551-inpaint-nans.

[7] E. Arbel H. Hel-Or. Shadow Removal Using Intensity Surfaces and Texture Anchor Points. — 2010.

[8] G.D. Finlayson S.D. Hordley Cheng Lu. On the removal of shadows from images // IEEE Transactions on Pattern Analysis and Machine Intelligence, том 28. — 2006.

[9] GlobalDigitalHeritage. — URL: https://sketchfab.com/GlobalDigitalHeritage.

[10] Gortler S. J. Grzeszczuk R. Szeliski R., Cohen M. F. // The Lumigraph. In SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, стр. 43–54. — 1996.

[11] H. Gong D. Cosker. Interactive Removal and Ground Truth for Difficult Shadow Scenes // JOSA A 33(9), стр. 1798–1811. — 2016.

[12] L. Zhang Q. Zhang C. Xiao. Shadow Remover: Image Shadow Removal Based on Illumination Recovering Optimization // IEEE Transactions on Image Processing. — 2015.

[13] M. Dellepiane L. Benedetti R. Scopigno. Removing shadows for color projection using sun position estimation // Visual Computing Laboratory, ISTI-CNR. — 2010.

[14] Ms.PatilV. A. Region Filling and Object Removal by Exemplar-Based Image Inpainting. — 2012.

[15] R. Guo Q. Dai D. Hoiem. Single-Image Shadow Detection and Removal using Paired Regions // Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference. — 2011.

[16] Relity Capture. — URL: https://www.capturingreality.com/.

[17] SketchFab. — URL: https://sketchfab.com.

[18] SuiteSparse. — URL: http://faculty.cse.tamu.edu/davis/suitesparse.html.

[19] SuperLU. — URL: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/.

[20] Tandent LightBrush. — URL: https://www.fxguide.com/featured/tandent-lightbrush-engineering-an-image/.

[21] Teorex Inpaint. — URL: https://www.theinpaint.com/inpaint-how-to-remove-shadow-from-photo.html.

[22] Unity De-Lighting Tool. — 2017. — URL: https://github.com/Unity-Technologies/DeLightingTool.

[23] Unreal Engine: Imperfection for perfection, part II. — 2015. — URL: https://www.unrealengine.com/en-US/blog/imperfection-for-perfection-part-2.

[24] Y. Shor D. Lischinski. The Shadow Meets the Mask: Pyramid-Based Shadow Removal // Comput. Graph. Forum. — 2008.