ФАКУЛЬТЕТ
ВЫЧИСЛИТЕЛЬНОЙ
МАТЕМАТИКИ И
КИБЕРНЕТИКИ
МГУ ИМЕНИ
М.В. ЛОМОНОСОВА

teach-in
ЛЕКЦИИ УЧЕНЫХ МГУ

КОМПЬЮТЕРНАЯ ГРАФИКА

ФРОЛОВ
ВЛАДИМИР АЛЕКСАНДРОВИЧ

ВМК МГУ

Оглавление

Лекция 1. Основы обработки изображений.

Свет и цвет

Сфера компьютерной графики состоит из трех областей:

1) Синтез изображений (компьютерная графика)
2) Извлечение информации из изображений и видео (компьютерное зрение)
3) Обработка изображений и видео

Компьютерная графика применяется во многих отраслях: в сфере развлечений (дополненная реальность, компьютерные игры, спецэффекты, кино, мультипликация и анимация), в сфере безопасности (слежение за дорогами), в робототехнике, медицине, и на производстве и др.

Цифровое полутоновое изображение (черно-белое) – матрица размера NxM, элементами которой являются значения яркости света, измеренного на двумерной прямоугольной сетке. Так как в компьютере процессы дискретны, соответственно и изображения тоже дискретны. Размер одного значения обычно составляет ***один байт*** (или восемь бит) и принимает значения $\in [0, 255]$. Сейчас иногда используются и большие размеры (по 10-16 бит), например, в работе фотографов или фотохудожников.

Цвет – это психологическое (физиологическое) свойство человеческого зрения, возникающее при наблюдении объектов и света, а не физическая характеристика объектов и света. Это замечание важно, так как один и тот же объект можно осветить разным способом, в результате чего этот объект будет принимать разные цветовые значения. ***Цвет*** – это результат взаимодействия света, сцены и зрительной системы наблюдателя. Восприятие света человеком изучают науки *фотометрия* и *колориметрия*.

Что такое свет с физической точки зрения? Любой источник света можно описать ***спектром***: количество излученной энергии в единицу времени для каждой длины волны в интервале 380-720 нм (Рис. 1.1)

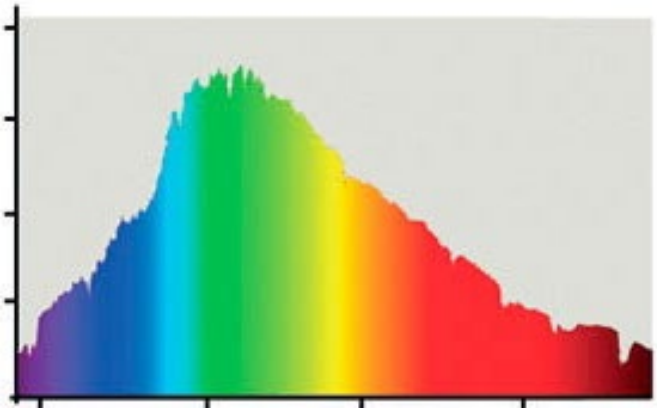

Рисунок 1.1. Спектр солнечного света. По оси x отмечено значение длины волны, по оси y – относительная чувствительность.

Чтобы достаточно точно описать цвет, необходимо знать значение интенсивности для каждой отдельной волны, что является большим объемом информации, который не обрабатывается человеческим глазом или мозгом. Отраженный от предметов свет существует в виде непрерывного спектра. Однако для того, чтобы разделить его на отдельные цвета, необходим инструмент – глаз человека, экран монитора или компьютер. В глазу (Рис. 1.2.) это делают специальные фоторецепторы, ***колбочки*** (измерение цвета) и ***палочки*** (измерение интенсивности освещения, яркости). Выделяется три типа колбочек, которые лучше воспринимают цвет из *синего*, *зеленого* или *красного* спектра. На этом принципе основана ***модель RGB***, в которой цвет для цветовоспроизведения кодируется с помощью трех ***базовых цветов*** (red - красный, green – зеленый и blue – синий, соответственно).

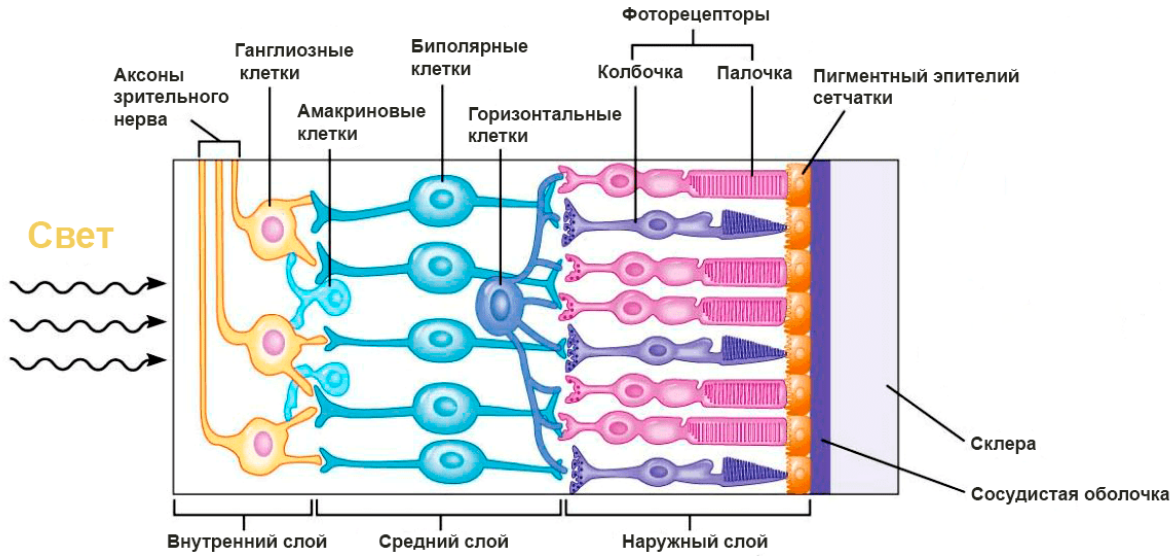

Рисунок 1.2. Строение сетчатки глаза. Фоторецепторы.

Трихроматическая теория – любой видимый свет можно получить при помощи линейной комбинацией трех *базовых* цветов (A, B, C). В результате получают положительную комбинацию aA + bB + cC.

Цветовые модели

Основываясь на трихроматической теории, в 1931 году компанией CIE, которая занимается стандартизацией цифровой информации, был проведен следующий эксперимент: несколько добровольцев посадили перед экраном, в котором было два отверстия, либо одно, разделенное некой плоскостью (Рис. 1.3). В одно отверстие попадал источник света с определенным целевым (тестовым) цветом, не являющимся базовым, а в другое – одновременно три базовых источника красного зеленого и синего цветов. Подопытному было дано три рычага, каждый из которых регулировал интенсивность излучения одного из трех базовых цветов. Задача подопытного состояла в том, чтобы регуляцией этих рычагов изменять цвета таким образом, чтобы комбинация трех базовых цветов в результате давала целевой цвет. Когда цвета в конечном итоге совпадали, положения интенсивности, которые были зафиксированы для трех базовых цветов становились *координатами* целевого цвета в цветовой модели RGB.

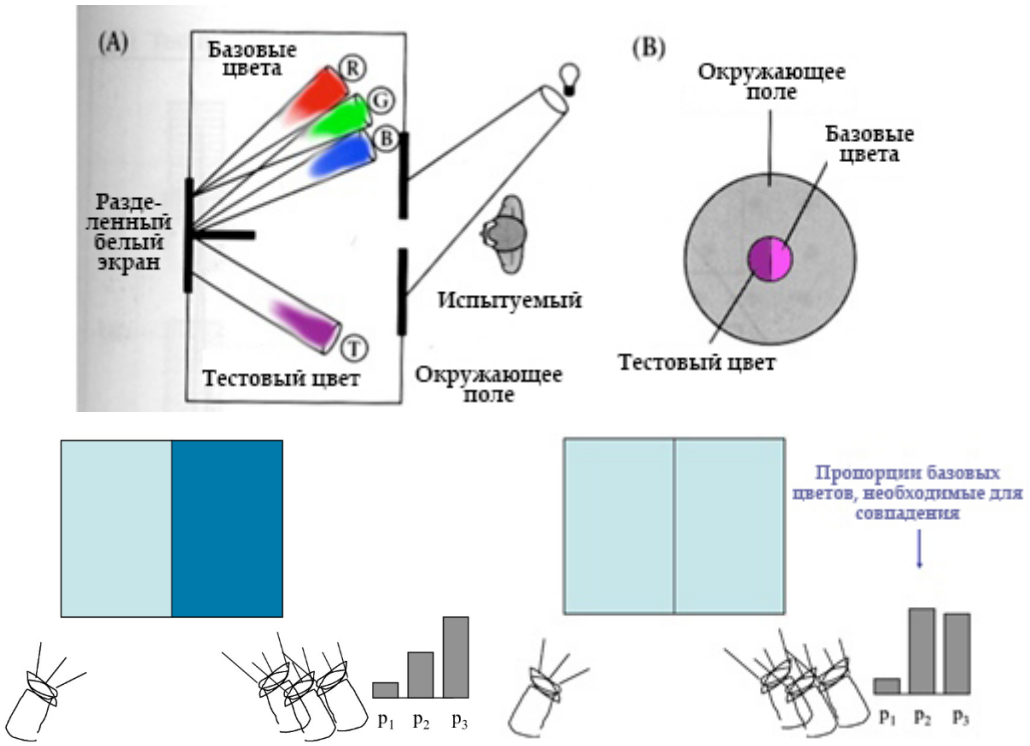

Рисунок 1.3. Эксперимент 1931г. Столбцами внизу как P1, P2, P3 обозначены значения координат, используемые для сопоставления целевому цвету (test light).

Закон аддитивности Грассмана: если наблюдатель задаст цвет лучей 1 и 2 как (R1, G1, B1) и (R2, G2, B2), тогда при суммировании источников 1 и 2, можно

воспроизвести их как (R1+ R2, G1+G2, B1+B2). Соответствие цветов выполняется на всех уровнях яркости. Т.е. если C1 = (R1, G1, B1), то kC1 = (kR1, kG1, kB1). Модели цвета, основанные на выборе базовых цветов и их линейных комбинациях – ***линейные цветовые модели***. Модель RGB является примером такой ***аддитивной*** линейной цветовой модели.

В модели CIE RGB 1931 были использованы следующие значения длин волн источников света P1 = 645,2 нм, P2 = 525,3 нм, P3 = 444,4 нм. Достаточно ли этих трех цветов для описания всех видимых цветов? Вышеописанная теория гласит, что при использовании трех источников можно получить любой видимый цвет. Однако, некоторые необходимые источники могут не существовать. Оказалось, что действительно есть такие цвета, которые невозможно было получить в данном эксперименте, как ни старались подопытные. Для этой задачи приходилось один из трех базовых источников добавлять к целевому цвету. Таким образом определенный цвет получал ***отрицательный коэффициент*** (***отрицательную координату***). Так как вычесть цвет нельзя, этой моделью оказалось невозможно описать все возможные видимые цвета. В результате этого были получены три функции, отображенные на графике на рис. 1.4. Функция, описывающая красный источник, имеет отрицательное значение на определенном этапе (причина проблемы модели CIE RGB 1931). «Вычитание» цвета необходимо для соответствия некоторым длинам волны.

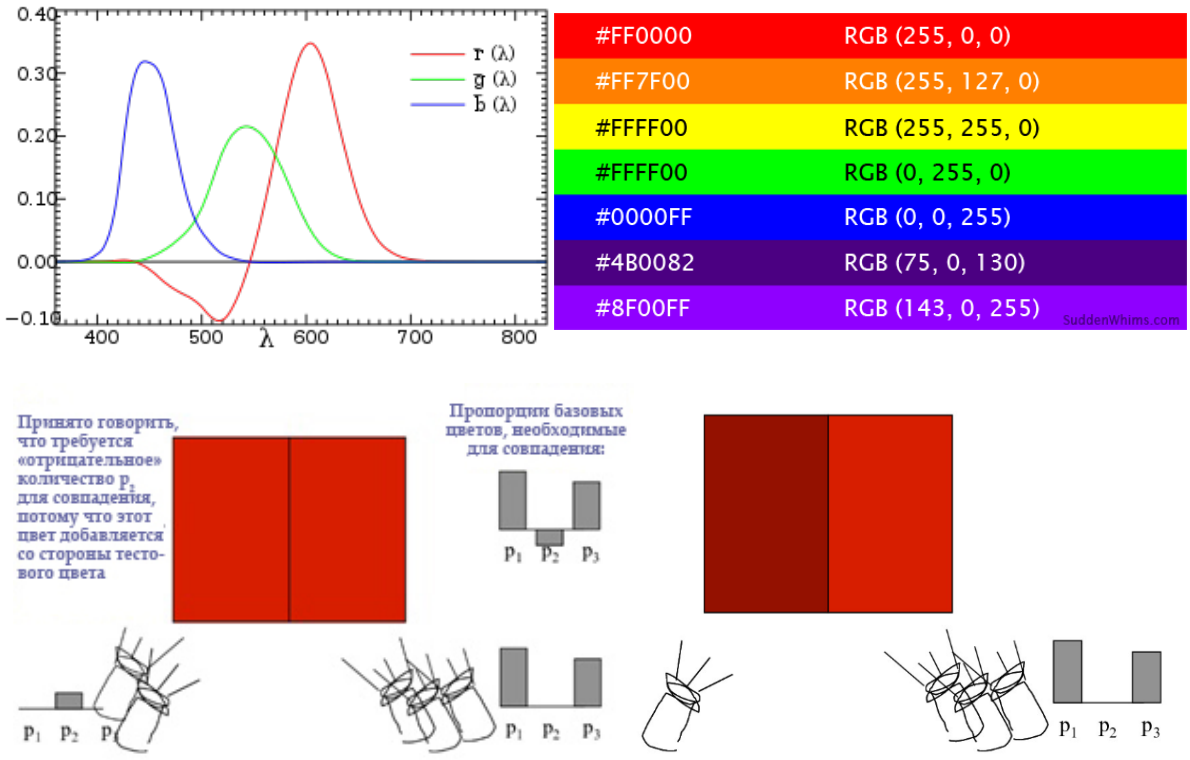

Рисунок 1.4. Функции трех источников, используемых в модели RGB. Цветовая кодировка, используемая в модели RGB. Отображение отрицательного коэффициента.

Вышеописанная RGB модель не является единственной. На данный момент в мире существует множество разных моделей, в которых в качестве базовых источников света используются различные цвета (Рис. 1.5). Самой популярной и широко используемой является ***sRGB***, которая сейчас по умолчанию используется как цветовая система в операционной системе Windows. Чаще всего именно ее используют для отображения цвета в мониторах компьютеров и экранах телефонов. В модели sRGB оказывается достаточно цветов, чтобы человек не испытывал дискомфорт или чувства нехватки насыщенности какого-либо цвета.

Существуют и более широкие цветовые пространства, такие как ***Adobe RGB***, которые используются в программах Adobe Photoshop или Adobe Lightroom, или ***ProPhoto RGB***, используемый в профессиональной фототехнике. В рамках данной лекции будет подробнее разобрана именно модель sRGB, которая является наиболее распространенной.

У цвета также выделяются две семантически ощутимые характеристики:

1) ***Яркость***
2) ***Цветность*** (комбинация ***тона*** [напр., красный или синий] и ***насыщенности***)

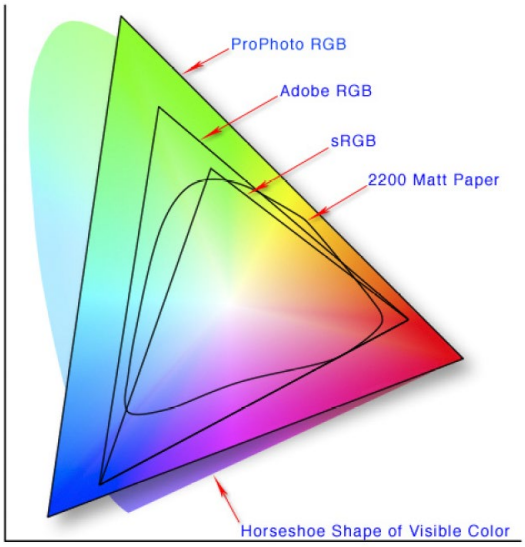

Рисунок 1.5. Виды линейных цветовых моделей.

В модели RGB эти характеристики не представлены в явном виде, что иногда может быть неудобным для работы. В иных моделях эти характеристики, например, яркость, выделены в виде отдельной координаты. Например, в модели ***YIQ,*** где координата Y представляет собой аппроксимацию реально воспринимаемой человеком яркости какого-либо цвета с помощью линейной комбинации RGB:

$Y = 0.299 * R + 0.587 * G + 0.114 * B;$

$I = 0.596 * R - 0.274 * G - 0.322 * B;$

Q = 0.211 * R - 0.522 * G + 0.311 * B;

где R, G, B — соответственно интенсивности цветов красного, зеленого и синего, Y — яркостная составляющая, I и Q — цветоразностные составляющие.

Существует и гораздо более интерпретируемая и удобная система ***HSV***, в которой три координаты обозначают: Hue – оттенок, Saturation – насыщенность и Value – интенсивность (Рис. 1.6). Удобно представлять эту модель в виде конуса, где по вертикали отложена интенсивность (внизу у вершины конуса независимо от насыщенности оттенка цвет черный), по радиусу отложена насыщенность (в центре ось конуса черно-белая, а ближе к краям оказываются сочные цвета), а по боковой поверхности конуса расположены оттенки (по меридианам). Эта система уже не является линейной, но ее очень удобно использовать для коррекции изображений по этим параметрам.

Другая распространенная модель, введенная благодаря печати изображений, ***CMYK***, основана не на сложении света (не аддитивная), а на сложении красок, т.е. поглощении света (***субтрактивная***). В отличие от RGB модели сумма всех цветов

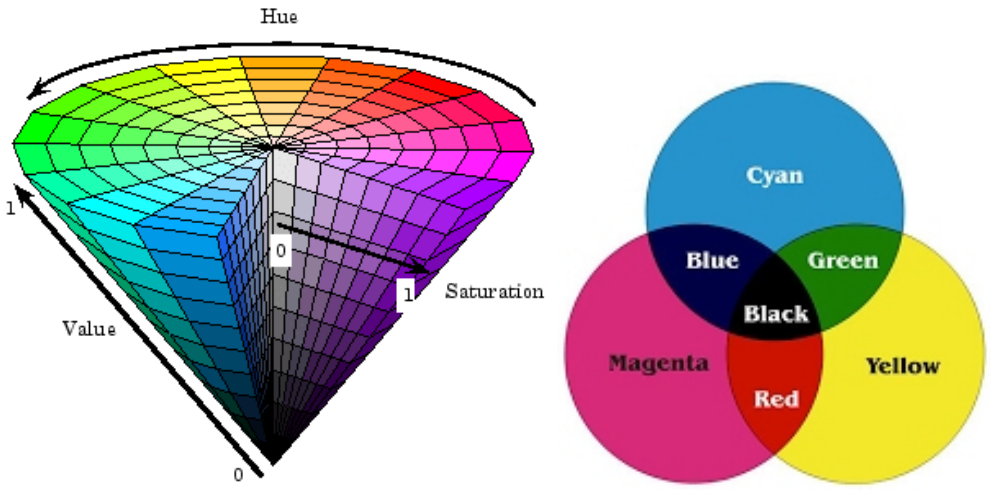

Рисунок 1.6. Системы HSV (слева) и CMYK (справа).

образует не белый, а черный (Рис. 1.6). Сходством этих моделей является то, что три цвета, используемые в ней, комплиментарны базовым источникам из системы RGB. Эту модель используют практически во всех принтерах. Так как краски, используемые при печати, не соответствуют электронным пикселям, а бумага не всегда идеально белая и обладает своими физическими свойствами, то в такой модели используется и черная краска. Название модели расшифровывается как Cyan, Magenta, Yellow, а черный цвет нельзя было назвать B (black), так как иначе возникла бы путаница с системой RGB, где B от Blue. Так, K на конце обозначает Key value. Таким образом, существуют и модели с большим, чем три, количеством базовых цветов.

Цифровое цветное изображение – матрица размера NxMxC, элементами которой являются значения одного из C каналов в каком-либо цветовом пространстве (в модели RGB C=3, вектор черного цвета, например, (0, 0, 0)). Также, как и в случае полутонового изображения размер одного значения обычно составляет ***один байт*** (или восемь бит) и принимает значения ∈ [0, 255]. Иногда используется большая точность (10-16 бит), такие изображения – ***изображения широкого динамического диапазона***, ***HDR***.

Как получить цвет из реального мира в компьютере? Когда человек фотографирует при помощи цифровой камеры, в этот момент сенсор в камере измеряет попадающий на него свет. В одной точке сенсора можно измерить только одно значение. Таким образом, можно либо измерить весь свет, который на него попал, в результате чего получается черно-белое одноканальное изображение, или можно наложением на сенсор цветовых фильтров измерять определенные цвета. Такой способ используется и в промышленности – на сенсор наложена ***цветовая мозаика Байера*** или ***Байеровский шаблон***. В результате получают одномерное изображение, в каждом пикселе которого содержится информация о разных цветах, после чего необходимо применить операцию ***демозаикинга*** – оценки пропущенных цветовых значений, т.е. восстановление неизвестных цветовых компонентов каждого пикселя, основываясь на известных значениях соседних пикселей. Эта операция стоит первой на пути от реального изображения к изображению на мониторе компьютера.

Обработка изображений

После получения изображения его можно изменить. Это может быть применено для следующих целей:

1) Для улучшения восприятия человеком
2) Для улучшения восприятия компьютером
3) Для технических нужд (извлечение определенных признаков)
4) Для развлечения (добавление спецэффектов и создание коллажей)

Изображение может потерять качество практически на любом этапе на пути из реального мира на экран монитора. Это может произойти на этапе мозаикинга на сенсоре в самом начале, в процессе дискретизации, при дискретизации значений из всего, что оказалось на сенсоре (например, 8 бит), при преобразовании данных, связанных с хранением. Также потери данных возможны при передаче информации с одного этапа на другой, при растеризации на мониторе, при передаче на монитор и отображении на нем. Основными ***дефектами изображения*** являются низкая яркость, слабый контраст, посторонние оттенки или неестественные цвета, шумы, размытие или повреждения на изначальном изображении. Некоторые дефекты будут разобраны подробнее далее.

Яркость и контраст

Чаще всего плохая яркость или контраст возникают из-за того, что чувствительность сенсора оказывается гораздо ниже, чем у человеческого глаза, т.е. из-за ограниченного диапазона чувствительности датчика. Также это может возникать из-за плохо работающей техники (передает информацию от сенсора в память с потерями или искажениями). В итоге чаще всего встречаются проблемы двух видов: либо на кадре что-то «пересвечено», а что-то «недосвечено», либо очень низкий контраст и изображение как будто «в тумане».

Как можно это исправить? Во-первых, необходимо оценить общее качество передачи тонов. Для этого строится ***гистограмма интенсивности пикселей,*** где по горизонтали откладываются все значения, которые могут принять пиксели и для каждого значения по вертикали откладывается их количество. Если изображение имеет плохое качество, то это может быть отображено двумя способами (Рис. 1.7): 1) Низкий контраст – диапазон яркости используются не полностью, т. е. в составе изображений отсутствуют очень темные и очень светлые пиксели; 2) Неравномерное заполнение диапазона яркостей – диапазон значений полон, но при этом почти все пиксели сгруппированы в одном узком участке, из-за чего сильно преобладает какой-то один вид освещения.

Для исправления могут быть использованы ***точечные операторы*** – это операторы, которые определяют значение выходного пикселя лишь по значению входного, т.е. все пиксели обрабатываются независимо друг от друга. Формула для коррекции:

$$f^{-1}(y) = x$$

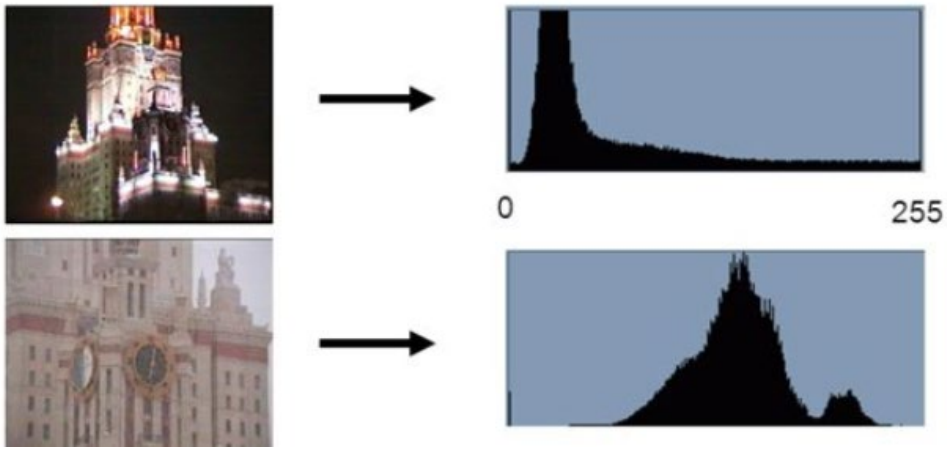

Рисунок 1.7. Оценка передачи тонов. Вверху - неравномерное заполнение диапазона яркостей, внизу - не полностью используется диапазон яркостей.

где y – яркость пикселя на исходном изображении, x – яркость пикселя после коррекции.

Здесь формула записана в «обратном виде»: y используется в качестве аргумента, а x в качестве значения, так как подразумевается, что изображение изначально должно быть хорошего качества, но по какой-то причине испортилось и необходимо вернуть исходное изображение.

Одним из способов компенсации узкого диапазона яркости является ***линейная коррекция***, при которой происходит *линейное растяжение* гистограммы (histogram equalization). При этом диапазон сдвигается к нулю, после чего растягивается на всю доступную длину.

$$f^{-1}(y) = (y - y_{min}) * \frac{(255 - 0)}{(y_{max} - y_{min})}$$

Линейная коррекция дает ощутимый эффект для изображений с низким контрастом. Однако, она не всегда устойчива, потому что на изображении могут

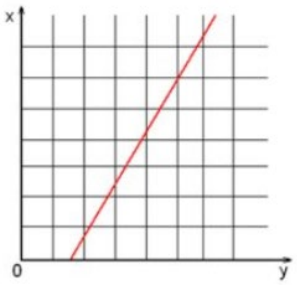

Рисунок 1.8. График функции $f^{-1}(y)$

присутствовать, например, маленькое количество очень темных и очень светлых пикселей из-за шума или других причин. В связи с этим, если такой метод применяется на практике, то обычно в качестве «краевых значений» диапазона, которые впоследствии подвергаются растяжению, выбираются не самые темные и светлые пиксели, а определенные интервалы значений - ***квантили***. Так, например, 5% светлых пикселей становятся белыми, а 5% темных пикселей – черными. Иными словами, решением проблемы является использование n% максимальных и n% минимальных в качестве границ приведения к 0 и 255.

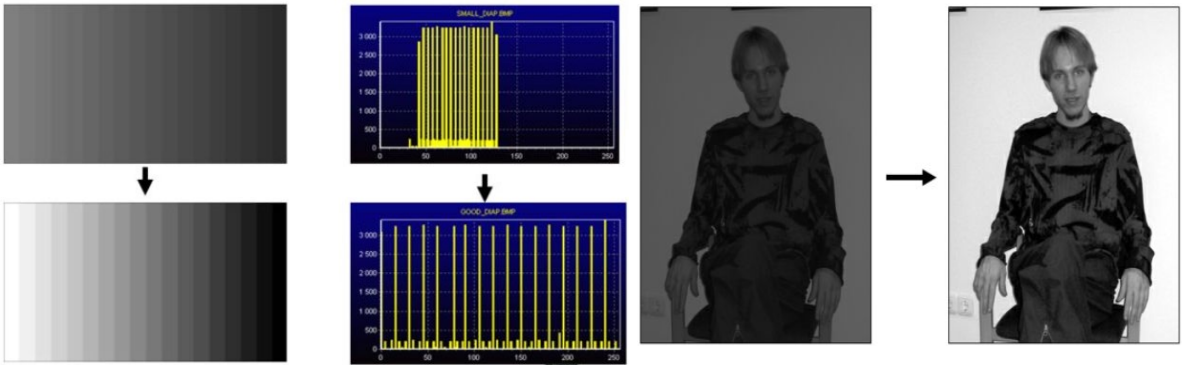

Рисунок 1.9. Линейная коррекция.

Такой способ работает часто, но не всегда. Это вызвано тем, что несмотря на то, что восприятие цвета человеком линейное, восприятие яркости – нелинейное. Если яркость распределить равномерно с точки зрения физики, то человеческому глазу это будет казаться намного более неравномерным. В связи с этим человеческим глазом лучше воспринимаются более субъективно равномерные яркости. Это происходит потому то глаз на самом деле гораздо более чувствителен к перепадам яркости в темных областях. Из-за этого для коррекции яркости используют ***нелинейную коррекцию*** изображений, в котором представляются наш значения пикселей так, чтобы сильнее воздействовать на темные пиксели и слабее на светлые. Одной из часто применяемых функций является гамма-коррекция.

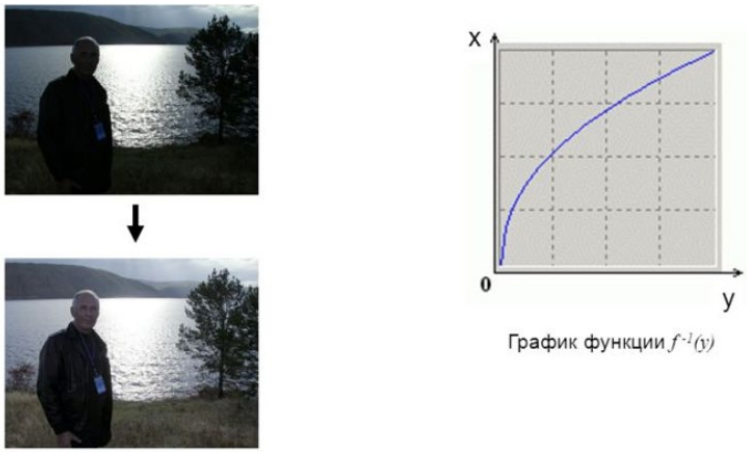

Рисунок 1.10. Нелинейная коррекция.

Гамма-коррекция — это преобразование вида $y = c \cdot x^{\gamma}$. Изначальной цельюявляется коррекция для правильного отображения на мониторе (на ЭЛТ-мониторах была нелинейная зависимость между напряжением и яркостью пикселя). Обычно значения яркости, которые изменил сенсор, линейные с физической точки зрения, а данный метод позволяет получить значения яркости боли равномерные с субъективной точки зрения (восприятие глазом человека). Гамма-коррекция направлена на то, чтобы человек мог увидеть больше информации на изображении. На Рис. 1.11 представлен пример того, как разные значения параметра гамма влияет на освещенность изображения изображения: становятся более различимы темные области, а светлый объект, девушка, достаточно хорошо видна на каждой фотографии. Значение параметра гамма, равное 2,2 – это стандарт в системе Windows и во многих камерах старого образца.

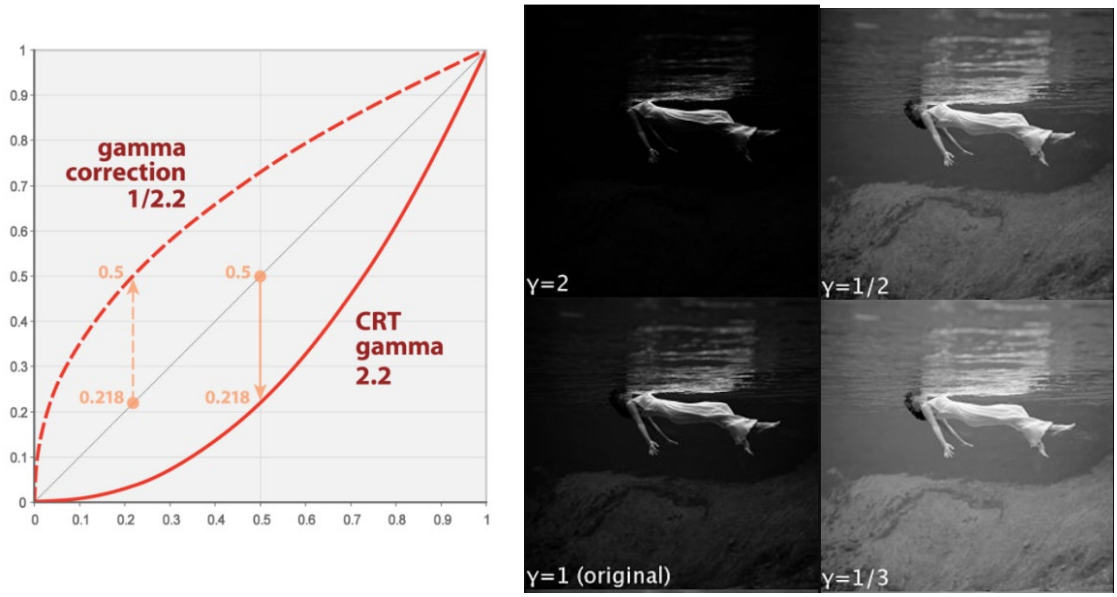

Рисунок 1.11. Гамма-коррекция.

Другой способ нелинейной коррекции – ***изменение формы кривой*** $\boldsymbol{f^{-1}(y)}$ в зависимости от того, что необходимо скорректировать в изображении (Рис. 1.12). При нелинейной коррекции необходимо учитывать, что чем круче кривая у нас в каком-то диапазоне, тем выше контраст именно в этом диапазоне. Если изменить крутизну кривой в одном месте, то она становится более пологой в другом, в результате чего в этой области информация теряется, контраст снижается. Таким образом, необходим анализ сцены, в каких областях нужно информацию сохранить и усилить, а в каких областях можно позволить потерять информацию. Основываясь на этом, принимается решение о форме кривой.

Другой тип коррекции – избавление от «плохих» оттенков или «плохих» цветов. Для этого существует коррекция путем изменения ***баланса белого***. Белый цвет под разным освещением выглядит по-разному. Если человеческие глаза и мозг вместе адаптируются так, что человек понимает, что бумага белая и под желтой лампой накаливания, и под голубоватой люминесцентной, то в свою очередь цифровая камера не может так адаптироваться, из-за чего могут появляться неестественные оттенки.

Часто белый цвет на полученных изображениях оказывается не белым, а голубым, оранжевым или другим неестественным цветом. Баланс белого позволяет это скорректировать. Для того, чтобы точно знать, какое значение имеет белый цвет, можно использовать *шаблоны,* в которых точно известен каждый цвет. Фотографии с шаблонами обычно делают в профессиональных студиях. Обыватели, когда корректируют цвета, чаще всего просто «угадывают» этот баланс цветов.

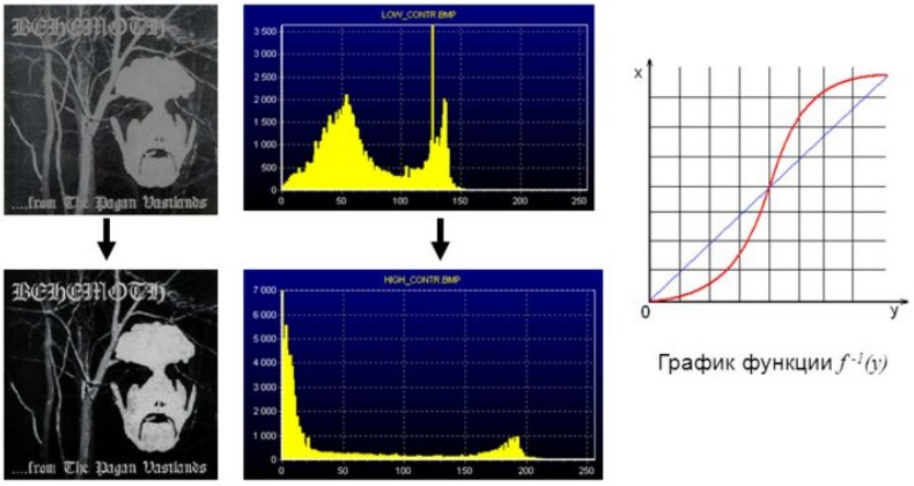

Рисунок 1.12. Коррекция изображения путем изменения формы кривой $f^{-1}(y)$

Самый простой способ угадывания – ***модель «Серого мира» (Grayworld)***. Если подразумевается, что в изображении присутствует посторонний оттенок, то среднее значение этого оттенка («серый») по изображению будет больше, чем значение других цветов. В идеале средний уровень по каждому каналу (в данном случае в одном из трех каналов системы RGB) должен быть одинаков для всех каналов. Для коррекции необходимо вычислить среднее значение по каждому каналу, привести их к одному значению, а после – вычислить мультипликативные коэффициенты для каждого канала, основываясь на том, какое среднее значение необходимо получить:

$$\bar{R} = \frac{1}{N}\sum R(x,y);\ \bar{G} = \frac{1}{N}\sum G(x,y);\ \bar{B} = \frac{1}{N}\sum B(x,y);\ \ Avg = \frac{\bar{R}+\bar{G}+\bar{B}}{3};$$

$$R' = R \times \frac{Avg}{\bar{R}};\ G' = G \times \frac{Avg}{\bar{G}};\ B' = B \times \frac{Avg}{\bar{B}}$$

Это решение помогает справиться с сильными дефектами баланса белого и посторонними оттенками. Существуют и другие способы угадывания, в частности ***модель блика***. Известно, что блики на изображении обычно имеют цвет источника света. Необходимо найти самый яркий пиксель и использовать веса для каналов, пропорциональные его цветам. Другой способ – ***линейное растяжение гистограммы для каждого канала***. Он заключается в выравнивании значений по каналам, если подразумевается, что при хорошем балансе цветов для каждого канала должен быть одинаковый диапазон. Однако, на самом деле это не так, и приводит к искажению цвета.

Шумоподавление

Другая проблема, с которой можно столкнуться – ***шум на изображении***. Шум может возникать по нескольким причинам: плохо работающая техника (несовершенство сенсоров), внешнее электромагнитное воздействие на сенсор (самая частая причина), загрязнение объектива, потери при сжатии данных и их передаче.

Фотографии, сделанные ночью гораздо сильнее подвержены шуму, так как в темноте сенсор становится более светочувствительным и старается уловить любой свет.

Первым видом шума на изображении является ***потеря информации (data drop-out noise)***. Это может происходить при потере пикселей с заменой их на черные и белые (***соль и перец, salt and pepper noise***), или при ***импульсном шуме***, когда возникают случайные белые пиксели. Это также может быть шум, который можно аппроксимировать каким-либо распределением, чаще всего Гауссовым или нормальным распределением.

Гауссов шум – колебания яркости, распределенные по нормальному закону. Такой шум является аддитивным: можно представить изображение как наложение картинки без шума с изображением, на котором присутствует только шум. Как корректировать такой дефект? В случае такого шума значение матожидания либо ноль, либо очень близко к нулю. Если сделать много абсолютно одинаковых картинок неподвижной сцены на неподвижной камере, а затем их все вместе сложить и усреднить, то чистая часть изображения не изменится, а весь шум при этом нивелируется. Такой способ называется ***временной фильтрацией*** и в реальности он очень осложнен тем, что очень трудно сделать абсолютно одинаковые изображения.

Пространственная фильтрация (свертка)

В таком случае применяется иной способ коррекции, ***пространственная фильтрация***. Вспомним вышеописанный точечный оператор, где каждый пиксель обрабатывался независимо друг от друга. В случае пространственной фильтрации каждому пикселю присваивается функция его окрестности – чаще всего это просто линейная комбинация, т.е. просто взвешенная сумма пикселей в заданной окрестности. Для шумоподавления можно просто усреднить значение весов окрестностей, тем самым слегка размыв изображение. Значения веса пикселей окрестностей, изображенные в окошке 9x9, называют ***ядром фильтра или ядром свертки*** (Рис. 1.13, слева).

Пусть f – изображение, g – ядро. Свертка изображения f с помощью g обозначается как $f * g$ и называется:

$$(f * g)[m,n] = \sum_{k,l} f[m-K, n-l]g[k,l]$$

О свертке можно говорить как о так называемом «окошке, едущем по изображению», которое обрабатывает пиксель за пикселем в зависимости от его окрестности (Рис.1.13, справа). Необходимо экстраполировать изображение для фильтрации по краям (***padding, паддинг***). Как происходит фильтрация по краям? Можно заполнить все изображение снаружи пикселями с нулевым значением, (clip filter [black]) либо повторять значения на границах изображения (copy edge), либо зеркально отражать изображение от края (reflect across edge), либо замостить изображение (wrap

around). На практике чаще всего используются либо «паддинг нулями», либо «зеркальный паддинг».

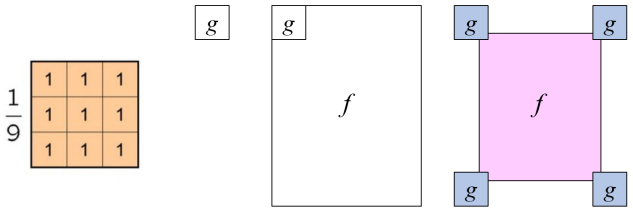

Рисунок 1.13. Ядро фильтра (слева). Свертка и паддинг (справа).

Свойства свертки:

1) ***Ассоциативность***: a * (b * c) = (a * b) * c

Последовательное применения нескольких фильтров свертки (((a * b_1) * b_2) * b_3) эквивалентно применению фильтра их свертки между собой a * (b_1 *b_2 * b_3).

2) ***Дистрибутивность по сложению***: a * (b + c) = (a * b) + (a * c)

Сумма двух фильтров равняется сумме сверток этих фильтров.

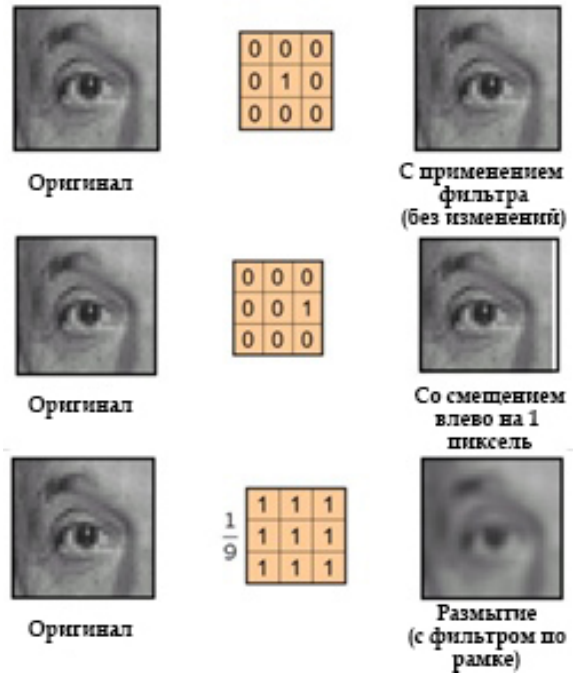

Рисунок 1.14. Примеры сверток.

Пример работы свертки представлен на рисунке 1.14. Если свертка представляет из себя единицу в центре – ничего не происходит, потому что выходному пикселю присваивается значение входного пикселя (Рис. 1.14, вверху). Если пикселю присваивается значение правого соседнего пикселя, то это приведет к тому, что изображение сдвинется на один пиксель влево (Рис. 1.14, по центру). Если усреднить

по окрестности, допустим из 9 пикселей, то возникнет эффект размытия (Рис. 1.14, внизу).

Box-фильтр

При использовании фильтров с положительными коэффициентами, сумма которых будет равняется единице, изображение будет размываться по-разному. Сумма всех пикселей в свертке обычно равняется единице для того, чтобы после свертки в изображения не присутствовали какие-то искажения по яркости: единица энергии «пришла» на входе, и единица энергии «вышла». Усредняющий фильтр носит название ***box-фильтра***. Однако у этого фильтра есть проблема: при размытии возникают угловые артефакты или артефакты в виде линий (Рис. 1.15).

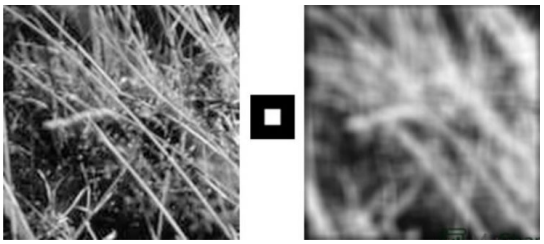

Рисунок 1.15. Box-фильтр. Артефакты, возникающие при свертке при помощи box-фильтра.

Фильтр Гаусса

Точка света, наблюдаемая с расфокусированного объектива, выглядит как кружок света, а усреднение давалось квадратным фильтром. Для более корректного размытия логично использовать те веса в фильтре, которые будут обратно пропорциональны расстоянию от центра фильтра. В этом помогает ***двумерная функция Гаусса***, а фильтр, соответственно, называется ***фильтром Гаусса***:

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

где параметр σ – стандартное отклонение (влияет на «пологость горки» и степень размытия, Рис. 1.16).

Необходимо учитывать, что функция Гаусса определена на бесконечной плоскости, а фильтр должен иметь конечный размер. Какой размер фильтра выбрать, чтобы не было обрывов его значений? Эмпирическим путем было показано, что хорошим подходом является использование размера фильтра, равный трем значениям параметра σ (3σ). Сглаживание при помощи Гауссова фильтра не имеет угловых артефактов, возникающих при использовании box-фильтра.

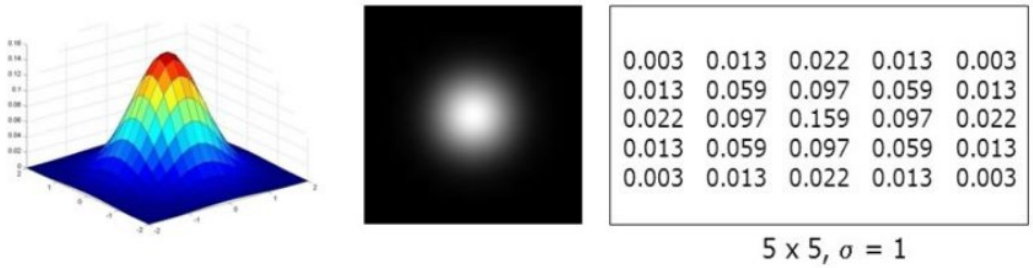

Рисунок 1.16. Фильтр Гаусса.

Медианный фильтр

Гауссов фильтр не идеален, так как он неустойчив к «выбросам» - к сигналам, которые сильно выделяются из всей нужной массы сигнала. Поэтому возникает необходимость в другом типе фильтра, который после себя не будет создавать новых значений. Таким фильтром является ***медианный фильтр*** (Рис.1.17). Если ранее описанные фильтры использовали взвешенные суммы внутри своего «окошка», то медианный фильтр берет медиану своих весов и присваивает ее центральному пикселю. Это помогает бороться с нежелательным воздействием сильных выбросов (сильного шума). Однако, несмотря на то что медианный фильтр не так сильно «размывает» края, как Гауссов или box-фильтр, при его использовании теряются (не сглаживаются) некоторые детали изображения, оно становится «однородным».

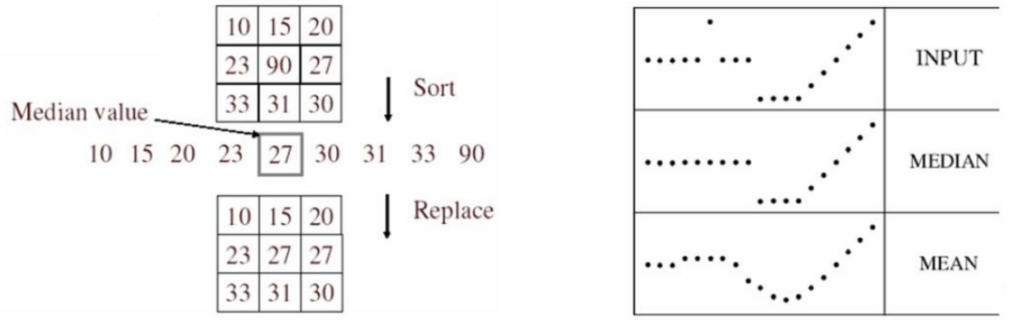

Рисунок 1.17. Принцип работы медианного фильтра.

Обратная задача – ***повысить резкость*** размытого изображения. Если изображение размыто, оно потеряет какую-то часть своих деталей. В таком случае можно вычесть размытое изображение из исходного, в результате чего останутся только детали. Полученные детали можно прибавить к исходному изображению и получится повышение резкости. С математической точки зрения все эти процессы можно сделать за одну операцию свертки (Рис. 1.18, слева).

Фильтр-шляпа

В результате получается фильтр, который выглядит как вычитание фильтра Гаусса из усиленного единичного фильтра. Полученный фильтр имеет вид, который называется ***Лапласиан Гауссиана или фильтр-шляпа***. Такой фильтр служит как

самый простой способ повышения резкости изображения. Так, например, если использовать фильтр размера три на три, в котором по центру оказывается большое положительное значение, а по краям – маленькие отрицательные, и нормировать его, то такой способ поможет повысить резкость размытого изображения (Рис. 1.18, справа).

Выделение краев

Более сложной задачей является ***выделение краев***. Края изображения очень важны, так как благодаря краям можно понять, какие предметы находятся в пространстве и какое между ними соотношение. Как понять, где располагаются края в изображении?

Рисунок 1.18. Повышение резкости.

Они могут возникать при резком изменении в нормали поверхности (изгибы предметов), при резком изменении глубины (расстояние от камеры или глаза до объекта) и при резком изменении цвета или яркости у предмета. По своей сути край – резкое изменение значения ***функции интенсивности*** изображения, что вычисляется при помощи вычисления первой производной этой функции (Рис. 1.19).

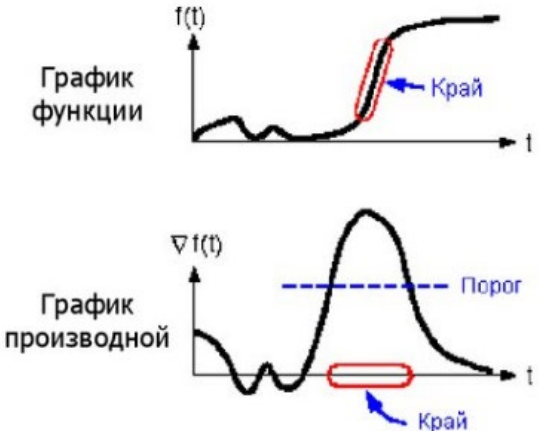

Рисунок 1.19. Обнаружение краев в изображении при помощи производной функции интенсивности.

Градиент изображения задается точно так же, как и у любой функции от двух переменных – это производная по двум направлениям (по горизонтали и по вертикали). Таким же образом задается и само значение градиента – как вектор из этих двух значений, также и направление градиента, и модуль градиента (квадратный корень из суммы квадратов этих двух значений).

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Градиент направлен в сторону наибольшего изменения интенсивности. Его направление задается как:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{2x}\right)$$

Изображение – это функция, но функция дискретная, поэтому обычная производная не подходит, необходима ***разностная производная*** (***производная на сетке***):

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

Если посмотреть на производную, например, по горизонтали, то окажется, что на самом деле это просто вычитание двух изображений. Такой результат можно получить при свертке с использованием простейшего фильтра -1 1. Производную изображения можно получить при прохождении такого фильтра по изображению. Однако, необходимо помнить, что разностные производные очень чувствительны к шуму. В связи с этим перед тем, как считать такие производные, необходимо размыть изображения. Не смотря на потерю части деталей, такой способ помогает избавиться от шума и при этом не сильно повредить края. Свертка может сочетать в себе оба процесса за один раз:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

Существует множество разных версий фильтров для получения производной изображения. Например, фильтр Робертса, Превитта или Собеля (Рис. 1.20). Также фильтры могут быть разных размеров. Чаще всего для вычисления градиента изображения используется фильтр Собеля размером три на три (Рис. 1.20, справа).

Рисунок 1.20. Популярные разностные фильтры.

Алгоритм Кэнни

Если размер фильтра маленький, то возникает очень «слабое» изображение с четкими краями (края достаточно точные по пространству, но зачастую «слабые» – границы будут пропадать), а если большой, то края будут более «сильные», с большим модулем, более толстые и размытые. Поэтому, когда необходимо найти края, то одного градиента недостаточно, из-за того, что у краев либо мало точности, либо они будут пропадать из-за их «слабости» и информация о связанности будет теряться. Кроме того, чтобы посчитать градиенты необходимо сделать еще несколько шагов, зафиксированных в ***алгоритме Кэнни***:

1) Свертка изображений ядром – производной фильтра Гаусса
2) Поиск направления и модуля (сила) градиента
3) Подавление локальных максимумов вдоль направлений градиента (позволяет истончить найденные края до 1 пикселя)
4) Отсечение по двойному порогу (гистерезису), заданному вручную
    1. Края с силой выше верхнего порога считаем сильными и оставляем
    2. Края с силой ниже нижнего порога отбрасываем
    3. Края между порогами остаются потенциальными кандидатами
5) Потенциальные края считаем итоговыми краями, если они прилегают к сильным краям

Данный алгоритм работает достаточно хорошо, однако он сильно зависит от используемых порогов, от степени размытия фильтром Гаусса.

Лекция 2. Изображения широкого динамического диапазона.

Общие понятия

Изображением широкого динамического диапазона, или High Dynamic Range Images, HDRI, называют изображение, которое содержит более 8 бит различных передаваемых значений яркости. Речь в данном термине идет именно о яркости, а не о цвете. При этом 8-битные изображения называют ***изображениями узкого диапазона*** или ***изображениями стандартного динамического диапазона***. При использовании узкого динамического диапазона теряется информация о светлых или темных областях. Использование изображений широкого динамического диапазона значительно уменьшает потери информации.

Динамическим диапазоном называется отношение между максимальным и минимальным уровнем значения некоторой измеряемой величины (цвет, звук или другой стимул). ***Минимум*** часто определяется минимальным различимым значением, которое определяется уровнем помех и шума. Впоследствии это соотношение обычно подвергается логарифмированию. Для фотографий используется такая величина, как ***стоп*** – двоичный логарифм этого отношения. В обработке звука используются ***децибел*** – десятичный логарифм отношения. Система зрения и обработки информации человеческого организма способна ориентироваться в пределах 10 порядков изменений яркости (*динамический диапазон*), а в пределах одной сцены – до 5 порядков яркости (*статический диапазон*), в то время как даже самые современные HDR мониторы не способны одновременно отражать больше трех порядков яркости (*статический диапазон*). Это означает, что ни одно современное устройство вывода изображений не может передать всю информацию о сцене, которую воспримет наблюдающий ее человек. Однако, существуют способы, позволяющие увеличить объем значимой информации (при этом придется отказаться от некоторой менее значимой).

Пространственное и временное разрешение

Почему для описания яркости не хватает 8, 10, а иногда даже 12 бит? Существуют непреодолимые ограничения на формат вывода – дисплей, его дискретизация: пространственная (разрешение изображения, dpi, ppm), временная (Hz), хроматическая (число значений по цвету), яркостная (число ступеней яркости).

В первую очередь рассмотрим ***пространственное и временное разрешение***. ***Разрешение*** зоркого глаза - одна угловая минута (1/60 градуса). Это значит, что с расстояния 20 см предельно различимая плотность пикселей составляет 436 пикселя на дюйм. На расстоянии 50 см – 174 пикселя на дюйм, на расстоянии 100 см – 87 пикселей на дюйм. Современные устройства способны выводить изображения с очень плотной частотой пикселей - Samsung galaxy 10 имеет плотность 550 пикселей на дюйм, т. е. этот телефон можно поднести к глазам практически вплотную и не видеть границ между пикселями. Таким образом, качество изображений современных устройств

оказывается достаточно высоким. Сходная ситуация наблюдается и для ***частоты*** – для большинства сцен достаточно 60 Hz (в современных мониторах 120 Hz), однако, чем выше яркость, тем больше нужно FPS. Таким образом, дискретизации по времени в современных устройствах также достаточно.

Человеческое зрение способно различать до 150 разных оттенков тона и до 25 оттенков насыщенности. Это означает, что глаз может дискредитировать цвет примерно на несколько тысяч значений. Дискретизация современных цветовых моделей уже превосходит эти показатели. Однако, человеческий глаз способен различать от 5 порядков яркости без адаптации зрачка и до 10 порядков яркости с адаптацией зрачка, а мониторы передают до 10 бит – 3 порядка. Яркость измеряется в *канделах на метр квадратный*. Самая темная сцена, которую наблюдает человек в естественной среде, это ночное небо без луны (10^{-5} cd*m^2), которое отличается на 10 порядков от самой яркой сцены - ясного дня (10^5 cd*m^2). При этом максимальная статическая яркость обычного монитора составляет около 10^4 cd*m^2, поэтому для передачи изображений на обычный монитор необходимо использовать специальные алгоритмы (так как не хватает передачи мониторов по яркости).

Задача тональной компрессии

Рассмотрим следующую задачу. Имеется некоторая сцена, которая в условиях планеты Земля может демонстрировать десять порядков яркости. В других солнечных системах порядков яркости будет больше, однако в данной задаче это не имеет значения. Для начала необходимо получить изображение объекта с помощью камеры или другого записывающего устройства, затем происходит хранение и обработка, а в конце – вывод на дисплей. Такая последовательность называется ***трактом получения и визуализации изображений*** (Рис. 2.1). Можно ограничиться узким динамическим диапазоном и преобразовывать свет в изображении узкого диапазона сразу в камере, что обычно и происходит. На этом этапе применяется гамма-коррекция, затем полученное 8-битное изображение хранится и выводится на устройство. Все изображения формата bmp, png и jpeg являются примером таких изображений. Однако, можно пойти по другому пути: не сжимать информацию сразу, а сохранить изображение широкого динамического диапазона и вывести его также в широком динамическом диапазоне. Для этого существуют дисплеи широкого диапазона, на которых можно визуализировать такие изображения. Однако есть способ показать изображение широкого динамического диапазона и на обычном мониторе, у которого 2-3 порядка яркости. Это достигается преобразованием из расширенного диапазона яркостей в суженый диапазон яркости, что называется ***задачей тональной компрессии***. Если сделать несколько LDR фотографий одной сцены с разной выдержкой, то окажется, что на одних фото потеряна информация (детализация) на светлых участках, а на других – на темных. Используя HDR изображения, т. е. сохраняя весь диапазон яркости и затем обрабатывая его с помощью алгоритма тональной

компрессии, можно получить изображение, значительно превосходящие исходную фотографию по реализму и художественной ценности. Также HDR может значительно повышать насыщенность и яркость изображения, сделанного при достаточно оптимальном значении выдержки (не было потерь информации).

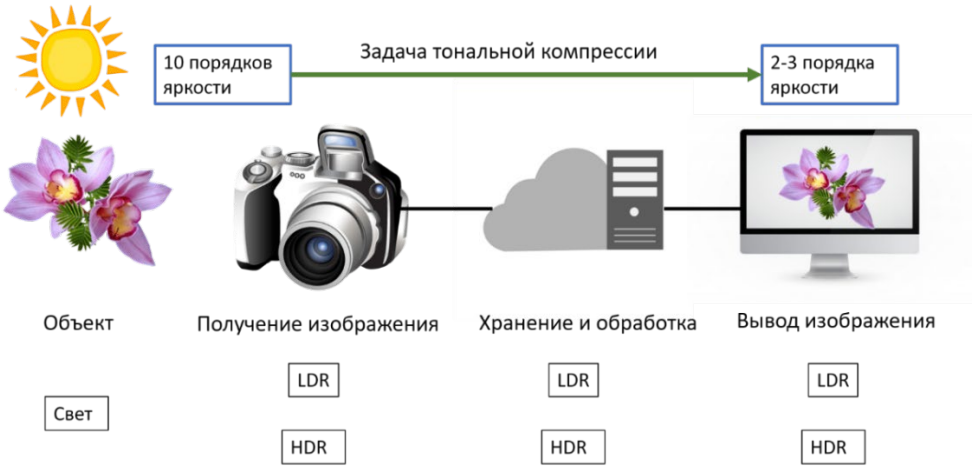

Рисунок 2.21. Тракт получения и визуализации изображений.

Визуализация (рендеринг)

В рамках курса стоит задача синтеза нового изображения: весь блок получения, хранения и обработки изображения реализуются в компьютерной программе. Для этого необходимо создать симуляцию процессов, происходящих при получении изображения на камеру и смоделировать его запись и хранение с обработкой. Можно использовать OpenGL, для материалов и источников – триплеты цветов (везде использовать LDR значения яркости). В таком случае не нужно делать моделирование, все происходит за счет системы визуализации. В результате получается изображение достаточно хорошего качества. Можно использовать трассировщик лучей, не применять усиленную детализацию к спектру источника и не делать пост-обработку (Рис. 2.2). Результат будет еще лучше.

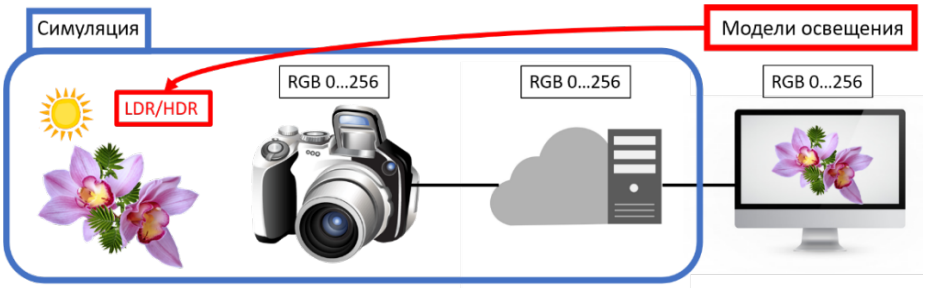

Рисунок 2.22. Не фотореалистичный рендеринг.

Можно использовать ***фотореалистичный рендеринг***, в рамках которого рассматривать физические яркости и спектры, и при достаточной детализации моделей

получить очень реалистичное изображение. Более того, можно промоделировать некоторые физические процессы, например, рассчитать освещенность в точке поверхности. На этом этапе симуляции использование HDR становится необходимостью, чтобы сохранить рассчитанную информацию и получить высокохудожественное изображение. Для такой симуляции необходимо использовать ***интегральное уравнение рендеринга*** (с физической яркостью):

$$L(p, \omega_0) = L_e(p, w_0) + \int_{\Omega} f(p, \omega_0, \omega_i) L_i(p, \omega_i) |\cos \theta_i| \, dw_i$$

где L – суммарная яркость, проходящая из точки p в направлении ω_0,

L_e – яркость, исходящая из точки p в направлении ω_0 (p – источник света),

L_i – яркость, приходящая в точку p из направления ω_i,

$f(p, \omega_0, \omega_i)$ – BDRF,

ω_0, ω_i – направления, заданные телесными углами.

После того, как блок объект-свет промоделирован, необходимо рассчитать цвет, который впоследствии будет передан на дисплей. Первый способ – промоделировать процесс фотографирования – построить модель камеры, для чего необходимо продублировать оптическую систему, матрицу Байеровского шаблона, и затем промоделировать обработку внутри камеры (например, осуществить простейшую гамма-коррекцию). Плюсом такого подхода является то, что модель простой камеры легко реализовать. Если же необходимо смоделировать полноценную камеру – будет сложнее. Однако у камер есть ограничения, как уже было описано, когда невозможно сфотографировать сцену так, чтобы не потерять детализацию.

Второй способ заключается в следующем. Необходимо построить физически точную модель, использовать спектры источников, получить изображение широкого динамического диапазона, «урезать» его до 10-12 бит и подать на HDR-монитор (или урезать до 8 бит и подать на монитор LDR). Как «урезать» – избавиться от лишней информации и при этом не потерять значимую? В теории необходимо построить модель глаза и мозга. Эта задача на данный момент не является полностью решенной (модель глаза изучена достаточно хорошо, а то, как работает зрительная кора, ученые представляют только в самых общих чертах). Промоделировать этот процесс позволяют алгоритмы тональной компрессии, которые с некоторой степенью точности описывают работу зрительной системы.

Можно ли на устройстве ограниченного диапазона показать изображение HDR, сохранив значимую часть информации? Этим на протяжении всей истории цивилизации занимались художники. На картине солнечного дня кажется, что солнце ярче, чем небо. Однако, если перевести изображение в монохром, то окажется, что это

не так. Объективная светимость краски одинакова у солнца и у неба, но в мозг человека уже заложена информация о том, что солнце яркое. Когда человек видит даже схематичное изображение солнца на ярком небе, мозг все равно считает, что солнце ярче. Это свойство человеческого восприятия может быть использовано в алгоритмах с получением того же эффекта. Известный художник Архип Куинджи создавал иллюзии света в своих картинках при помощи красок с одинаковой светимостью.

Таким образом, HDR-изображения используются повсюду: в рендеринге игр, для использования панорам освещения и в фотографии. Однако, нерассмотренными остались вопросы: как получить HDR-изображение, как его хранить и как визуализировать? На них можно ответить или с точки зрения технологий, или с точки зрения алгоритмов. Сначала рассмотрим технологии HDR-изображений.

Технологии HDR

В общем случае технология изображений со стандартным диапазоном состоит в следующем: получают некоторое изображение, которое сразу на этапе камеры сужается до изображения LDR, затем хранится, обрабатывается и передается на экран, и затем снова воспринимается глазом (Рис. 2.3, вверху). Для некоторых сцен диапазона этого оказывается достаточно, однако изображение может оказаться блеклым. В случае ***pipeline с HDR изображением***, не происходит значительного урезания диапазона, и в результате он весь передается на HDR экран, а изображение кажется более приближенной к наблюдаемой в реальности (Рис. 2.3, внизу). Существует нюанс, дисплей с широким динамическим диапазоном способен передавать максимум 10-12 бит, а человеческому зрению нужно около 20 бит.

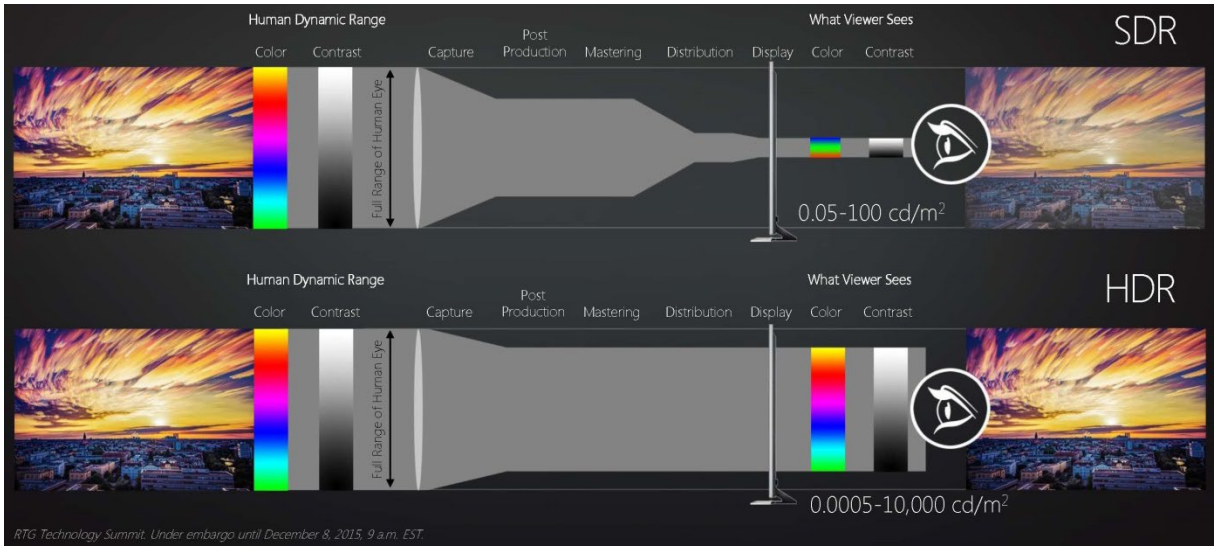

Рисунок 2.23. Отличия в получении SDR и HDR изображений.

Как получить изображение широкого диапазона с помощью записывающей аппаратуры? Во-первых, существуют специальные камеры. Это очень перспективный

путь, но не распространенный, поскольку такая аппаратура очень дорогостоящая. Иногда создают еще более сложное устройство, которое в себе сочетает две синхронизированные камеры, настроенные на разную выдержку. Эти камеры одновременно в каждый момент времени снимают одну сцену с двумя значениями выдержки и затем, с помощью некоторых алгоритмов, эти два значения объединяются и восстанавливается вся яркость сцены (Рис. 2.4). Немецкие исследователи за несколько лет наладили работу этой системы и сняли несколько видеофильмов, которые можно впоследствии использовать для тестирования алгоритмов тональной компрессии. Таким образом, для того, чтобы получить HDR-изображение, необходимо приобрести дорогостоящую камеру или создать свою систему, потратив на это много времени (годы).

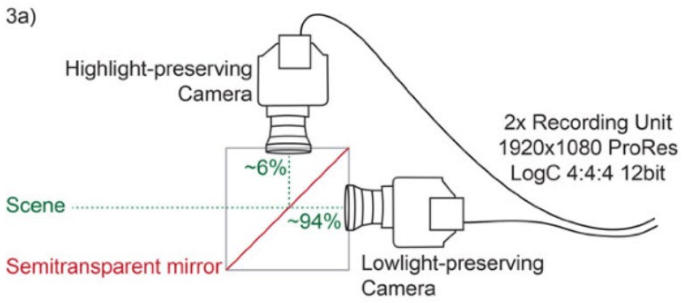

Рисунок 2.24. Устройство, созданное немецкими учеными для получения HDR-изображений.

Хранение HDR

Как хранить HDR-изображения? Существует целый ряд форматов хранения таких изображений, у каждого из которых есть свои плюсы и минусы. Простейший способ - ***сохранить RAW данные камеры*** – такое изображение можно обработать не гамма-коррекцией или каким-то простым алгоритмом, встроенным камеру, а более сложным алгоритмом на ПК или смартфоне. RAW данные занимают в памяти устройства больше места. Такие данные также имеют расширенный диапазон, но неравный диапазону зрения.

Другой формат хранения – ***HDR***. Его суть состоит в том, чтобы сохранить отдельно цвет и яркость. Замечено, что источники редко бывают цветными и насыщенными, обычно они монохроматические, и из-за этого пиксели на изображении обладают четко выраженной корреляцией – если два канала цвета, например, зеленый и синий, и отложить по оси x интенсивность пикселя в зеленом канале, а по оси y эту величину в синем канале, то в результате будет получено множество точек, близкое по форме к диагонали (Рис. 2.5). Таким образом, существует корреляция между значениями в зеленом и синем каналах. Аналогичная корреляция существует между красным и синим каналом, красным и зеленым каналом. Это означает, что можно хранить меньше информации – только такую кривую яркости (разброс яркости на

изображении) и впоследствии применять только небольшие модуляции для красного, синего и зеленого каналов. В случае формата HDR так и происходит – хранится общая экспонента яркости, на которую умножаются отдельно мантиссы красного синего и зеленого цветов (это позволяет хранить HDR в 32 битах, что на 1 байт больше, чем формат BMP):

$$E = [\log_2(\max(R_w, G_w, Bw)) + 128]$$

$$R_M = \left\lfloor \frac{256\, Rw}{2^{E-128}} \right\rfloor \qquad R_w = \frac{R_M + 0{,}5}{256} 2^{E-128}$$

$$C_M = \left\lfloor \frac{256\, G_w}{2^{E-128}} \right\rfloor \qquad G_w = \frac{G_M + 0{,}5}{256} 2^{E-128}$$

$$B_M = \left\lfloor \frac{256 Bw}{2^{E-128}} \right\rfloor \qquad B_w = \frac{B_M + 0{,}5}{256} 2^{E-128}$$

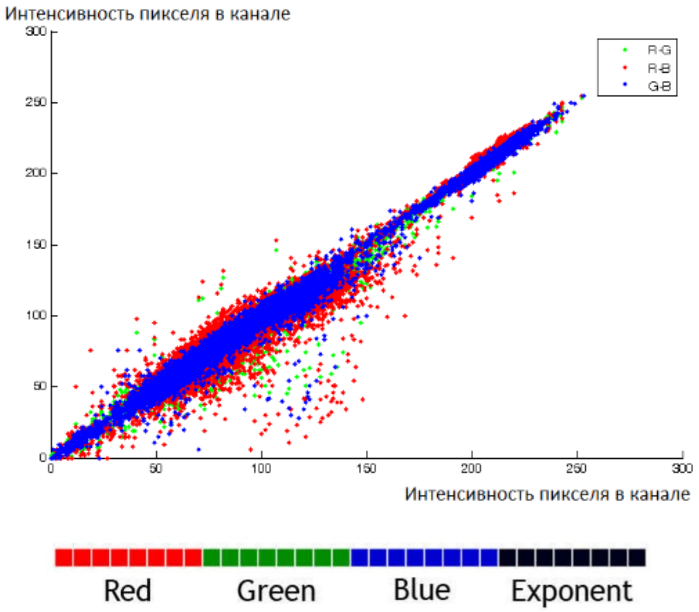

Рисунок 2.25. Формат HDR.

Другой формат хранения – ***TIFF***. Он является общим форматом хранения изображений узкого диапазона. У данного формата есть множество вариаций, но в случае HDR он позволяет осуществить сжатие без потерь – можно хранить информацию даже в *числах с плавающей точкой яркости* по каждому из каналов. Недостаток этого формата заключается в том, что при таком хранении используется 96 бит на пиксель, что в три раза больше, чем в формате HDR. Для того, чтобы немного уменьшить объем информации, авторами формата TIFF было изобретено ***LogLuv кодирование*** – яркость хранится отдельно, ее квантуют на 10 значений и отдельно хранится цвет, записывая в 14-битный индекс точку на диаграмме тона и насыщенности (Рис.2.6).

$$L_15 = \lfloor 256(\log_2 Y_w + 64) \rfloor \qquad L_{10} = \lfloor 64(\log_2 Y_w + 12) \rfloor$$

$$Yw = 2^{\frac{L_{15}+0{,}5}{256}-64} \qquad Y_w = 2^{\frac{L_{10}+0{,}5}{64}-12}$$

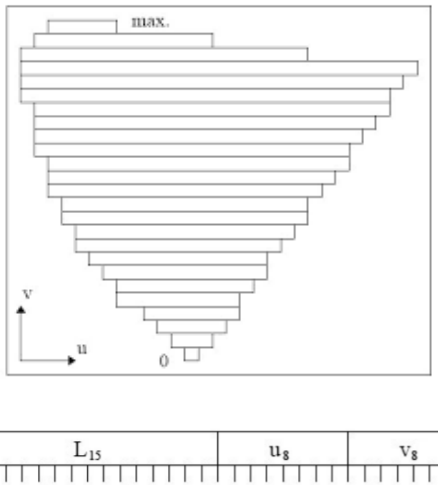

Рисунок 2.26. LogLuv кодирование. Распределение битов LogLuv кодирования 32-бит.

Еще один формат хранения – ***OpenEXR*** (наиболее рекомендуемый формат в рамках курса). Этот формат также достаточно вариативен и поддерживает хранение 16, 24, 32 бит на канал. Этот формат позволяет использовать различные алгоритмы сжатия (ZIP, PIZ – формат сжатия на основе вейвлетов) и хранить дополнительные данные, такие как прозрачность или глубину сенсора глубины. Он также использует разложение на мантиссу и экспоненту (Рис. 2.7), но отдельно для каждого из каналов. Формат OpenEXR позволяет хранить большой диапазон значений 0,000061-65,504 (до 9 порядков, включая числа меньше 0,000061), и также имеет очень маленький шаг квантования <0,1% (у HDR шаг квантования больше, поскольку на мантиссу отводятся меньше бит). Пример кодирования 16-бит:

$$h = \begin{cases} (-1)^5 2^{E-15}\left(1 + \dfrac{M}{1024}\right) \\ (-1)^5 2^{-14}\dfrac{M}{1024} \end{cases}$$

где $l \leq E \leq 30, E = 30$

Рисунок 2.27. Разложение бит в OpenEXR кодировании

Другой, более редкий формат – ***сжатие HDR с потерями***. Идея заключается в том, что любое изображение можно представить в виде двух изображений, умноженных друг на друга «попиксельно», одно из которых можно выводить на LDR монитор, а второе использовать, если монитор обладает расширенным диапазоном. Изображение с меньшим разрешением можно хранить в формате JPEG.

Визуализация HDR

Существует два основных подхода визуализации изображений: в рамках первого подхода напрямую регулируется яркость пикселя, а второй подход использует два или более модуляторов. В первом подходе необходимо очень точно варьировать яркость каждого пикселя и цвет одного пикселя не должен перетекать на соседний. Этого можно добиться при помощи двух основных технологий. Первая - ***сканирующий лазерный дисплей (SLDT Scanning Laser Display Technology developed by JENOPTIK GmbH)***, который обеспечивает хороший статический контраст 100000:1, поскольку очень низкий уровень черного цвета достигается при выключенном лазере, а также мощные яркие лазерные RGB источники обеспечивают хороший цветовой охват. Недостаток метода заключается в том, что пятно лазера «подразмывается», в результате чего пиксели немного наезжают один на другой. Также значимым недостатком является высокая цена источников. Второй подход – ***органические излучающие диоды Organic light emitting diodes, OLED***. Этот способ дешевле, но его максимальная яркость оказывается достаточно низкой.

В подходе с использованием ***модуляторов*** используется подсветка, пассивный экран или проектор и два экрана. Один из них представляет собой обычную LCD-панель, а другой – попиксельно включает дополнительную подсветку (похожим образом, что и при использовании формата хранения с разложением на две плоскости). Для каждого изображения строятся две текстурные карты и затем они умножаются, уже на физическом уровне. Существует другой способ устройства, когда имеется некоторый источник света, который затем разделяется на три равных луча, каждый из которых проходит через свой модулятор и свой светофильтр. В итоге собираются лучи синего, зеленого и красного цвета. Эти модуляторы управляются программным образом, потом эти три луча снова смешиваются и через линзы проектора выводят HDR изображение.

Еще один способ – ***DLP-проектор***. Это широко распространенное устройство.

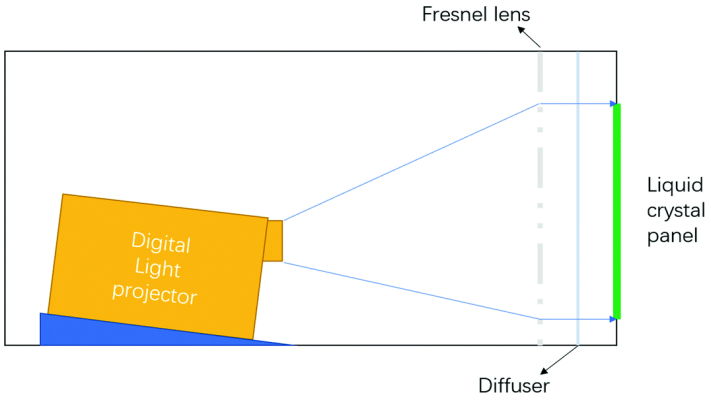

Рисунок 2.28. Схема простейшей визуализации HDRI с использованием модуляторов.

В нем светофильтры нанесены на вращающийся диск, который вращается с очень большой частотой, луч света проходит через один из светофильтров, затем отражается и попадает на управляемое микрозеркало, которое либо отражает луч либо в камеру, либо в сторону и тем самым гасит его. Минусами такого подхода и устройств такого типа является то, что несмотря на неплохой результат, им свойственна засветка от соседних пикселей (утечка яркости на соседние пиксели), также они ослабляют или перенаправляют свет – появляется не очень темный черный цвет (слишком яркий) и возникают энергопотери.

Алгоритмы HDR

Изображение HDR можно получить как путем синтеза, так и из набора некоторых фотографий, снятых LDR камерой. Можно также снять HDR видео с применением более сложных алгоритмов, которые способны убрать дефекты-призраки. Для визуализации на обычном мониторе можно использовать алгоритмы тональной компрессии. В случае применения их к видео последовательностям, необходимо позаботиться об устранении мерцания. Существует алгоритм, позволяющий напрямую построить качественное LDR изображение из набора LDR фотографий.

Первый способ получения изображений – путем ***синтеза изображений***. В этом случае HDR изображения представляют собой результат работы алгоритмов визуализации. Второй способ - ***комбинирование LDR изображений с разной выдержкой***.

В первом случае необходимо рассмотреть уравнение рендеринга (описанное выше), рассчитать освещенность в каждой точке на матрице фотоаппарата. Краткое описание механизма работы рендеринга – спектр источника взаимодействует с материалами, лучи разной энергии летят в камеру (глаз наблюдателя), после чего можно применить кривые чувствительности глаза (или кривые чувствительности матрицы камеры, если моделируется камера) и получить RGB значения разной величины.

В реальности свет ведет себя следующим образом (Рис. 2.9): спектр отражается, затем применяются кривые чувствительности глаза и после этого что-то происходит в мозгу человека (пока точный механизм неизвестен в науке). Для моделирования отражения на практике используют упрощение или не рассматривают весь спектр, а только три луча – красного, зеленого и синего цвета, используют для них сложную модель материала двулучевую функцию отражения и получают достаточно качественное изображение. Такой способ подходит для рендеринга HDR панорамы, но можно также обойтись и без HDR и осуществлять рендеринг, используя более простые модели материала: модели Фонга, Ламберта и др. Также можно рассматривать источник как флотовые единицы от нуля до единицы и приближенно описывать свойства материалов. Однако при использовании системы такого уровня абстракции имеет смысл моделировать HDR только для моделирования адаптации наблюдателя к сцене. Например, такой подход используется в компьютерных играх.

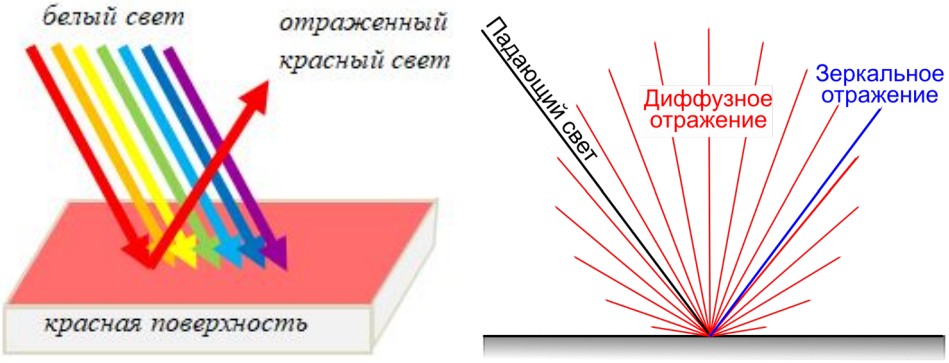

Рисунок 2.29. Отражение света и восприятие цвета в реальном мире.

На практике часто используются ***HDR панорамы*** – некоторые карты окружения для последующего освещения. Сначала фотографируют или синтезируют некоторую панораму, сферическую, цилиндрическую или кубическую, затем вокруг наблюдателя строится некоторый большой объект, удаленный от него, а затем на него «натягивается» карта окружения. Каждая точка это карты окружения рассматривается как источник света, проецируемый на каждый объект в сцене. Это значительно позволяет повысить реализм изображения. Также использование HDR-панорам важно, если имеются некоторые полупропускающие объекты, например, стекла с рельефом. Если не использовать HDR, то потеряется значительная часть энергии и могут возникать нереалистичные диффузные размытые блики.

Второй способ, как говорилось выше, это ***получение HDR из наборов фотографий с разными выдержками***, если имеется слабая камера, которая производит «пересвеченные» или затемненные фотографии. При регулировании экспозиции можно получить весь диапазон яркости фотографии. После этого необходимо установить, какие значения яркости были исходно в заданной точке для пикселя P_{ij} (красная точка на Рис. 2.10).

Предположим, что отклик камеры линеен. Это означает, что некоторое значение яркости пришло в эту точку, затем она была выдержана с величиной выдержки T1 и в результате возникла некоторая яркость A в диапазоне от 0 до 256, и так для каждой фотографии. Значения выдержки известны, они хранятся при съемке в ZIP-файле. Для получения исходной яркости достаточно яркость пикселя поделить на выдержку, что следует из уравнения:

$$I_{iJ} * T_1 = A_{iJ}$$

и уравнений для других фотографий:

$$I_{iJ} * T_2 = B_{iJ} \qquad I_{iJ} * T_3 = C_{iJ}$$

Если бы не производилось отсечения диапазона от 0 до 256, то в теории хватало бы одной LDR фотографии, но так как на каждой фотографии теряется та или иная информация, то в реальности так не работает, отклик камеры никогда не бывает строго

Рисунок 2.30. Получение HDR из набора фотографий с разными выдержками.

линейным, и, более того, он всегда неизвестный. В результате для того, чтобы найти исходную яркость, необходимо не умножение, а функция:

$$f(I_{iJ} * T_1) = A_{iJ} \qquad f(I_{iJ} * T_2) = B_{iJ} \qquad f(I_{iJ} * T_3) = C_{iJ}$$

В общем случае необходимо произвести линеаризацию отклика камеры, найти функцию f, затем привести яркости всех пикселей в единое пространство, а затем усреднить три полученных приближенных значения, отбросив ненадежные значения, либо применив функцию смешивания. Например, если один и тот же пиксель при трех значениях выдержки получился яркостью 50, 150 и 250, то желательно выбрать значение пикселя с большим весом, яркость которого соответствует диапазону наиболее хорошо передаваемых (надежных). В результате, таким образом, можно восстановить HDR изображение. На рисунке 2.11 разными цветами закодировано с какой фотографии пришли пиксели с большим весом.

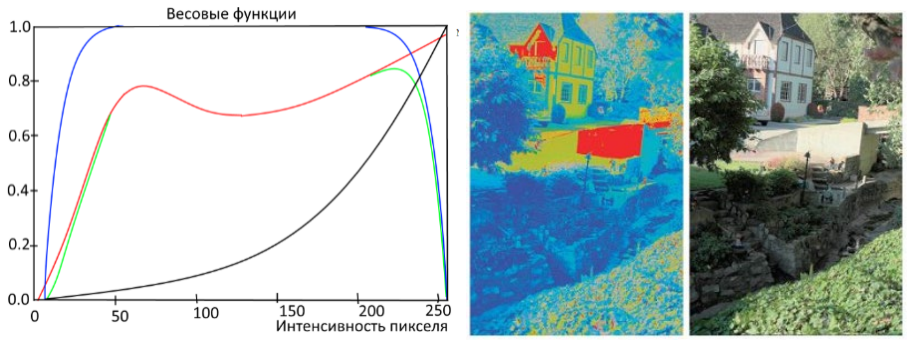

Рисунок 2.31. Изображение с использованием ложных цветов, отображающая относительные значения яркости пикселей.

Необходимо восстановить кривую отклика камеры, поскольку не всегда кривая отклика камеры реализует гамма коррекцию, чаще производители вносят изменения в кривую, чтобы скрыть шум, увеличить насыщенность и контраст. Также, если необходимо сделать съемку движущихся объектов, или случайно дернулась камера, тонужно скомпенсировать эти эффекты. Для восстановления кривой отклика используется оптимизационный процесс:

$Z_{iJ} = f\left(E_i * \Delta(t_J -)\right)$ - значение экранной яркости пикселя i на кадре J

$$f^{-1}(Z_{iJ}) = E_i * \Delta(t_j) \qquad \ln\left(f^{-1}(Z_{iJ})\right) = \ln E_i + \ln \Delta(t_J) = g(Z_{iJ})$$

Если сделано пять фотографий с разными значениями выдержки, в каждой взято три пикселя, то можно создать кривую зависимости $g(Z_{iJ})$ от Z_{iJ}, где пиксели окажутся на одной линии (Рис. 2.12). Благодаря таким предположениям получается восстановить параметры кривой отклика камеры.

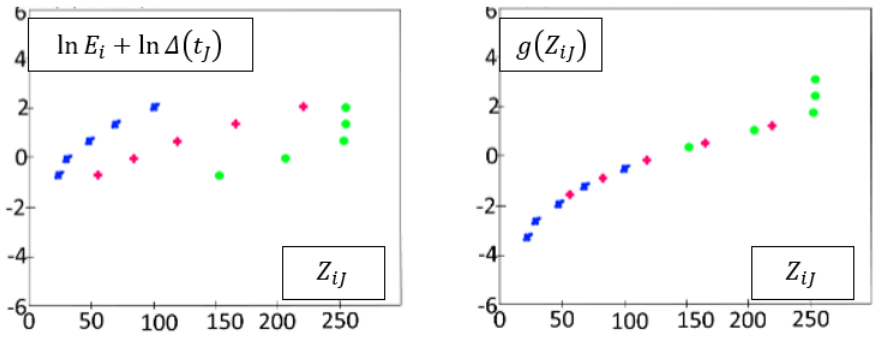

Рисунок 2.32. Восстановление кривой отклика камеры из изображений.

Предположим, что при съемке видео с разной выдержкой на фиксированном штативе оказалось, что по сцене проехал велосипедист. Его положение на каждом кадре менялось. Если использовать алгоритм в таком случае, то возникают артефакты,

гостинг (ghosting). Для того, чтобы избавиться от ***призраков***, существуют сложные алгоритмы, которые не будут рассматриваться в рамках курса.

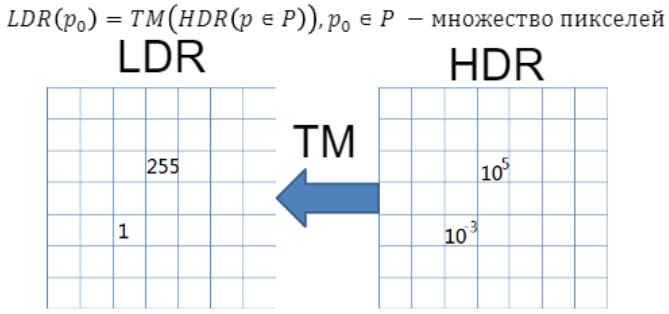

Рисунок 2.33. Преобразование HDR в LDR при помощи алгоритма тональной компрессии.

Рассмотрим ***алгоритм тональной компрессии*** (Рис. 2.13). Задача стоит следующим образом: необходимо отобразить изображение широкого диапазона в изображении узкого диапазона с минимальными потерями информации с точки зрения человека.

Таких алгоритмов очень много и каждый год появляются новые. По способу реализации их можно классифицировать на ***глобальные операторы*** и на ***пространственно-зависимые операторы***. По целям использования выделяют алгоритмы:

- Требующие абсолютные значения\относительные значения в каждом пикселе
- Теоретические\экспериментальные (подобраны эмперически)
- Яркость\детали\художественная ценность

Рассмотрим некое тестовое изображение, сделанное при различных значениях экспозиции (Рис. 2.14). Такая сцена очень сложная, так как у нее очень широкий диапазон яркостей – множество пересвеченных и затемненных областей при разных значениях выдержки.

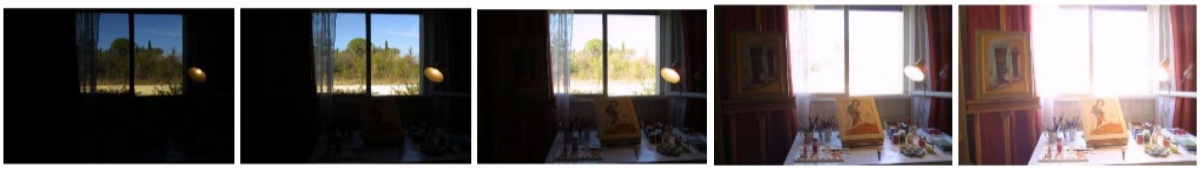

Рисунок 2.34. Тестовое изображение.

Рассмотрим ***глобальные операторы***. Они применяют идентичную кривую сжатия для всех пикселей – для каждого пикселя поочередно применяется одинаковая формула (Рис. 2.15). Достоинство этих алгоритмов заключается в их высокой

эффективности, а недостаток в том, что они не справляются с очень широким диапазоном. Глобальные операторы можно легко создать самостоятельно. Можно использовать линейное сжатие, отбрасывание предельных значений, нелинейную

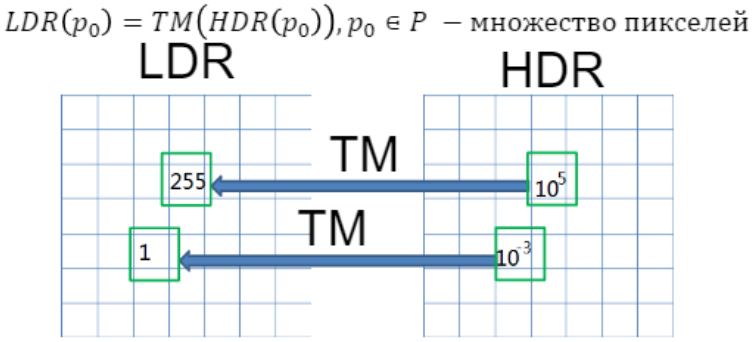

Рисунок 2.35. Глобальные операторы.

коррекцию или комбинацию этих методов. Метод или комбинация подбирается под конкретную сцену.

Лучше использовать ***адаптивное логарифмическое отображение для контрастных сцен*** (***Adaptive Logarithmic mapping for Displaying High Contrast Scenes***). В нем используется *закон Вебнера-Фехнера*, логарифмическое восприятие, учитывается глобальная адаптация, ограничивается максимум выходной яркости и используются некоторые параметры сдвига, которые можно настраивать вручную. На рисунке 2.16 представлены кривые, которые входят в семейство кривых, описанных этих этим алгоритмом. В зависимости от настроек можно по-разному использовать данный алгоритм.

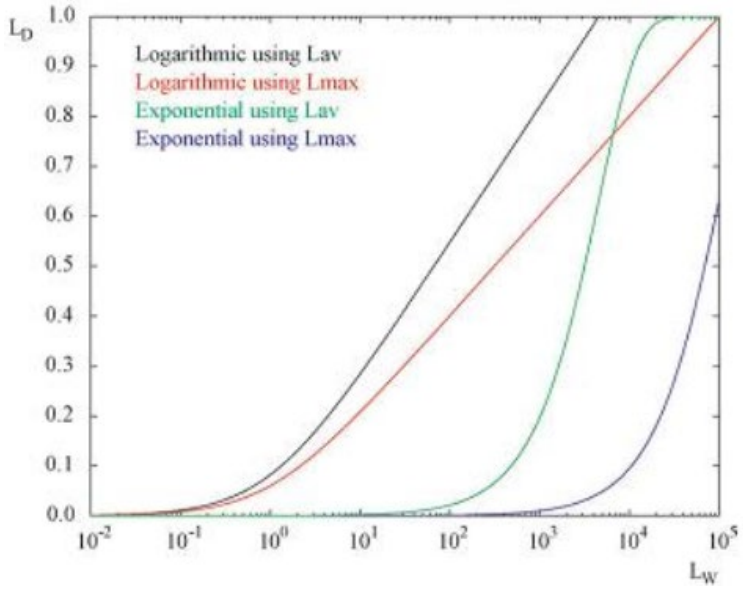

Рисунок 2.36. Adaptive Logarithmic Mapping for Displaying High Contrast Scenes.

Преобразование в данном алгоритме описывается с помощью двух формул. В начале рассчитывается статистика, ***ключ сцены***, и затем на него нормируются все яркости изображения, после чего используются логарифмические кривые с настраиваемым параметром b. Сама по себе формула сложна, но алгоритм значительно упрощается за счет того, что одна формула применяется одинаково к каждому пикселю.

$$L_m = \exp\left(\frac{1}{N}\sum_{P \ni N} \ln(y_P)\right) Lw = \frac{y}{Lm}$$

$$L_d = \frac{L_{dmax} * 0{,}01}{\log_{10}(L_{wmax}+1)} * \frac{\log(L_w+1)}{\log\left(2+\left(\left(\frac{L_w}{L_{w\max}}\right)^{\frac{\log(b)}{\log(0{,}5)}}\right)*8\right)}$$

где L_{dmax} – потенциальный максимум устройства, L_{wmax} – максимальное значение нормированной яркости по изображению, b (~0,7-0,9) – регулирует отображение пикселей с низкой и средней яркостью.

В свою очередь, в случае ***пространственно-зависимых операторов*** на вход берется не одно значение пикселя, которое нужно отобразить, а значения соседствующих пикселей, что может повлиять на яркость в итоговом изображении (Рис. 2.17).

Рассмотрим ***локальные алгоритмы***, которые используют такой подход как инвертирование контраста. Предположим, что есть солнце с яркостью 10000, вокруг него небо с яркостью 10, тогда на горизонтальном срезе яркость будет меняться ступенчатым образом (Рис. 2.18, слева). Можно применить такой алгоритм тональной компрессии, что небо станет просто синим, а солнце – просто желтым, яркости у них станут равны, как

$$LDR(p_0) = TM\big(HDR(p \in P_w)\big),$$
$$p_0 \in P \ - \text{множество пикселей},$$
$$P_w \in P - \text{подмножество пикселей, включающее } p_0$$

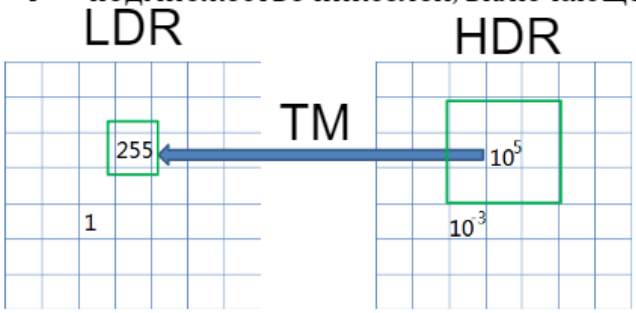

Рисунок 2.37. Пространственно-зависимые операторы.

было выше описано для картин (Рис. 2.18, справа). Наблюдатель будет понимать, что солнце ярче неба, хотя на деле это не так. Однако, чтобы добавить эффект свечения, похожий на то, что делал Архип Куинджи, необходимо применить подход ***инвертирования контраста*** - по мере приближения к солнцу небо должно быть темнее, что усилит контраст, за счет чего солнце будет казаться ярче. Для этого нужно применить следующую формулу:

$$L_d(x,y) = \frac{1}{kL_w^{blur}(x,y)} L_w(x,y)$$

Необходимо к яркости исходного изображения применить фильтр Гаусса, изображение в результате размывается, затем исходное изображение делится на размытое, которое помножили на некоторый коэффициент.

Похожий подход используется в алгоритме ***фотографической кривой репродукции, Photographic Tone Reproduction for Digital Images.*** Этот алгоритм использует зонную систему, помечает различные участки, относит их к разным зонам

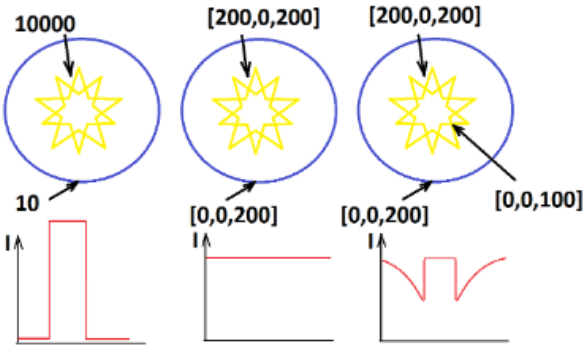

Рисунок 2.38. Общий подход локальных методов. Инвертирование контраста.

(использует сегментацию), а затем локально применяет ретушь – селективно осветляет и затемняет участки изображения. Это схоже с вышеописанным инвертированием контраста. Этот процесс имитирует то, что делают фотографы, когда аналоговым образом обрабатывают фотографию.

Существуют еще две группы методов: градиентные и частотные методы. ***Частотные методы*** разделяют изображение на плоскость освещенности и плоскость «отражательности». Установлено, что человек менее чувствителен к изменениям освещённости и более – к динамике изменений «отражательности». Таким образом, информацию об освещенности можно сжимать. Частотные методы сжимают большие градиенты в областях источника света – они выделяют некоторые градиенты (производные) на изображении, сжимают большие градиенты, оставляют маленькие, а затем применяют обратно градиенты к оставшимся базовому изображению и получают результат.

Частота изображения может быть представлена в виде ***двумерного преобразования Фурье***. Каждому изображению соответствует некоторое другое изображение в преобразовании Фурье (на самом деле таких изображений два, так как присутствует вещественная и мнимая часть). В каждой точке указан коэффициент для

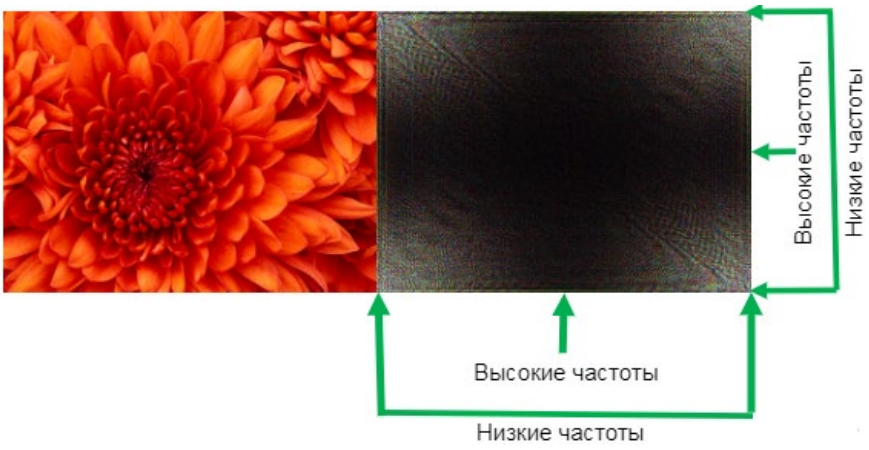

Рисунок 2.39. Двумерное преобразование Фурье.

одного из элементов *базиса Фурье*. Центральная область соответствует высоким частотам, а краевые области – низким. Изображение на рисунке 2.19 конвертируется в пространство Фурье, а потом обратно «в себя». Поэтому можно вырезать в пространстве Фурье область высоких частот, конвертировать назад и в результате получится размытое изображение. Это еще один пример размытия, помимо фильтра Гаусса.

На рисунке 2.20 представлена функция контрастной чувствительности человеческого глаза. Белые и черные полосы чередуются, сужаясь в направлении слева направо. На рисунке наблюдается некоторый пик из этих полос (когда человеческий глаз способен различить их) и спады по краям. Человеческое зрение чувствительно к определенному набору частот, что позволяет сжимать очень низкие (а иногда и высокие) частоты.

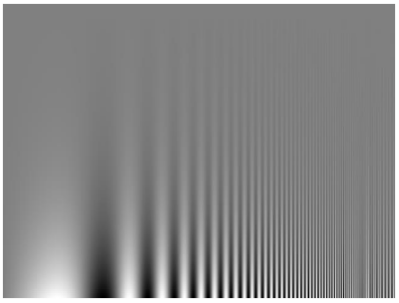

Рисунок 2.20. Функция контрастной чувствительности человеческого глаза.

Примером частотного алгоритма является ***быстрая билатеральная фильтрация, Fast Bilateral Filtering for the Display of HDRI***. В ней выделяется

плоскость освещенности и отражательности, и сжимается освещенность. Примером ***градиентного метода*** является ***Gradient Domain HDR Compression***. С его помощью создаются насыщенные и контрастные, но иногда не самые реалистичные изображения.

В случае, если тональная компрессия применяется к видео последовательностям, то возникает необходимость бороться с такими проблемами, как мерцание (если алгоритм базируется на неустойчивой статистике), несогласованность временной яркости (несохранение яркости в исходном HDR и в результирующем LDR), или появление артефактов (гостинг, в локальных методах призраки немного отличаются, скорее похожи на мерцающие пятна). Гостинг и артефакты возникают из-за того, что локальные методы в принципе менее устойчивы и вероятность того, что что-то поменяется в окрестности больше, чем в самом пикселе. Для того, чтобы решить проблему с мерцанием, можно использовать либо отдельный алгоритм пост-процессинга, либо детектировать мерцание и корректировать индивидуальные параметры, чтобы они стали устойчивы во времени.

Предположим, что происходит съемка синтетической сцены, в которой располагается цветной калибровочный объект и мигающий диод (Рис. 2.21). Для данной

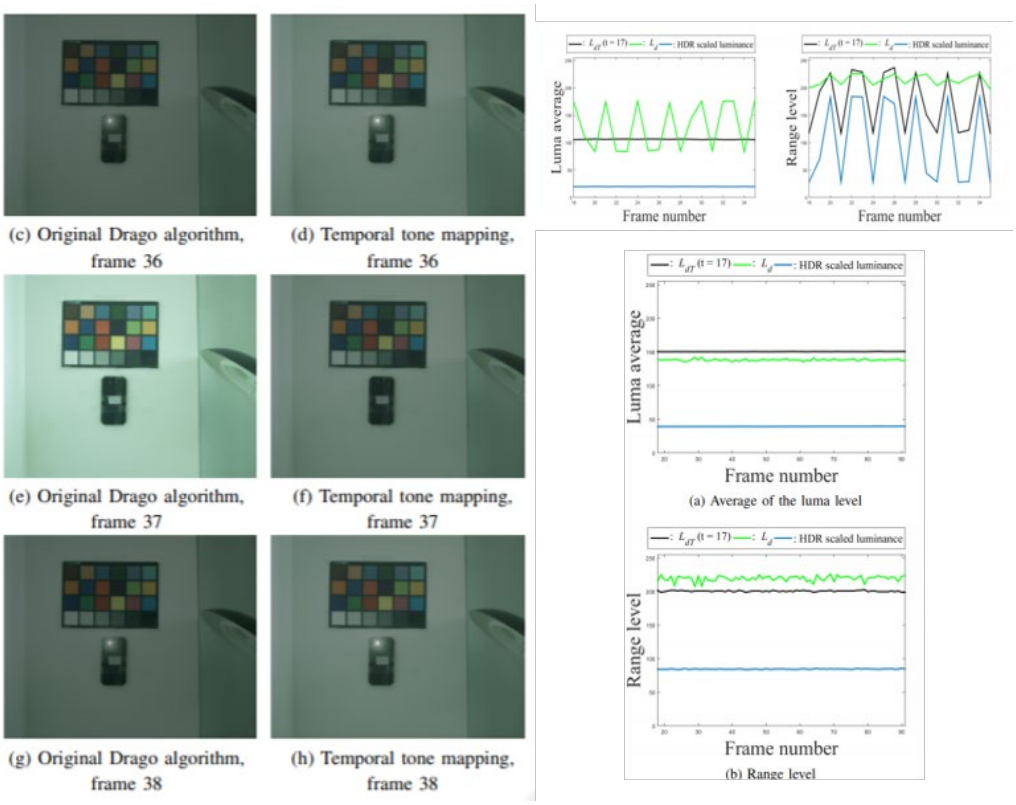

Рисунок 2.21. Применение глобального оператора к временным последовательностям. Слева показаны кадры видео при применении оператора (в левом столбце) и при

временном сглаживании параметров (в правом столбце). На графиках показаны значения яркостей на изображении.

задачи можно использовать вышеописанный глобальный оператор. Если данный алгоритм применяется покадрово, то будет производиться адаптация изображения на мигающий диод, у него все время будет приблизительно одна яркость (зеленая линия на графике вверху, справа). Реальная яркость диода показана зеленой линией на графике вверху, слева. Из-за того, что нормирование производилось на мигающий диод, остальные значения яркости меняются скачкообразно, и после применения алгоритма будет чередование темных и светлых кадров. Чтобы этого происходило, необходимо применять ***временное сглаживание параметров***. В данном случае необходимо смешать среднюю яркость со средними яркостями на предыдущих кадрах. Это приведет к тому, что колебания значительно уменьшится, а статистические части алгоритма станут практически одинаковыми. Это не очень сложное улучшение, но оно сильно влияет на итоговый результат. При написании программы, в которой наблюдатель может перемещаться, необходимо обязательно привязывать значение статистик последующих кадров к значениям статистик на предыдущих, иначе возникает мерцание.

Локальные алгоритмы требуют еще более сложных подходов к устранению мерцания. Самый сложный – ***Temporally Coherent Local Tone Mapping of HDR Video***. В нем используется оптический поток, карты пространственно-временной и временной фильтрации. Другие два более простых алгоритма: ***Tone Mapping for Video Gaming Applications***, который применяется к видеоиграм, он интерактивный, быстрый, в нем используется кодировщик яркости, для которого вычисляется гистограмма и затем настраиваются параметры глобальной кривой с помощью статистики, полученной с гистограммы, и ***VR+HDR***, созданный для приложений виртуальной реальности, в котором наблюдателю в шлеме VR демонстрируется панорама с очень большим диапазоном яркости, наблюдатель может поворачиваться в этой панораме, осматривать ее. Наблюдателю необходимо демонстрировать панораму в узком диапазоне, поэтому предлагается использовать небольшой участок поля зрения в 100˚ для расчета параметров кривой адаптации, и в каждый момент времени показывать панораму, нормированную на 100 ˚. Для того, чтобы не было резких переходов, предварительно рассчитываются некоторые статистики для разных участков этой панорамы и сглаживаются между собой.

Exposure Fusion

Рассмотрим альтернативный подход «как не использовать изображение широкого диапазона, но получить хороший результат». Можно использовать технику ***смешения экспозиции, Exposure Fusion***. При классическом подходе для набора изображений рассчитывается кривая отклика камеры, комбинируется с весовой функцией, получается HDR изображение, применяется алгоритм tone mapping, а затем

это выводится на дисплей. В Exposure Fusion алгоритме используются некоторые методы для вычисления коэффициента смешения изображений, изображения комбинируются с некоторыми «весами» и выводятся на экран. Для того, чтобы рассчитать веса деле используется прием, аналогичный частотным градиентным методам тональной компрессии – выделяются некоторые градиенты, частоты, эвристики для определения насыщенности на данном кадре и вычисляется, на каком кадре этот пиксель представлен лучше. Этот метод дает хорошие результаты. Это является примером того, что эвристические методы могут работать лучше, чем теоретически хорошо продуманные и обоснованные алгоритмы. На практике этот метод используется очень часто.

Восстановление HDR

Рассмотрим обратную задачу - ***восстановление изображения широкого диапазона***. Понятно, что точно восстановить изображение по яркости невозможно, так как большая часть информации была потеряна. Однако, теоретически можно аппроксимировать какие были значения, перед этим было накоплено достаточно знаний: во-первых, за счет того, что количество источников ограничено, во-вторых, за счет того, что источники «переотражаются» в объектах сцены и влияют на их внешний вид, можно попытаться понять, что за источник использовался. Вручную это делать очень сложно, хотя такие алгоритмы также разрабатывались. В последнее время для этого используется ***машинное обучение*** – на вход обучаемой нейросети подается изображение узкого диапазона, она использует кодировщик и затем декодировщик, и получает HDR изображение. Для обучения было взято более тысячи изображений широкого диапазона, для них было искусственно построено почти 47000 изображений узкого диапазона, используя разные кривые отклика камеры (5 типов кривых) и девять значений выдержки.

Лекция 3. Графический процесс

Типовой графический процесс

Большое количество исследователей и разработчиков занимаются поиском способа создания изображений, удовлетворяющим поставленным задачам. Например, создать реалистичное изображение, фотореалистичное изображение, решить задачи научной визуализации и так далее. Это та часть компьютерной графики, которая занимается ***синтезом***. На выходе получается цифровое изображение, показанное в виде матрицы. В каждом пикселе изображения находится цвет, представленный координатами в цветовой системе. С точки зрения психофизиологии, цвет неоднороден, то есть можно выделить следующие его признаки: яркость, насыщенность и оттенок. Оттенок и насыщенность вместе называются ***«хрома»***. Восприятие в целом по хроме линейно, то есть довольно ограничено. С восприятием яркости возникают сложности. Во-первых, яркость нелинейна, во-вторых, диапазон устройств, на которых мы смотрим изображения, существенно уже диапазона, воспринимаемого нашим глазом.

Рисунок 3.1. Графический процесс.

Рассмотрим модель ***типового графического процесса*** (Рис. 3.1).

Весь графический процесс разделён на блоки:

1) ***Модель*** – это тот объект, изображение которого необходимо получить в итоге. В это блок могут включаться геометрическая модель, модель освещения, модель анимации и т.д. Это та информация, которая подаётся на входе в алгоритмы синтеза.
2) ***Синтез*** может включать в себя трассировку лучей, растеризацию, использование различных алгоритмов, и может не зависеть от модели.

3) ***Изображение*** – результат работы синтеза. Как правило, это изображение широкого динамического диапазона.

Как определить, является ли изображение реалистичным? Как правило, если мы можем сказать, что то, что мы видим на изображении, может существовать в реальном мире, то это изображение реалистично. Однако, могут возникать некоторые трудности при восприятии изображения мозгом человека. Во-первых, информация с изображения проходит несколько этапов, перед тем как мозг её обработает. И одним из этих этапов является накопленный человеком опыт, благодаря которому он может сказать, является ли изображение реалистичным или нет. Во-вторых, особенность строения органов зрения. На реальный мир и на искусственно созданное изображение человек смотрит двумя глазами, что вызывает различие в восприятии увиденного.

Какой выбрать подход к синтезу реалистичного изображения? В современном мире можно рассмотреть фотографию как пример наиболее реалистичного изображения. Отсюда возникает понятие ***фотореализма***. Цель создания реалистичного изображения будет достигнута, если получится создать изображение, неотличимое от фотографии. При создании такого изображения стоит ориентироваться на процессы, происходящие в фотоаппарате. Таким образом, процесс моделирования разделяется на две части: моделирование ***распространения света*** и моделирование ***попадания света в линзу фотоаппарата*** до момента появления изображения на матрице. Модель распространения света является ключевой для создания эффекта фотореализма.

Фотореализм включает в себя два элемента:

- Качественная геометрическая модель
- Хорошая физическая модель освещения (отражения, преломления света, тени, текстуры в материале объекта и т.п.)

Модель освещения

Когда мы смотрим на какой-то объект, зачастую наш мозг игнорирует информацию о реальной геометрии этого объекта. Любая геометрия воспринимается глазом как реакция на освещение. На рисунке 3.2 показан пример объекта, который не может существовать в реальном мире. Однако, глаз человека мгновенно пытается уловить трёхмерную форму этого объекта несмотря на то, что эффект объёма достигается путём простейшего двумерного алгоритма с помощью градиента. Такая особенность восприятия заставляет человека быстро и не задумываясь восстанавливать в голове форму объекта по его освещению.

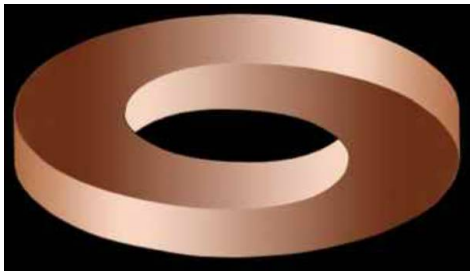

Рисунок 3.2. Имитация объёма плоской фигуры с помощью эффектов освещения.

Ещё одна особенность восприятия – левый и правый глаз получают информацию об изображении по-разному. Но несмотря на отсутствие стереоэффекта в данном изображении, при взгляде на него не возникает никаких сомнений, что он трёхмерный. Это ещё одно доказательство того, что при синтезе реалистичного изображения больше внимания стоит уделять моделированию правильной формы затенения, чем делать упор на геометрию или генерировать стереоизображение.

Моделирование переноса световой энергии

Каждый предмет вокруг нас – это источник света. Разница только в том, является ли он первичным или вторичным. Рассмотрим схему, приведенную на рисунке 3.3. Объект, воспринимающий освещение – ***наблюдатель***. Это кто-то или что-то, у чего есть глаз, линза, любой оптический прибор. Наблюдатель видит какую-то точку объекта, отражающую свет. И всё, что он видит, складывается из множества таких точек. Каждая такая точка представляет собой ***источник света***. Каждый предмет вокруг нас является источником света. Разница только в том, первичный он или вторичный. ***Первичный источник*** – это источник, который испускает свет самостоятельно, благодаря химическим или физическим процессам, происходящим в нём. ***Вторичный источник*** света отражает или преломляет свет, попадающий на него. Если такого света нет, то он чёрный.

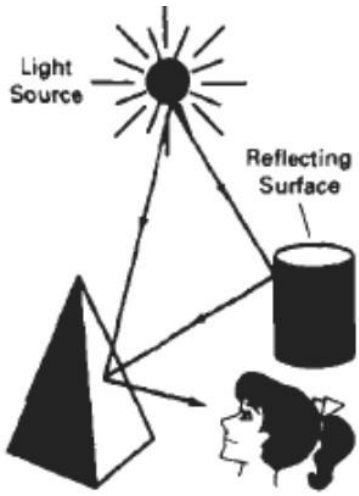

Рисунок 3.3. Отражение света от рассматриваемого объекта.

Окружение точки, испускающей свет в линзу наблюдателя, точно так же будет состоять из многочисленных источников света. Некоторые из этих источников будут первичными (как правило, они самые яркие), другие вторичными. Таким образом, чтобы рассчитать количество света, попадающего в рассматриваемую точку, необходимо суммировать свет, попадающий в неё со всех направлений. Если источник для этой точки первичный, то в этом случае достаточно легко рассчитать количество света. Зная параметры источника, можно узнать, сколько света пойдёт в данном направлении. В случае вторичного источника света расчёт идет с помощью рекурсии. Необходимо вернуться к этому источнику света и рассчитать излучение, которое в неё попадает.

Перейдём к моделированию процесса передачи энергии (Рис. 3.4). Рассмотрим ***точечный источник света*** (источник, у которого нет площади). Для него можно посчитать количество энергии, которое он будет излучать в направлении указанной поверхности. Будем считать, что энергия направлена в конкретную точку этой поверхности. В данном случае нет привязки к конкретным единицам измерения, они необходимы только в тех случаях, когда необходимо рассчитать относительные единицы, либо данные для расчёта берутся из таблиц. Рассматриваемая поверхность определяется ***нормалью N*** и геометрическим расположением относительно источника света.

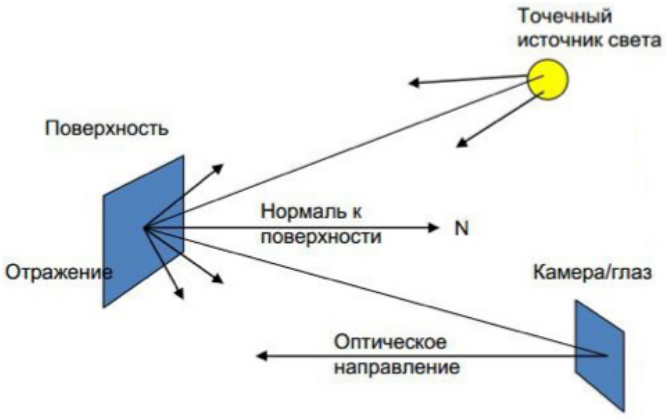

Рисунок 3.4. Процесс передачи световой энергии от источника к наблюдателю.

Третий элемент моделирования – ***приёмник*** (камера или глаз). У приёмника есть оптическое направление, то есть куда он смотрит относительно данного объекта. Направление отражения света от рассматриваемой точки в сторону приёмника и направление света от источника к поверхности определяются относительно нормали N.

В рамках данной лекции стоит задача разработать модель передачи энергии между этими элементами. Необходимо рассмотреть, что происходит, когда свет попадает на какой-либо объект и отражается. Именно благодаря этому процессу человек может видеть окружающие объекты, их цвет, текстуру, прозрачность и т.д.

Задача: ***рассчитать количество энергии, излучаемой в сторону наблюдателя при заданном входящем излучении.***

Двулучевая функция отражения

Рассмотрим схему отражения света с другого ракурса (Рис. 3.5). На схеме можно так же увидеть три основных элемента: источник, точка на отражающей поверхности и приёмник. «Корона» вокруг рассматриваемой точки является стандартным представлением визуализации распределения величины некоторой функции, зависящей от углов. То есть, показывает, сколько энергии при освещении этим источником в какую сторону будет отражено.

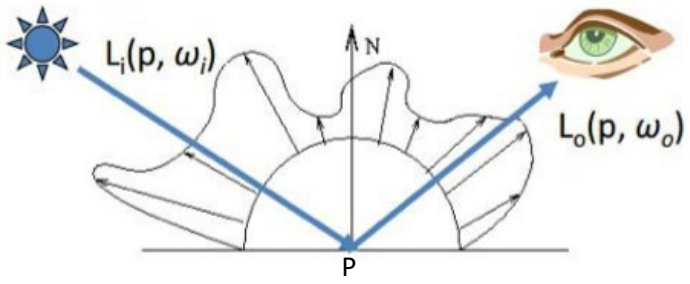

Рисунок 3.5. Схема отражения света в точке P.

L_o – ***исходящее излучение***, зависит от характеристик точки P и направления ω_o. L_i – ***входящее излучение***, зависит от характеристик точки P и направления ω_i. У каждого материала есть своя характеристика, определяющая, сколько энергии будет отражено в интересующем нас направлении при условии облучения точки из этого материала с некоторого заданного направления. Задача моделирования: для данного материала найти или построить функцию, описывающую эту характеристику. Тогда возможно будет вычислить цвет пикселя, соответствующего данной точке объекта.

Эта функция называется ***двулучевая функция отражения, BRDF*** (***Bidirectional Reflection Distribution Function***). Основное фундаментальное требование к этой функции – можно поменять метами источник и приёмник и значение функции не изменится.

Рассмотрим освещённость поверхности в некоторой окрестности точки P (Рис. 3.6). Освещенность поверхности – это плотность светового потока, который создаёт заданный источник на площади заданной поверхности (Вт/м^2). Предположим, что излучение поверхности в векторном направлении будет пропорционально освещённости. Это следует из закона сохранения энергии. Чем сильнее поверхность осветили, тем больше энергии уйдёт в заданном направлении. В этом случае играет роль ориентация поверхности относительно направления источника света.

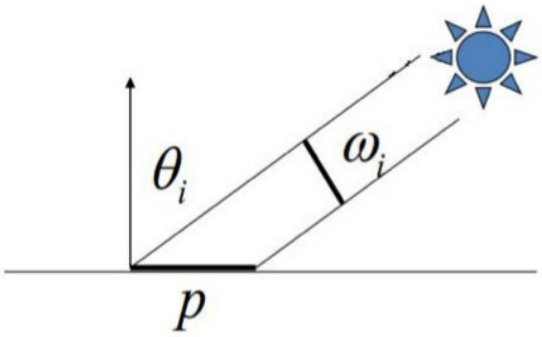

Рисунок 3.6. Освещённость поверхности в окрестности точки P.

Таким образом, освещенность поверхности в точке P для источника, находящегося по направлению ω_i равна энергии, которая передается от этого источника через площадку $d\omega_i$, умноженная на косинус θ, где $cos\theta_i$ – это угол между нормалью поверхности и направлением ω_i:

$$E(p, \omega_i) = L_i(p, \omega_i) cos\theta_i d\omega_i$$

Поток распространяется на таких малых углах, что считается, что он идёт параллельно (Рис. 3.7). И чем сильнее наклоняется площадка, тем на большую площадь этот поток рассеивается и тем меньшую освещённость он создаёт. Поэтому соотношение освещённости по площадке перпендикулярной потоку и площадки, находящейся под углом θ является косинус. Соответственно, если он равен нулю, освещенность получается максимальной. Если значение косинуса получается $\pi/2$, то площадка не освещена вообще.

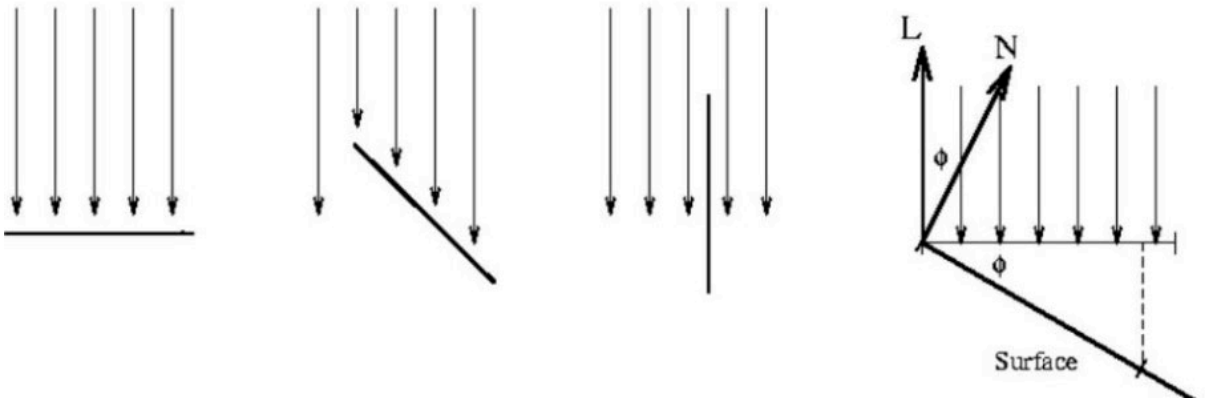

Рисунок 3.7. Степень освещённости в зависимости от угла наклона поверхности.

Таким образом, если предполагается пропорциональная зависимость, то эту зависимость можно представить в виде функции *f*:

$$f(p, \omega_o, \omega_i) = \frac{L_o(p, \omega_o)}{E(p, \omega_i)} = \frac{L_o(p, \omega_o)}{L_i(p, \omega_i) cos\theta_i d\omega_i}$$

Не составляет труда посчитать $L_o(p, \omega_o)$ и $L_i(p, \omega_i)$ и получить табличные значения для различных материалов. Но необходимо учитывать несколько ограничений. Во-первых, подобные расчёты необходимо провести для каждой длины волны. Во-вторых, модель ДФО не предполагает, что мы можем осветить площадку одной длиной волны, а получить отклик на другой длине волны. Можно пойти другим путём и аппроксимировать эту функцию, используя представление о том, как освещённый объект выглядит, либо о том, что влияет на то, как выглядит освещённый объект.

ДФО имеет два основных свойства, которые должны соблюдаться, чтобы не нарушались универсальные законы окружающей реальности. Первое свойство – ***обратимость***. То есть, если поменять местами направления ω, то функция должна сохраниться (Рис. 3.8). Это вывод делается из законов геометрической оптики: луч, проходящий в одном направлении, должен пройти обратно в том же направлении.

$$\forall \omega_o, \omega_i \; f_r(p, \omega_o, \omega_i) \; = \; f_r(p, \omega_i, \omega_o)$$

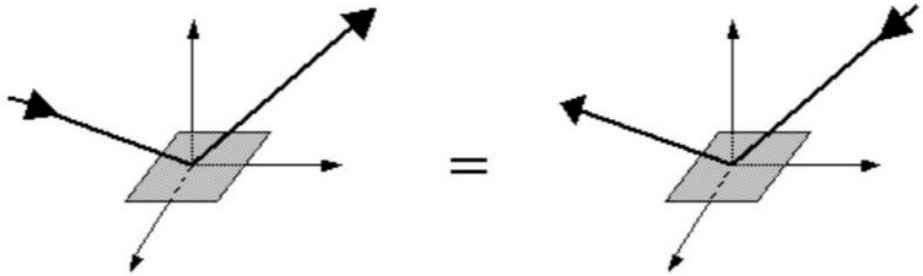

Рисунок 3.8. Обратимость светового луча.

Второе свойство – ***сохранение энергии.*** Если входной луч имеет интенсивность 1, то сумма энергий всех исходящих лучей должна быть меньше или равна 1 (Рис. 3.9)

$$\int f_r(p, \omega_i, \omega') cos\theta' d\omega' \leq 1$$

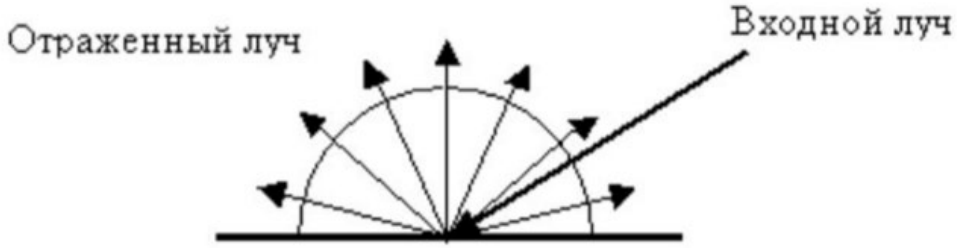

Рисунок 3.9. Сохранение энергии при отражении.

При наличии ДФО и представления о том, как с её помощью вычислить L_o задача ***расчёта яркости точки*** не имеет затруднений:

$$f_r = \frac{L}{E}$$

$$L_o = f_r E = f_r L_i cos\theta_i d\omega_i$$

Расчёт излучения точки поверхности через интегрирование по всем входящим направлениям

Полное излучение из точки P в направлении ω_o равно сумме всех излучений, которые в этом направлении появились из-за освещения источниками, находящимися в направлении ω_i:

$L_o(p, \omega_o) = \int L_{oduetoi}(p, \omega_o, \omega_i)$

где $f_r(p, \omega_o, \omega_i) = \frac{L_o(p,\omega_o)}{L_i(p,\omega_i) cos\theta_i d\omega_i}$

$$L_o(p, \omega_o) = \int_{\Omega} f_r(p, c, \omega_i) L_i cos\theta_i d\omega_i$$

Рассмотренная модель справедлива почти во всех случаях моделирования макромира, за исключением случаев моделирования сложных эффектов, например, дифракции или дисперсии.

Полученное уравнение иногда называют ***основным уравнением компьютерной графики*** в простейшем виде. Оно показывает, что излучение в точке P в направлении ω_o зависит от интеграла Ω. Это сфера, в которой по всем направлениям $d\omega_i$ берется входящее излучение и с учётом косинуса находится любое излучение, формирующее впоследствии цвет изображения в матрице. При расчёте может возникнуть сложность в том, что наличие L в обоих частях уравнения может привести к громоздкому решению большой системы линейных и нелинейных уравнений.

Расчёт излучения точки поверхности: дискретный случай

Для упрощения расчёта удобнее использовать локальную модель. В локальной модели источники света дискретны и их конечное число. В этом случае интеграл превращается в сумму:

$$L_o(p, \omega_o) = \int_{\Omega} f_r(p, c, \omega_i) L_i cos\theta_i d\omega_i$$

$$L_o(p, \omega_o) = \sum_{j=0}^{n-1} f_r\left(p, c, \omega_i^j\right) L_i^j cos\theta_i^j$$

где ω_i^j – направление на j-тый источник света, θ_i^j – угол между направлением на j-тый источник света и нормалью к поверхности. То есть при расчёте игнорируется вторичное освещение. Суммирование происходит только по первичным источникам от

1 до n. Но в этом случае изображение будет не очень реалистичным, поскольку большая часть видимого освещения идет от вторичных источников.

При попытке систематизировать видимые объекты по степени отражений и поведению их материалов при взаимодействии со светом, можно выделить следующую классификацию: ***диффузное отражение, идеальное зеркальное отражение, зеркальное отражение***.

На одном краю спектра находится ***диффузные материалы***. Они не имеют бликов и формально определяются как тип материала, ***равномерно рассеивающий свет во все стороны*** (матовый пластик, дерево, бумага и т.д., Рис. 3.10). То есть это означает, что для данного источника света и данного материала для него нет зависимости отражения от того, откуда смотрит наблюдатель.

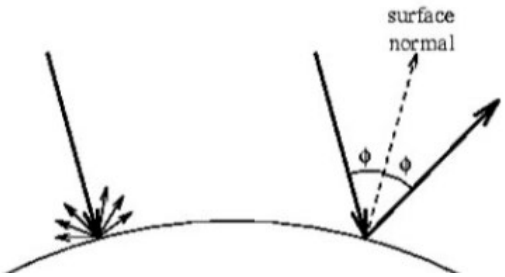

Рисунок 3.10. Отражение света в диффузных и зеркальных материалах.

Модель Ламберта

В реальности не существует идеального диффузного материала. В любом материале присутствует хотя бы минимальная зеркальная составляющая. Примером идеальной диффузной поверхности является ***модель Ламберта***.

Ламбертова поверхность выглядит одинаково ярко со всех направлений (Рис. 3.11). В природе её не существует, но есть близкие приближения (например, бумага).

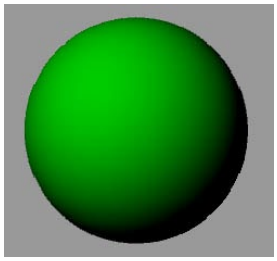

Рисунок 3.11. Ламбертова поверхность.

Модель Лаберта учитывает только идеальное рассеивание света. То есть ДФО для этой модели является константой. ДФО влияет на общую яркость (сколько оъект поглощает и сколько объект отражает). Если ДФО равна 0, то получается абсолютно чёрный объект. Расчёт яркости в итересующем направлении для одного источника свет выглядит следующим образом:

$$L_o = L_i k_d (\vec{s} \cdot \vec{n})$$

где $\vec{s} \cdot \vec{n}$ является $cos\omega_i$, $k_d = C \,/\, Pi$ – диффузный коэффициент (C определяет процент отражения для днной длины волны), $\vec{s}$ – направление на источник света, $\vec{n}$ – нормаль к поверхности (Рис. 3.12).

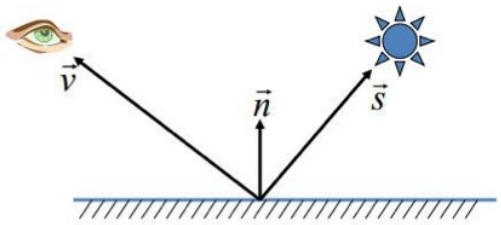

Рисунок 3.12. Схема для расчёта яркости в заданном направлении.

Идеальное зеркальное отражение

На другом конце спектра находится ***зеркальное отражение***. При идеальном зеркальном отражении энергия не теряется. Она не тратится на нагревание, преломление, не рассеивается. ДФО в этом случае равно нулю везде, кроме направлений $\theta_o = \theta_i$ и $\phi_o = \phi_i + \pi$ (Рис. 3.13).

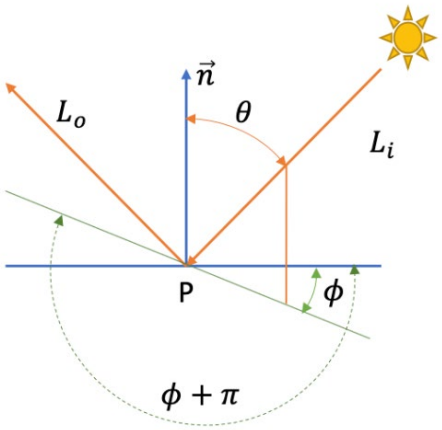

Рисунок 3.13. Схема зеркального отражения.

Тогда ДФО зеркального отражения примет вид дельта-функции:

$$f_{rmirror} = \frac{\delta(\theta_i - \theta_o)\delta(\phi_i - \phi_o)}{\cos(\theta_i)}$$

Результатом такого расчёта является идеальная зеркальная поверхность. Такой тип отражения практически не втречается в реальном мире. Максимально похожие характеристики отражения имеет плоское зеркало. Однако существует тип отражения, сочетающий в себе признаки диффузной и абсолютно зеркальной поверхности. Такой тип отражения называется ***зеркальным***.

Модель Фонга

Большинство объектов, человек видит в окружающем мире имеют и матовую, и зеркальную составляющую. ***Модель Фонга*** является одной из классических для описания бликующих объектов (Рис. 3.14). Она *добавляет зеркальные блики в модель Ламберта*. Практически все объекты, смоделированные с помощью модели Фонга имитируют пластиковую поверхность, и добиться схожести с другими материалами в этом случае представляет затруднения. Данная модель не имеет под собой физической основы, она базируется на эмпирических наблюдениях. Чаще всего она используется в приложениях, не имеющих требований к фотореализму.

ДФО для Фонга выглядит следующим образом:

$$f_{rphong} = \frac{k_d(s \cdot n) + k_s(r \cdot v)^k}{cos\theta_i}$$

$k_d(s \cdot n)$ – диффузная составляющая, $k_s(r \cdot v)^k$ – зеркальная составляющая. С помощь коэффициентов k можно регулировать интенсивности диффузной и зеркальной составляющих.

Полный расчёт с помощью модели Фонга:

$$L_o = k_a L_a + L_i(k_d(\vec{s} \cdot \vec{n}) + k_s(\vec{r} \cdot \vec{v})^{k_e})$$

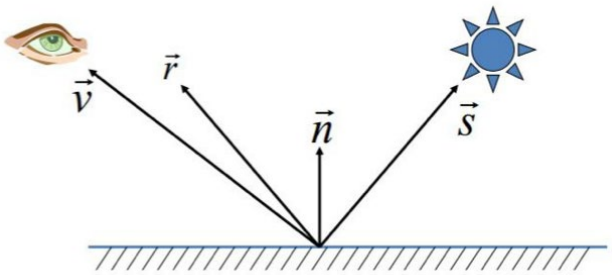

Рисунок 3.14. Схема отражения для модели Фонга.

$k_a L_a$ – константа, составляющая фонового освещения. Она нужна для того, чтобы неосвещённые области объекта не выглядели абсолютно чёрными. На рисунке 3.15 представлен пример модели Фонга.

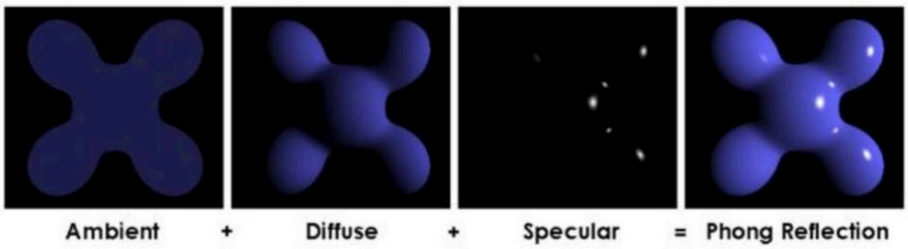

Рисунок 3.15. Пример применения модели Фонга.

Проблема модели Фонга заключается в том, что она не является физически корректной. Общая энергия сцены не совпадет с энергией, расчитанной с помощью модели Фонга - *нарушается закон сохранения энергии*. Так же она *не является обратимой*. Несмотря на это она очень широко применяется.

Оправдано ли разделение на зеркальную и диффузную составляющие? Существуют модели, основанные на представлении поверхности объекта набором микрочастиц. Это соответствует физическим процессам. Зеркальная поверхность состоит из гладких частиц, ориентированных в одном направлении. Частицы диффузной поверхности ориентированы в случайных направлениях, поэтому свет, отраженный от этих частиц, рассеивается в разные стороны. На практике не моделируется каждая частица по-отдельности, а используется статистические распределения для моделирования сложной системы.

Микрофасетная модель

В случае ***микрофасетного моделирования*** поверхность представляется набором оптически плоских частиц, при этом каждая частица может иметь свою ориентацию. Можно заметить, что свет от этой частицы попадет в глаз наблюдателя только тогда, когда нормаль расположена на полпути между направлением источника света и направлением наблюдателя, поскольку угол падения равен углу отражения. В этом случае в графике вместо нормали используется так называемый ***half-вектор*** (вектор половинного угла). Он часто фигурирует в моделях вместо нормали (Рис. 3.16). То есть, чем больше частиц ориентировано таким образом, что угол отражения соответствует направлению взгляда наблюдателя, тем более яркая получится поверхность.

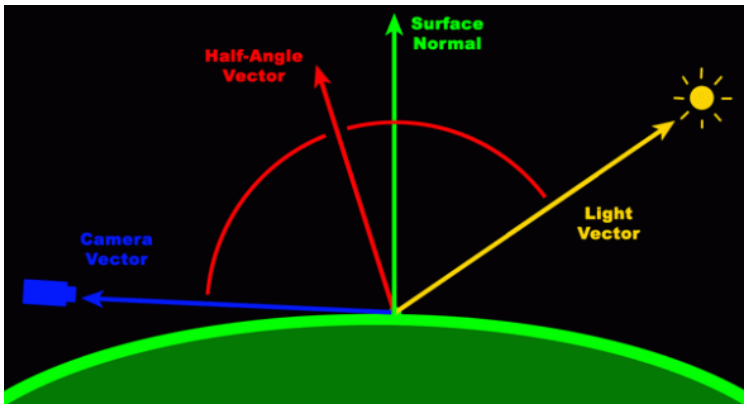

Рисунок 3.16. Half-вектор при микрофасетном моделировании.

Следующие две характеристики – это ***затенение*** и ***маскирование***. Всегда на поверхности имеются частицы, на которые свет может либо не попасть (затенение), либо может отразиться от неровности и не достигнуть наблюдателя (маскирование, Рис. 3.17).

Рисунок 3.17. Схема затенения и маскирования.

Микрофасетное ДФО выглядит следующим образом:

$$f(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{l},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})D(\mathbf{h})}{4(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})}$$

Функция F, G и D описывают поведение микрофасет для данного материала. Функция F – это ***коэффициент френелевского отражения***. Это базовый коэффициент, который задаётся либо спектрально, либо в RGB, и он задает для зеркала процент света, который будет отражён для данного угла между наблюдателем и падающим светом. Он зависит от индекса рефракции. Обычно, когда человек смотрим на любой объект перпендикулярно, он видит отражение базового цвета. При наблюдении под углом зеркальность объекта резко увеличивается. Отсутстсвие отражения при наблюдении под прямым углом – это важная характеристика поверхности. Обычно она определяет её ***базовое отражение*** или ***базовый цвет***. Параметр D – это ***распределение микрофасетов в зависимости от их нормали***. Самый частоиспользуемый параметр для этой модели – шероховатость. Чем он больше, тем более диффузной становится поверхность (Рис. 3.18). Параметр G – геометрическая компонента, задаёт вероятность от 0 до 1, что микрофасета для этих направлений была затенена или маскирована.

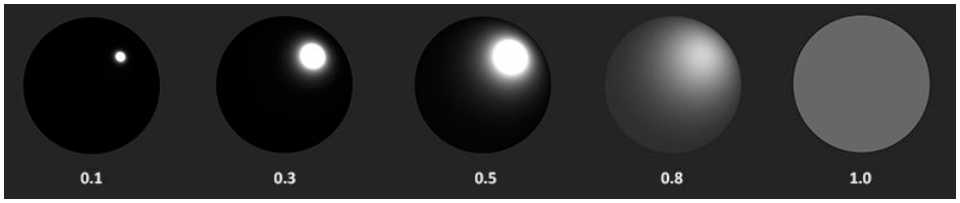

Рисунок 3.18. Влияние значения шероховатости на диффузность поверхности.

Для микрофасетной модели имеется либо два, либо три параметра. Основной параметр – $F(0°) = c_{spec}: (RGB)$ – ***базовый цвет***. Это тот цвет, который наблюдатель увидет, посмотрев на поверхность ровно перпендикулярно. И один или два параметра, связанных с шероховатостью поверхности и определяющих блик. Количество параметров зависит от того, изотропный или анизотропный рассматриваемый материал.

Кроме того, во всех этих моделях варьируется эти параметры по поверхности модели. То есть задаётся текстура. Обычно это две текстуры. Одна из них варьирует базовый цвет. Вторая варьирует шероховатость и уровень того, насколько этот материал обладает свойствами «металлика». Варьируя эти параметры можно получить фотреалистичные изображения.

Аппроксимация Шлика

Данную аппроксимацию удобно использовать, чтобы избежать использования табличных значений или более сложных аппроксимаций. Аппроксимация для параметра c_{spec}:

$$F_{Schlick}(c_{spec}, l, n) = c_{spec} + (1 - c_{spec})(1 - (l \cdot n))^5$$

Для микрофасетной BRDFs (где n = h):

$$F_{Schlick}(c_{spec}, l, h) = c_{spec} + (1 - c_{spec})(1 - (l \cdot h))^5$$

Полученную $F_{Schlick}$ можно подставить в микрофасетную ДФО и иметь на входе параметр $F(0°)$, который можно задавать снаружи модели.

Лекция 4. Трассировка лучей.

Алгоритм трассировки лучей

Алгоритм трассировки лучей заключается в том, что луч пропускается через каждый пиксель и происходит поиск пересечений с ближайшим объектом. Самый простой способ найти пересечение – циклом перебрать все объекты в сцене и найти ближайший из них. После этого необходимо понять, находится ли точка в тени или не свету. Нужно пустить лучи в источники света (считаем, что источники света точечные) и проверить есть ли пересечения. Если их нет – точка на свету, если они есть, и находятся между точкой и источником (т.е. расстояние от этой точки до объекта меньше, чем расстояние от точки до источника), то пересечение произошло в тени. Для всех источников, не имеющих тени, освещенность складывается. Освещенность считается по локальной формуле (например, одна из упомянутых в Лекции 3: модель Фонга, микрофасетная модель или модель Ламберта). Имеет смысл считать освещенность только для тех объектов, которые имеют шероховатую форму, а не для зеркал или стекол. На рисунке 4.1 представлен пример объекта с бликом, материал которого имеет смешанные свойства, сочетающие модель Фонга и Ламберта, способные и к зеркальному отражению без размытия.

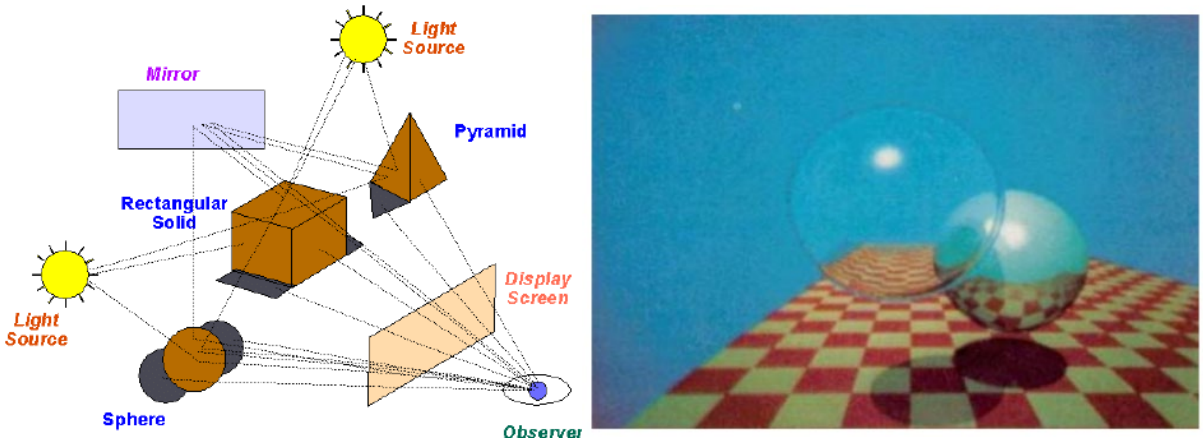

Рисунок 4.40. Алгоритм трассировки лучей. Whitted Ray Tracing.

Далее необходимо пустить отраженные лучи (в самом простом случае угол падения равен углу отражения, в более сложных случаях луч преломленный). Для стекла может быть и отраженный, и преломленный луч. Для расчетов в случае стекла существуют формулы Френеля (Рис. 4.2), показывающие соотношение пропущенной энергии (красным на графике) к количеству отраженной.

Генерация луча

Существует простой способ сгенерировать луч, руководствуясь геометрическими соображениями. Есть точка, из которой луч выходит (позиция камеры) и экран с некоторым разрешением (ширина на высоту). Необходимо

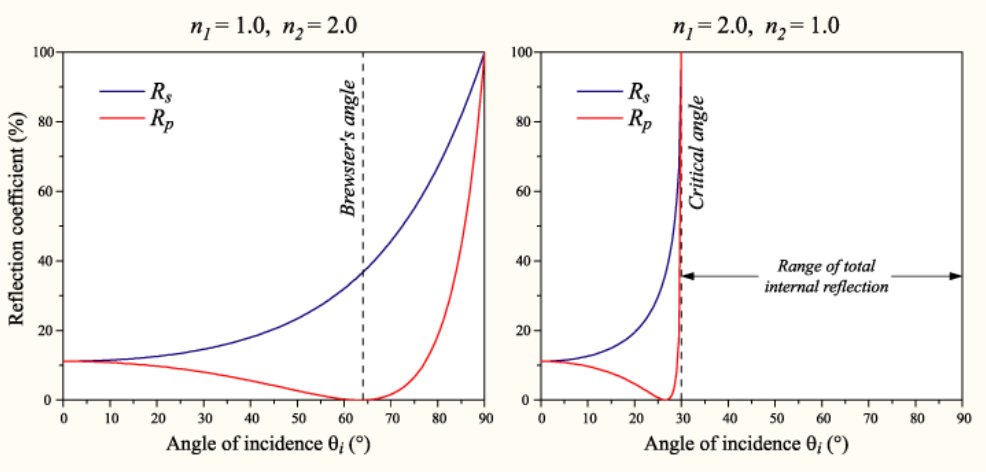

Рисунок 4.41. Формулы Френеля.

нарисовать прямоугольный треугольник. Затем нужно подставить значения x и y – это координаты пикселя, а z можно найти через тангенс. После этого вектор нужно нормализовать, чтобы получить направление луча. Этот несложный метод позволяет генерировать лучи и пускать их в сцену. Однако есть нюанс – если необходимо сделать трассировку лучей, которая будет совпадать геометрически с изображением, получаемым при растеризации в какой-либо системе, то поскольку растеризация практически во всех системах работает в терминах матриц, модельно-видовой матрицы и матрицы проекции. С этой функцией не получится добиться нужного результата. В этом случае нужно взять матрицы, которые подаются на вход системе (модельно-видовую матрицу и матрицу проекции), посчитать обратную матрицу, и применить ее к лучу. Это важно, потому что, когда некоторая матрица применяется к объекту, для того чтобы получить тот же самый результат к лучу, нужно применить обратную матрицу. Таким образом, луч из пространства камеры при помощи обратной матрицы переносится в мировое пространство, а модельно-видовая матрица, наоборот, мир из мирового пространства переносит в пространство камеры.

```
float3 EyeRayDir(float x, float y, float w, float h)
(
        float fov = 3.141592654f/ (2.0f) ;
        float3 ray_dir ;
        ray_dir.x = x+0.5f – (w/2.0f) ;
        ray_dir.y = y+0.5f – (h/2.0f) ;
        ray_dir.z = -(w)/tan(fov/2.0f) ;
        return normalize(ray_dir) ;
)
float3 EyeRayDir2(float x, float y, float w, float h)
```

```
(
        float4 pos( 2.0f * (x + 0.5f) / w – 1.0f,
                    -2.0f * (y + 0.5f) / h + 1.0f, 0.0f, 1.0f ) ;
        pos = g_nViewProjInv*pos; // (mView*mProj)-1*pos
        pos /= pos.w;
        pos.y *= (-1.0f) ;
        return normalize(pos.xyz) ;
)
```

Трассировка лучей и растеризация

В визуализации трехмерных объектов существует два самых базовых подхода: трассировка лучей и растеризация. При трассировке через каждый пиксель пускается луч и находится пересечение. Однако существует и обратный подход, когда объект проецируют на экраны при помощи матрицы проекции. Плюс растеризации заключается в том, что не нужно для каждого пикселя выполнять множество вычислений, как в случае трассировки лучей, а достаточно спроектировать только точки вершины треугольников. При этом все, что находится внутри треугольника, линейно интерполируется - все атрибуты или в простейшем случае цвет. Растеризация существенно дешевле, чем трассировка лучей, сейчас этот подход распространен, он аппаратно ускорен, в том числе на смартфонах.

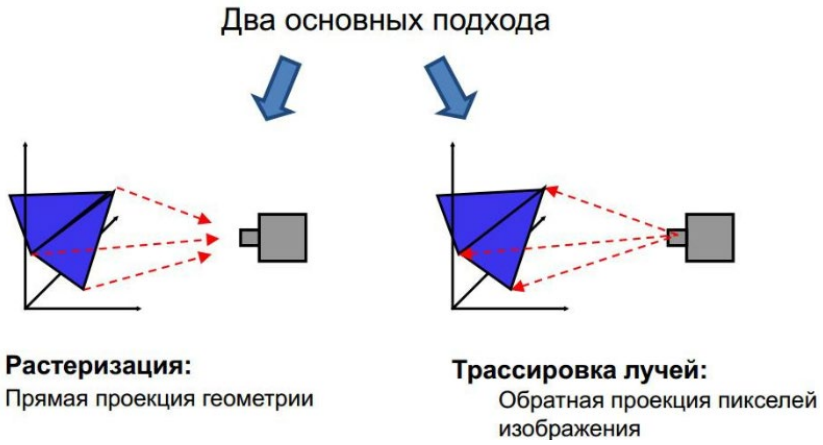

Рисунок 4.42. Растеризация и трассировка.

Оба подхода имеют достоинства и недостатки. В случае трассировки лучей можно реализовать практически любой аналитический объект, который задается формулой пересечений луча этого объекта. В свою очередь при растеризации необходимо этот объект переводить в полигональную форму – генерировать треугольную сетку (ее можно не сохранять в памяти, она может генерироваться на чипе в шейдерах). Поэтому на современных шейдерах подход с растеризацией выглядит

более перспективным. Для трассировки лучей тоже существует ускорение с помощью аппаратного конвейера, например, в случае последних карт Nvidia.

Между двумя подходами много общего. Если геометрия треугольная, то по определенному соглашению, которое есть у производителей компьютеров и всех API, при растеризации треугольника вопрос о том, закрашивать пиксель или нет, решается следующим образом – пиксель считается перекрытым, если центр пикселя находится внутри треугольника (Рис. 4.4). На иллюстрации видно, что некоторые пиксели не закрашены, хотя они не перекрываются треугольником геометрически. По сути это то же самое, если бы лучи выпускали через центр пикселей и искали пересечения. Таким образом, задача растеризации в данном случае ковалентна задаче трассировки лучей.

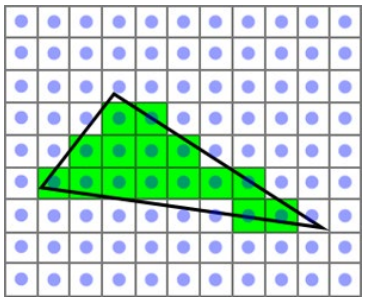

Рисунок 4.43. Правила растеризации треугольника.

Пересечение луча и треугольника

Как определить лежит ли центр пикселя внутри треугольника или снаружи? Если речь идет о 2D пространстве, то имеются координаты трех вершин и точка, центр пикселя. В графической аппаратуре это называется ***Half Space тест*** или ***тест полупространства***. Если задан некоторый порядок обхода треугольника, то в любой точке пространства можно посчитать знаковое расстояние от границ треугольника. В случае, если все расстояния больше 0, то точка находится внутри, а если величина хотя бы одного расстояния отрицательна, то точка снаружи.

Другой способ, который используется достаточно редко: положим, речь идет о трехмерном пространстве. Искомая точка и две вершины треугольника образуют три новых треугольника с площадью S_1, S_2, S_3. Если точка находится внутри треугольника, то сумма площадей новых треугольников будет равна площади исходного треугольника.

Можно получить соотношения площадей этих треугольников к площади большого, что является барицентрическими координатами (S_1/S, S_2/S, S_3/S). Сумма барицентрических координат должна быть равна 1, что позволяет рассматривать значения координат как некоторые веса (U, V, T1). Как правило, в моделях в вершинах задаются какие-то свойства: нормаль, текстурные координаты и др. Когда луч попадает в какую-то точку в треугольнике (не в одну из вершин), необходимо получить значения

трибутов на поверхности, используя барицентрическую интерполяцию (т. е. просто умножая каждую величину вершины на барицентрическую координату противолежащего треугольника и суммируя полученные значения). Этот способ достаточно популярен. Как считать пересечение лучей с треугольником в 3D (Рис. 4.5):

$u := u/S$ $\qquad$ $z(u, v) = (1 - u - v)*v1 + u*v2 + v*v0$

$v := v/S$ $\qquad$ $z(t) = p + t*d$

$t1 := t1/S$ (или $t1 = 1 - u - v$) $\qquad$ $p + t*d = (1 - u - v)*v1 + u*v2 + v*v0$

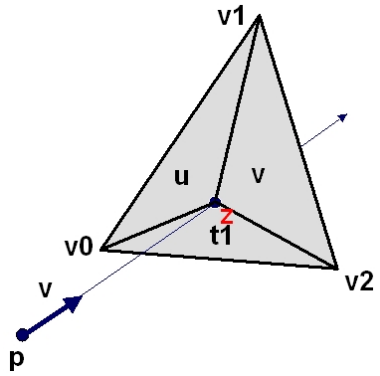

Рисунок 4.44. Пересечение луча и треугольника.

Решая данное уравнение в аналитическом виде, получаем следующий результат:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{dot(P, E1)} * \begin{bmatrix} dot(Q, E2) \\ dot(P, T) \\ dot(Q, D) \end{bmatrix}$$

$$E1 = v1 - v0$$
$$E2 = v2 - v0$$
$$T = p - v0$$
$$P = cross(D, E2)$$
$$Q = cross(T, E1)$$
$$D = v$$

Другим популярным тестом является ***Unit-тест***. Для него необходимо перейти в такое пространство, где будет проще считать пересечение луча и треугольника (или любой другой фигуры с пространственной ориентацией) при помощи матрицы:

$$T_{\Delta}^{-1}(X) = \begin{pmatrix} A_x - C_x & B_x - C_x & N_x - C_x \\ A_y - C_y & B_y - C_y & N_y - C_y \\ A_z - C_z & B_z - C_z & N_z - C_z \end{pmatrix} \cdot X + \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix}$$

Если применить данное преобразование к треугольнику (1,0,0); (0,1,0); (0,0,0); то получим исходный треугольник:

$$T_\Delta^{-1}\begin{pmatrix}1\\0\\0\end{pmatrix}=A \quad T_\Delta^{-1}\begin{pmatrix}0\\1\\0\end{pmatrix}=B$$

$$T_\Delta^{-1}\begin{pmatrix}0\\0\\0\end{pmatrix}=C \quad T_\Delta^{-1}\begin{pmatrix}0\\0\\1\end{pmatrix}=N$$

Для того чтобы найти нужное преобразование необходимо дополнить матрицу TΔ-1 до 4x4 (добавить еще одну строчку (0,0,0,1)) и найти обратную матрицу. Это будет соответствовать обратному TΔ-1 преобразованию. После, необходимо умножить луч на полученную матрицу и посчитать пересечение с единичным треугольником:

```
float3 o = mul3x4(m, ray_origin);
float3 d = mul3x3(m, ray_direction);
float t = -o.z/d.z;
float u = o.x + t*d.x;
float v = o.y + t*d.y;
```

Функция mul3x4 выполняет умножение подматрицы 3x3 на трехмерный вектор и добавляет к результату последний столбец (3 его компоненты). Функция mul3x3 просто умножает подматрицу 3x3 на трехмерный вектор.

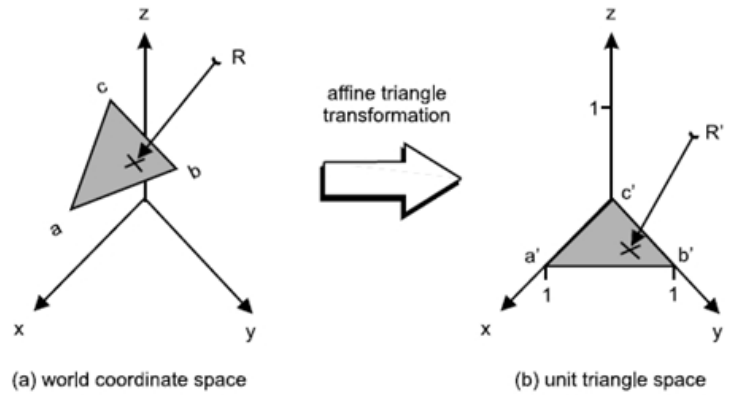

Рисунок 4.45. Афинное преобразование треугольника в пространство, где он имеет единичное представление.

После того, как было найдено пересечение, и обнаружен источник (вклад которого необходимо учитывать, например, по локальной модели Фонга), почти всегда встречается некий нормирующий ***множитель*** $\frac{1}{R^2}$ (т. е. у источника света появляется некое затухание). Откуда возникает затухание при растеризации или трассировке?

Почему при расчете затенения (shading) используется квадрат расстояния до источника света? Это не влияние рассеяния в воздухе, т.к. даже в вакууме все равно использовался бы именно такой множитель. Так происходит, потому что только часть от всей энергии, которая выходит из источника, доходит до искомой точки. Чем точка дальше от источника, тем меньше энергии она получает, что пропорционально обратному квадрату расстояния. Если сделать протяженные источники света, то возникает эффект ***мягких теней***. Для этого источник разбивается на множество меньших источников с помощью регулярной сетки. Однако в этом методе возможно возникновение артефактов – шума. Другая проблема, с которой можно столкнуться при трассировке лучей – ***проблема точности*** (***проблема повторных пересечений***). В этом случае после того, как пересечение с поверхностью найдено, пущен отраженный луч, снова обнаруживается пересечение с той же поверхностью (Рис. 4.7, слева).

Функция расстояния

Для сложных и абстрактных поверхностей используется ***функция расстояния*** (расстояние от некоторой точки в пространстве до поверхности) D = f (x, y, z). В этом алгоритме считается расстояние от точки в пространстве до поверхности. Размер следующего шага определяется этим расстоянием (Рис. 4.7, справа). На практике этот метод не имеет широкого применения, так как не для всех поверхностей такую формулу можно вычислить аналитически.

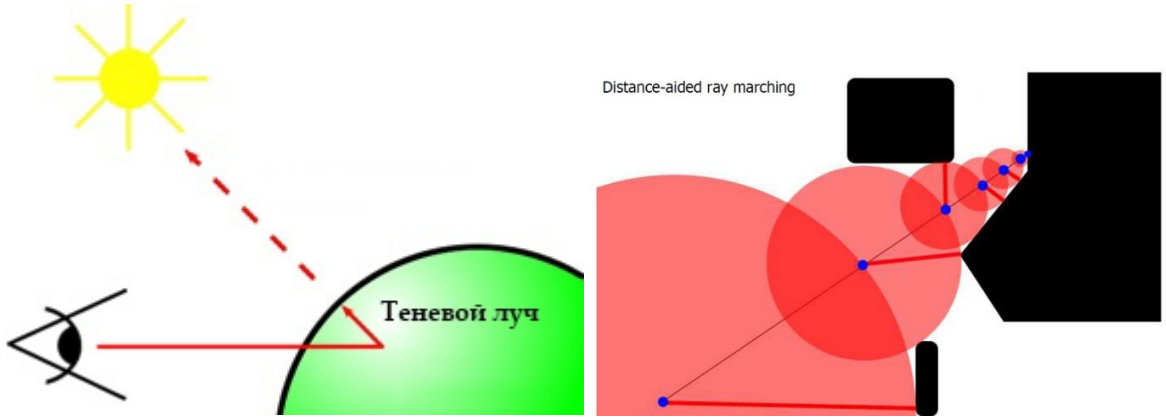

Рисунок 4.46. Трассировка лучей. Проблема точности (слева). Алгоритм Distance-aided ray marching (справа).

На практике часто встречаются ***сплайны и поверхности Безье***. Для этих поверхностей на самом деле такой формулы в аналитическом виде не существует. Чем отличаются поверхности сплайнов от поверхностей Безье (или просто их кривые)? Кривые и поверхности Безье отличаются тем, что они не обязаны проходить через контрольные точки, они только к ним приближаются (часто используются для реализации задач художников). В свою очередь, сплайны обязательно проходят через все заданные контрольные точки.

Ускоряющие структуры

В трассировке лучей возникает сложность, когда имеется большое количество примитивов (треугольников). В любой современной компьютерной игре в кадре десятки миллионов полигонов, что серьезно усложняет подсчет. Для этого существуют ***ускоряющие структуры***. Один из таких способов – разбить пространство на ***регулярную сетку*** (кубики), записать какие объекты пересекают границы кубиков, а дальше перебрать те объекты, на которые ссылаются кубики. Регулярная сетка занимает много памяти, она не адаптивна (т. е. это подойдет если сцена игрушечная, а если это стадион, на котором множество движущихся объектов, не понятно, какой размер сетки выбрать – маленькая поглотит много памяти, а крупная не даст ускорения). Помимо этого, существует проблема повторных пересечений, например, если примитивы разного размера, то с некоторыми из них пересечения посчитаются множество раз. Этот способ один из первых, он прост и примитивен. Алгоритм состоит из двух частей: инициализирующая часть (при попадании в сетку) и часть, которая осуществляет перебор:

```
if(tMaxX <= tMaxY && tMaxX <= tMaxZ)
{
   tMaxX += tDeltaX;
   x += stepX;
}
else if(tMaxY <= tMaxZ && tMaxY <= tMaxX)
{
   tMaxY += tDeltaY;
   y += stepY;
}
else
{
   tMaxZ += tDeltaZ;
   z += stepZ;
}
```

Для того, чтобы избежать недостатков метода, можно использовать ***иерархическую сетку***, но это не сильно улучшит результат, так как дешевым является именно алгоритм обхода внутри сетки, а алгоритм перехода между уровнями сетки довольно дорогостоящий.

Другим простым вариантом ускоряющих структур является ***kd-tree***. Сцену разбивают плоскостью на две части, возникают два узла – A и B. Затем части делятся снова и получаются узлы AA, AB, BA и BB. Подобный механизм описан на рис. 4.8. Существуют несколько способов выбора плоскости разделения. Простейший – по центру. Другой способ – по медиане (по центру области с большим числом объектов), так называемое «***сбалансированное дерево***». Сбалансированное дерево подходит для

тех случаев, когда распределение объектов достаточно равномерно. Третий способ – на основе оптимизации стоимости – самый оптимальный.

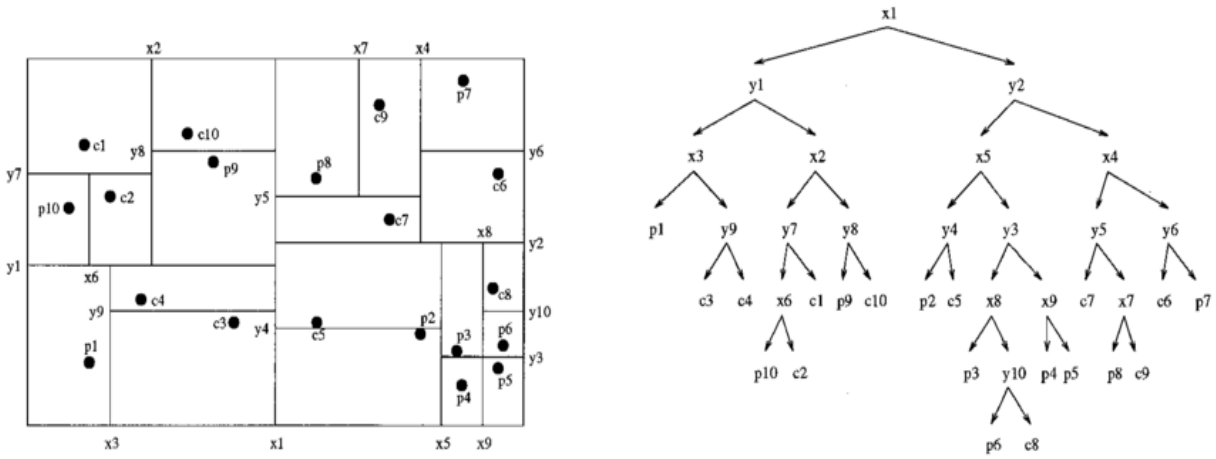

Рисунок 4.47. Механизм построения kd-tree.

В данном случае алгоритм пресечения очень простой. Есть плоскость, которая разбивает пространство неким образом и три случая пересечения: пересечен правыйузел, левый узел или оба (Рис. 4.9). В случае пересечения одного из узлов сложностей не возникает, а при пересечении обоих луч попадает в ближайший узел, а дальний помещается в стек в случае не рекурсивного алгоритма.

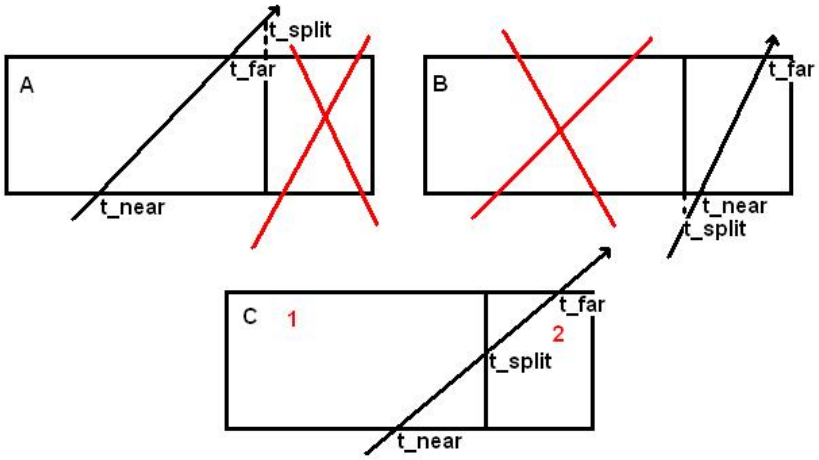

Рисунок 4.9. Варианты пересечения луча с узлами.

Структура kd-tree очень компактна, она может иметь вес всего 8 байт. Однако, вместе с этим есть недостатки. Во-первых, не любая геометрия может быть разбита таким образом (как ни провести плоскость, и в правом, и в левом узле по-прежнему столько же примитивов, сколько было в узле). В этом случае рекомендуется проводить плоскости не параллельно осям координат, а произвольно (так называемый ***классический BSP***). Классический BSP теоретически более эффективен, но на деле

построение его дерева очень трудное из-за того, что нужно определять не только позицию, но и ориентацию плоскости.

В современности лидирует другой подход – ***BVH-дерево*** (***Bounding Volume Hierarchy***). В этом случае имеется множество отсортированных объектов. Например, массив объектов разбивается на два подмассива, а затем вокруг этих подмножеств пересчитывается bounding box. Для эффективности метода необходимо сортировать объекты по некоторой оси координат и рассматривать все возможные разбиения на подмножества таким образом, будто они разбиты плоскостью. В рамках метода нужно подсчитать количество примитивов, которое попадает в левый и правый бокс, на площадь бокса и искать оптимальное разбиение. Затем такая процедура повторяется рекурсивно.

Принципиальное отличие от предыдущих способов заключается в том, что в них разбивается пространство при помощи плоскости, а в случае BVH-дерева разбивается множество. В методе kd-tree, когда плоскость рассекает геометрию, возникает ситуация, когда примитвы попадают на плоскость сечения, в результате чего один и тот же треугольник может попасть в левый и правый узел. Из-за этого нельзя предсказать, сколько памяти потратится на процедуру. В случае BVH-дерева всегда разбивается массив на меньшие подмножества и с каждым следующим уровнем дерева сложность задачи уменьшается. Проблема метода заключается в том, что узлы в нем имеют пересечения. В этом случае необходимо проверять каждый узел из стека на предмет пересечений. Также на поверхности может быть огромный примитив, например, вытянутый треугольник, из-за которого при любом варианте разбиения массива не будет происходить сортировки. Эта проблема решается тем, что большой примитив представляют как набор маленьких примитивов.

Подход BVH лидирует еще и по той причине, что kd-деревья очень неэффективны для современных компьютеров и особенно для центральных процессоров из-за того, что механизм очень итеративен и выбор нового узла для перехода не осуществляется до тех пор, пока не были проведены расчеты.

Интеграл освещенности и метод Монте-Карло

Для того, чтобы получить реалистичное качественное изображение, необходимо показать микрорельеф, от которого свет не просто отражается в одном направлении, а рассеивается по некоторым сложным законам. Чтобы правильно посчитать освещения, для описания объекта, необходимо посчитать ***интеграл освещенности***:

$$I(\varphi_r, \theta_r) = \iint\limits_{\varphi_i \theta_i} L(\varphi_i, \theta_i) R(\varphi i, \theta_i, \varphi_r, \theta_r) \cos\left(n, l_{(\varphi_i, \theta_i)}\right) d\varphi_i \, d\theta_i$$

где n – нормаль, $l_{(\varphi_i,\theta_i)}$ – вектор. В данном методе трассировка лучей происходит во всех направлениях по полусфере.

Интеграл будет оценен ***методом Монте-Карло***:

u – случайная величина, равная распределению на [a, b]; p(u) – плотность вероятности случайной величины u, тогда f(u) – тоже случайная величина:

$$Ef(u) = \int_a^b f(x)p(x)\,dx$$

$$p(x) = \frac{1}{b-a}$$

$$\int_a^b f(x)\,dx = (b-a)Ef(u) \approx \frac{b-a}{N}\Sigma f(u_i)$$

Известно, что есть матожидание случайной величины, оценив которое, можно оценить интеграл. Если известна плотность распределения случайной величины, то можно сделать ее равномерной, вынести ее за скобки и окажется, что интеграл можно оценить, усредняя большое количество значений этой функции. Ниже представлена оценка интеграла освещенности методом Монте-Карло:

$$I(\varphi_r,\theta_r) = \iint_{\varphi_i\theta_i} L(\varphi_i,\theta_i)R(\varphi i,\theta_i,\varphi_r,\theta_r)\cos\left(n, l_{(\varphi_i,\theta_i)}\right)d\varphi_i\,d\theta_i$$

$$\int_a^b f(x)\,dx = (b-a)Ef(u) \approx \frac{b-a}{N}\Sigma f(u_i)$$

$$I(\varphi_r,\theta_r) \approx \frac{\sum_i L(\varphi_i,\theta_j)R(\varphi_i,\theta_i,\varphi_r,\theta_r)\cos\left(n, l_{(\varphi_i,\theta_i)}\right)}{N}$$

В данном методе необходимо производить выборку по значимости – когда распределение заданной случайной величины подстраивается под форму интеграла, если заведомо известен источник. В случае с выборкой по значимости метод Монте-Карло начинает выглядеть по-другому: если раньше была просто сумма функций, деленная на N, то теперь будет сумма функций, деленная на плотность распределения случайной величины:

$$\int_a^b f(x)\,dx \approx \frac{1}{N}\sum_i \frac{f(u_i)}{p(u_i)}$$

ФАКУЛЬТЕТ
ВЫЧИСЛИТЕЛЬНОЙ
МАТЕМАТИКИ И
КИБЕРНЕТИКИ
МГУ ИМЕНИ
М.В. ЛОМОНОСОВА

teach-in

ЛЕКЦИИ УЧЕНЫХ МГУ