Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н. Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Р А С Ч Ё Т Н О - П О Я С Н И Т Е Л Ь Н А Я З А П И С К А

к курсовой работе на тему:

Метод сохранения качества изображений при понижении его разрядности

Студент Капустин А.И. И.О. Фамилия
(Подпись, дата)

Руководитель курсового проекта Оленев А.А. И.О. Фамилия
(Подпись, дата)

Москва 2016

Содержание

Глоссарий

Артефакт — аномалия, возникающая во время визуального представления изображения[1]

Цифровой шум — дефект изображения, вносимый фотосенсорами и электроникой устройств, которые их используют вследствие несовершенства технологий. Цифровой шум заметен на изображении в виде наложенной маски из пикселей случайного цвета и яркости [2]

Фиксированная пороговая обработка (fixed thresholding) — -обработка пикселя на основе какого-то фиксированного порового значения. В случае если текущее значения пикселя больше этого порогового значения, пиксель закрашивает одним фиксированным цветом, иначе -другим. [3]

Интенсивность цвета — степень отличия хроматического цвета от равного ему по светлоте ахроматического, «глубина» цвета. Два оттенка одного тона могут различаться степенью блёклости. При уменьшении насыщенности каждый хроматический цвет приближается к серому.[4]

Ахроматические цвета — оттенки серого (в диапазоне белый — чёрный).[5]

Монохромное изображение — изображение, содержащее свет одного цвета (длины волны), воспринимаемый, как один оттенок (в отличие от цветного изображения, содержащего различные цвета).[**?**]

Цвета шума — система терминов, приписывающая некоторым видам стационарных шумовых сигналов определённые цвета исходя из аналогии между спектром сигнала произвольной природы.[6]

Спектральная плотность S(w) стационарного случайного процесса x(t) — это частотная функция, характеризующая спектральный (частотный) состав процесса, и представляет собой частотную характеристику для средних значений квадратов амплитуд гармоник, на которые может быть разложен случайный процесс.[7]

Светлый пиксель — пиксель,код цвета которого более или равен 128 для одноцветной палитры 0-255.

Темный пиксель — пиксель,код цвета которого менее 128 для одноцветной палитры 0-255.

Обозначения и сокращения

☒ — Искомый пиксель

☐ — Пустой пиксель

■ — Закрашенный пиксель

⊟ — Произвольный пиксель

Введение

Изображения, хранимые в цифровом виде, представляются как массив из значений атрибутов; при этом для представления полноцветных фотографий используется диапазон из нескольких миллионов значений на каждый атрибут. Но количество выводимых отображающим устройством оттенков ограничено. Если графическое устройство не способно воссоздавать достаточное количество цветов, тогда используют растрирование — независимо от того, растровое это устройство или нерастровое.Такие способы используют особенности человеческого зрения и в первую очередь — пространственную интеграцию. Если достаточно близко расположить маленькие точки разных цветов, то они будут восприниматься как одна точка с некоторым усредненным цветом. Если на плоскости густо расположить много маленьких разноцветных точек, то будет создана визуальная иллюзия закрашивания плоскости определенным усредненным цветом. Однако, если увеличивать размеры точек и (или) расстояние между ними, то иллюзия сплошного закрашивания исчезает — включается другая система человеческого зрения, которая обеспечивает способность различать объекты, подчеркивать контуры. В компьютерных графических системах часто используют эти методы. Они позволяют увеличить количество оттенков цветов за счет снижения пространственного разрешения растрового изображения(иначе говоря — это обмен разрешающей способности на количество цветов). В компьютерной графике такие методы растрирования получили название дизеринг(eng.dithering - разрежение, дрожание). В последнее время, в связи с распространением высококачественного видео(4K видео, 360° видео), высококачественных фотографий, проблема сохранения качества изображения при уменьшении его размера становится все более актуальной.

1 Аналитический раздел

В компьютерной графике дизеринг используется для создания иллюзии глубины цвета для изображений с относительно небольшим количеством цветов в палитре. Отсутствующие цвета составляются из имеющихся путём их «перемешивания». Например, если необходимо получить отсутствующий в палитре фиолетовый цвет, его можно получить, разместив красные и синие пиксели в шахматном порядке; оранжевый цвет может быть составлен из красных и желтых точек.

При оптимизации изображений путём уменьшения количества цветов, применение дизеринга приводит к визуальному улучшению изображения, однако для отдельных сжатых форматов (например, PNG), увеличивает его размер.

Алгоритмы дизеринга подразделяются на следующие категории:

— Случайный дизеринг(Random dither);
— Шаблонный дизеринг(Patterning);
— Упорядоченный дизеринг(Ordered);
— Дизеринг рассеивания ошибок(Error-diffusion);
— Прочие виды дизеринга

Нижеприведенные алгоритмы описываются для черно-белых изображений. Для цветных изображений алгоритмы аналогичны.

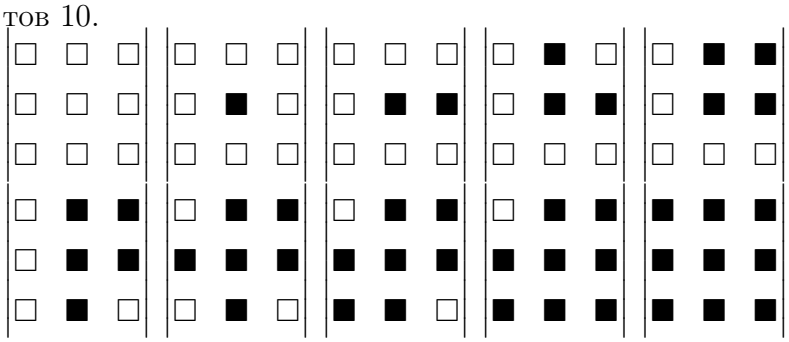
1.1 Случайный дизеринг

Этот алгоритм тривиален. По сравнению с остальными алгоритмами, его качество слишком низко, поэтому он применяется лишь там, где необходима высокая скорость работы в ущерб качеству.[3] Для каждого пикселя в нашем черно-белом изображении мы генерируем случайное число в диапазоне 0-255: если случайное число больше, чем значение в данной точке, то отображаем белый пиксель, иначе отображаем черн пиксель. Это создает изображение с большим количеством шумов. Хотя изображение выглядит неточным и зернистым, оно не содержит артефактов[3]]. Этот метод дизеринга полезен при воспроизведении низкокачественных изображений, где отсутствие артефактов более важно, чем наличие шумов. Например, изображение содержит градиент всех уровней от черного до белого. Это изображение не будет иметь артефактов после того, как к нему применят случайный дизеринг, а остальные методы дизеринга приведут к возникновению артефактов [3].

1.2 Шаблонный дизеринг

Шаблонный дизеринг подрузамевает то, что мы увеличиваем разрешение изображения. Так же, как и случайный дизеринг, это тривиальный алгоритм, но он гораздо более эффективен.[8] Для каждой точки изображения мы генерируем «шаб-

лон» пикселей, который аппроксимирует эту точку. Тем самым имтируется больший набор оттенков, чем поддерживает наша глубина цвета. Например, шаблон 3x3. Он имеет 512 вариантов возможных расположений пикселей, но расположение пикселей не влияет на интенсивность цвета шаблона. Интенсивность формируется на основе количества черных пикселей, содержащихся в шаблоне. То есть возможных вариантов 10.



1.3 Упорядоченный дизеринг

Значительным недостатком шаблонного дизеринга является пространственное увеличение картинки(и увеличение её разрешения). Упорядоченный дизеринг позволяет избежать этого пространственного искажения. Для того, чтобы достичь этого, каждый пиксель в исходном изображении сопоставляется с пикселем на конечном изображении один-к-одному.Существуют два вида упорядоченного дизеринга: кластерный и дисперсный. Суть этих методов заключается в том, что исходное изображение разбивается на квадраты пикселей и значения маски в каждой точке квадрата выступает в роли порогового значения. Если значение цвета пикселя(отмастшабированное под интервал маски) в данной точке больше значения маски, то красим пиксель в черный цвет, иначе в белый. Кластерные паттерны выглядят следующим образом:

$$\begin{vmatrix} 8 & 3 & 4 \\ 6 & 1 & 2 \\ 7 & 5 & 9 \end{vmatrix} \qquad \begin{vmatrix} 1 & 7 & 4 \\ 5 & 8 & 3 \\ 6 & 2 & 9 \end{vmatrix}$$

Кластерные паттерны применяются в случаях когда понятие «конкретный пиксель» у устройства вывода информации отсутсвует(ЭЛТ-мониторы и подобное). Во многих исследованиях [9][3] было отмечено, что если устройство вывода позволяет применить дисперсный метод, то его применение является предпочтительным.Так же Байер [9] показал, что для матриц порядков степени двух существует оптимальная структура дисперсных точек, которая приводит к наименьшему количеству шумов(для матрицы 2x2 и 4x4 соотвественно):

$$\begin{vmatrix} 1 & 3 \\ 4 & 2 \end{vmatrix} \qquad \begin{vmatrix} 1 & 9 & 3 & 11 \\ 13 & 5 & 15 & 7 \\ 4 & 12 & 2 & 10 \\ 16 & 8 & 14 & 6 \end{vmatrix}$$

Основным недостатком данного метода считается то, что в результате его работы формируется большое количество артефактов[8].

1.4 Дизеринг при помощи диффузии ошибок

Метод, обладающий наилучшим качеством среди представленных, - метод рассеивания ошибок. Но так же он, к сожалению, самый медленный.[3] Существуют несколько вариантов этого алгоритма, причем скорость алгоритма обратно пропорционально качеству изображения.[3] Суть алгоритма: для каждой точки изображения находим ближайший возможный цвет. Затем мы рассчитываем разницу между текущим значеним и ближайшим возможным. Эта разница и будем нашем значением ошибки.Это значение ошибки мы распределяем между соседними элементами, которые мы ещё не посещали. Для последних точек ошибка распределяется между уже посещенными точками.

1.5 Вариации алгоритма дизернга при помощи диффузии ошибок

Линия сканирование движется слева-направо. Когда линия сканирования доходит до конца горизонтальной строки пикселей, переходим к первому пикселю следующей строки и повторяем необходимые действия.
*Примечание: числа на схемах - это доли от значения ошибки. Например, 7/16 на схеме выглядит как 7. То есть 7 обозначает некую величину, равную значение ошибки*7/16*

1.5.1 Фильтр Флойда-Стейнберга

Каждый пиксель распределяет свою ошибку на соседние с ним пиксели. Коэффициенты были подобраны таким образом, что в районах с интенсивностью $1/2$ от общего количество оттенков, изображение выглядело похожим на шахматную доску.

$$\begin{vmatrix} \boxminus & \boxtimes & 7 \\ 3 & 5 & 1 \end{vmatrix} (1/16)$$

1.5.2 "Ложный"фильтр Флойда-Стейнберга

В случае сканирования слева-направо этот фильтр порождает большое количество артефактов.Чтобы получить изображение с меньшим количеством артефактов, нужно чётные строки сканировать справа-налево, а нечетные строки сканировать слева-направо.

$$\begin{vmatrix} \boxtimes & 3 \\ 3 & 2 \end{vmatrix} (1/8)$$

1.5.3 Фильтр Джарвиса,Джунка и Нинка

В случае когда фильтры Флойда-Стейдберга дают недостаточно хороший результат, применяются фильтры с более широким распределением ошибки. Фильтр Джарвиса, Джунка и Нинка требует связи с 12 соседями, что очевидно ведет в большим затратам памяти и времени[3]:

$$\begin{vmatrix} \boxminus & \boxminus & \boxtimes & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{vmatrix} (1/48)$$

1.5.4 Фильтр Стаки

Фильтр разработан на основе фильтра Джарвиса, Джунка и Нинка.После такого как мы вычислим 8/42 ошибки, остальные значения можно получить при помощи побитовых сдвигов, тем самым сокращая время работы алгоритма.

$$\begin{vmatrix} \boxminus & \boxminus & \boxtimes & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{vmatrix} (1/42)$$

1.5.5 Фильтр Бурка

Стаки. Результат можно получить чуть быстрее за счет использования побитовых операций.

$$\begin{vmatrix} \boxminus & \boxminus & \boxtimes & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \end{vmatrix} (1/32)$$

Существует много различных вариантов фильтро дизеринга при помощи диффузии ошибок, здесь приведены наиболее популярные алгоритмы.[3]

1.6 Прочие виды дизеринга

Компьютерная графика не стоит на месте и завидной переодичностью появляются всё новые виды дизеринга, не укладывающиеся в привычную классификацию.

1.6.1 Алгоритм Юлиомы

Существует три вариации этого алгоритма [10] значительно отличающиеся друг от друга. Основная идея этих алгоритмов заключается в оптимальном подборе цвета ограниченной цветовой палитры, основываваясь на визуальном восприятии этого цвета, а не на его числовом коде. Отличается от остальных алгоритмов значительными временными затратами, что делает применение этого алгоритма для решения практических задач затруднительным.

1.7 Выбор оптимального класса алгоритма

Вышеприведенные методы упорядочены по качеству получаемого на выходе изображения, однако, такие соображения как время, экономия памяти и прочие являются определяющими при выборе алгоритма[3]. На основе вышеприведенных данных сравним классы алгоритмов дизеринга.

Характеристика / Вид алгоритма	Скорость	Качество	Доп память
Случайный	+	-	-
Шаблонный	+-	-+	+
Упорядоченный	-+	+-	-
Диффузия ошибок	-	+	+
Остальные	-	+	+

Самым быстрым классом алгоритмов дизеринга является случайный дизеринг.Однако, качество получаемых при помощи изображений низко. Алгоритмы упорядоченного дизеринга позволяют получить изображения более высокого качества, однако работают более медленно и требуют дополнительных затрат. В зависимости от вычислительных ресурсов и требуемого результата рациональным будет применение различых алгоритмов.

1.8 Оценка алгоритмов

n - количество пикселей в изображении

Характеристика / Вид алгоритма	Время	Память
Случайный	O(n)	O(1)
Упорядоченный	O(n)	O(1)
Диффузия ошибок	O(n)	O(n)
Первый алгоритм Юлиомы	$O(n^3)$	O(n)

2 Конструкторский раздел

2.1 Оценка качества изображений

2.1.1 PSNR

Пиковое отношение сигнала к шуму (англ. peak signal-to-noise ratio) - соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала.[11]

$$PSNR = 20\log_{10}\frac{MAX_i}{\sqrt{MSE}} \tag{2.1}$$

Где MAX_i - это максимальное значение, принимаемое пикселем изображения, MSE - среднеквадратичное отклонение. Для двух монохромных изображений I и K размера m×n, одно из которых считается зашумленным приближением другого, вычисляется так:

$$MSE = \frac{1}{m*n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|^2 \tag{2.2}$$

2.1.2 SSIM

Индекс структурного сходства (SSIM от англ. structure similarity) — метод измерения схожести между двумя изображениями путем полного сопоставления. SSIM-индекс является развитием традиционных методов, таких как PSNR (peak signal-to-noise ratio) и метод среднеквадратичной ошибки MSE, которые оказались несовместимы с физиологией человеческого восприятия.

Отличительной особенностью метода, в отличие от MSE и PSNR, является то, что он учитывает «восприятие ошибки» благодаря учёту структурного изменения информации. Идея заключается в том, что пиксели имеют сильную взаимосвязь, особенно когда они близки пространственно. Данные зависимости несут важную информацию о структуре объектов и о сцене в целом. Особенностью является, что SSIM всегда лежит в промежутке от -1 до 1, причем при его значении равном 1, означает, что мы имеем две одинаковые картинки. Общая формула имеет вид

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(\sigma_x y + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.3}$$

Тут μ_x среднее значение для первой картинки, μ_y для второй, σ_x среднеквадратичное отклонение для первой картинки, и соотвественно σ_y для второй, $\sigma_x y$ это уже ковариация. Она находится следующим образом:

$$\sigma_x y = \mu_x y - \mu_x\mu_y \tag{2.4}$$

c_1 и c_2 - поправочные коэффициенты, которые нужны вследствие малости знаменателя.

$$c_1 = (0{,}01 * d)^2 \tag{2.5}$$

$$c_2 = (0{,}03 * d)^2 \tag{2.6}$$

d - количество цветов, соответствующих данной битности изображения Для подтверждения или опровержения вышеописанных гипотезы реализуются несколько алгоритмов случайного распределения, алгоритм Флойда-Стейнберга, модификации на основе алгоритма Флойда-Стейнберга. Результаты работы алгоритмов сравниваются по времени работы, по количеству затрачиваемой памяти,а так же по SSIM и PSNR.

2.1.3 Алгоритм случайного распределения

P(x,y) - цвет конкретного пикселя

Листинг 2.1 — Алгоритм случайного распределения

```
for x in range(height):
    for y in range(weight):
        if P(x,y)>127:
            P(x,y) = 255
        else:
            P(x,y) = 0
```

2.2 Виды случайных распределений

2.2.1 Белый шум

Белый шумом называют сигнал с равномерной спектральной плотностью на всех частотах и дисперсией, равной бесконечности. Является стационарным случайным процессом. В качестве сигнала в задаче дизеринга рассматривается последовать последовательность чисел, получаемых от генератора случайных чисел.

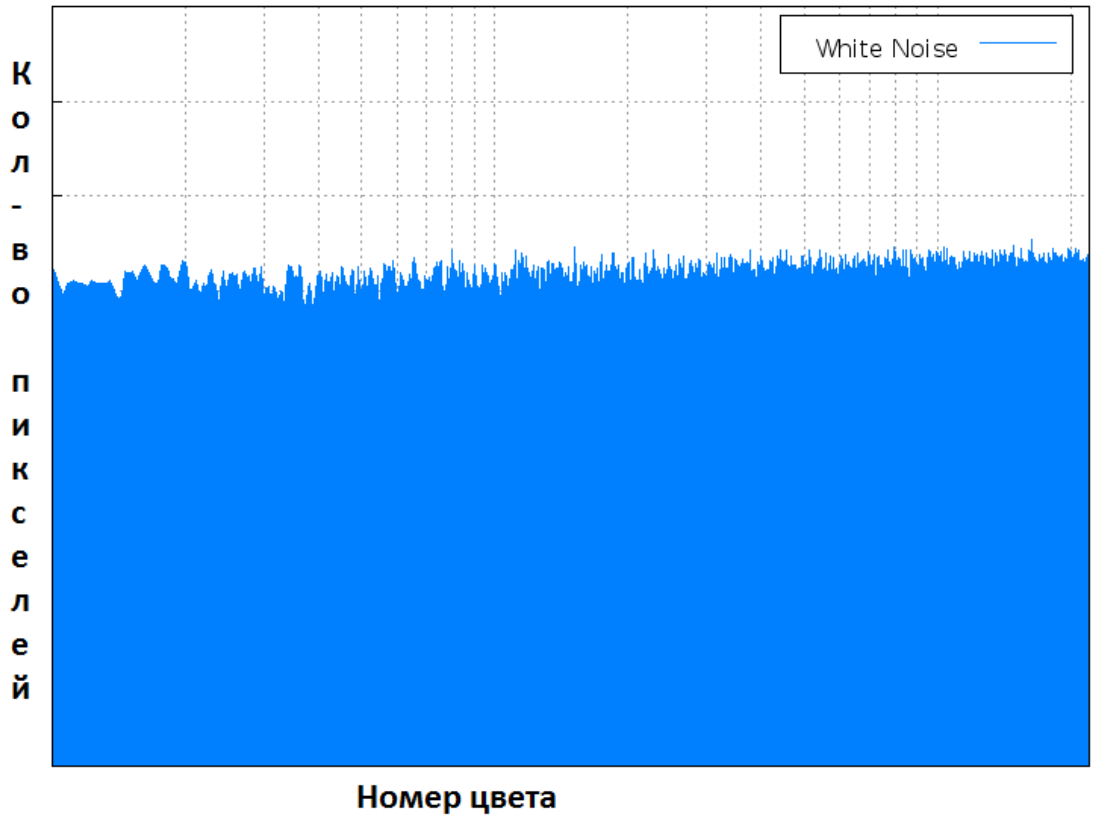


Рисунок 2.1 — Диаграмма белого шума

2.2.2 Коричневый шум

Спектральная плотность коричневого шума пропорциональна $1/f^2$, где f — частота. Это означает, что на низких частотах шум имеет больше энергии, чем на высоких. То есть пикселей темных цветов большей, чем пикселей светлого цвета.Применение фильтра коричневого шума в целом затемняет получаемое изображение.

Листинг 2.2 — Получение коричневого шума

```
1 def smoother(noise):
2     output = []
3     for i in range(len(noise) - 1):
4         output.append(0.5 * (noise[i] + noise[i+1]))
5 return output
```

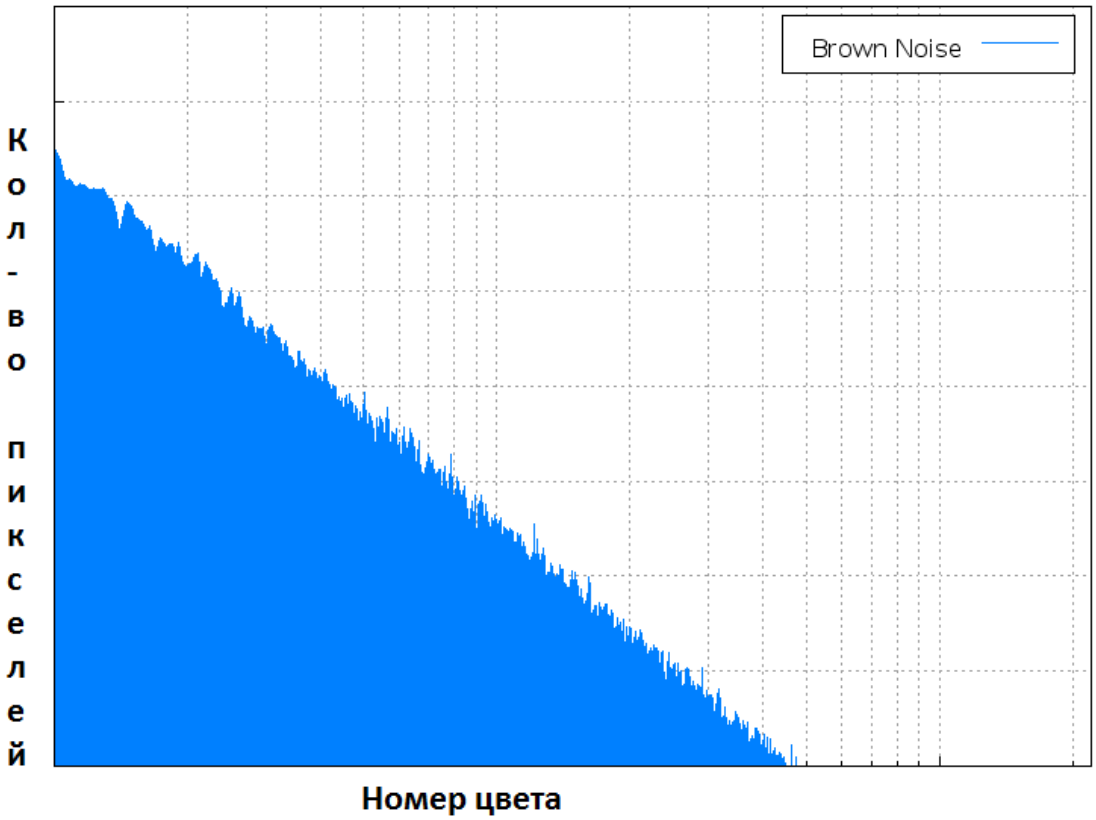


Рисунок 2.2 — Диаграмма красного шума

2.2.3 Гауссовский шум

Гауссовский шум - шум, имеющий функцию плотности вероятности (PDF), равную нормальному распределению, которое также известно как гауссово распределение.

$$p_g(z) = \frac{1}{\sigma\sqrt{2 * \pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{2.7}$$

z - количество цветов, μ -среднее значение, σ - стандартное отклонение.

2.2.4 Фиолетовый шум

Фиолетовый шум представляет собой противооложноть между коричневому шуму. Получение его аналогично получению коричневого шума.Применение фильтра фиолетового шума в целом засветляет получаемое изображение.

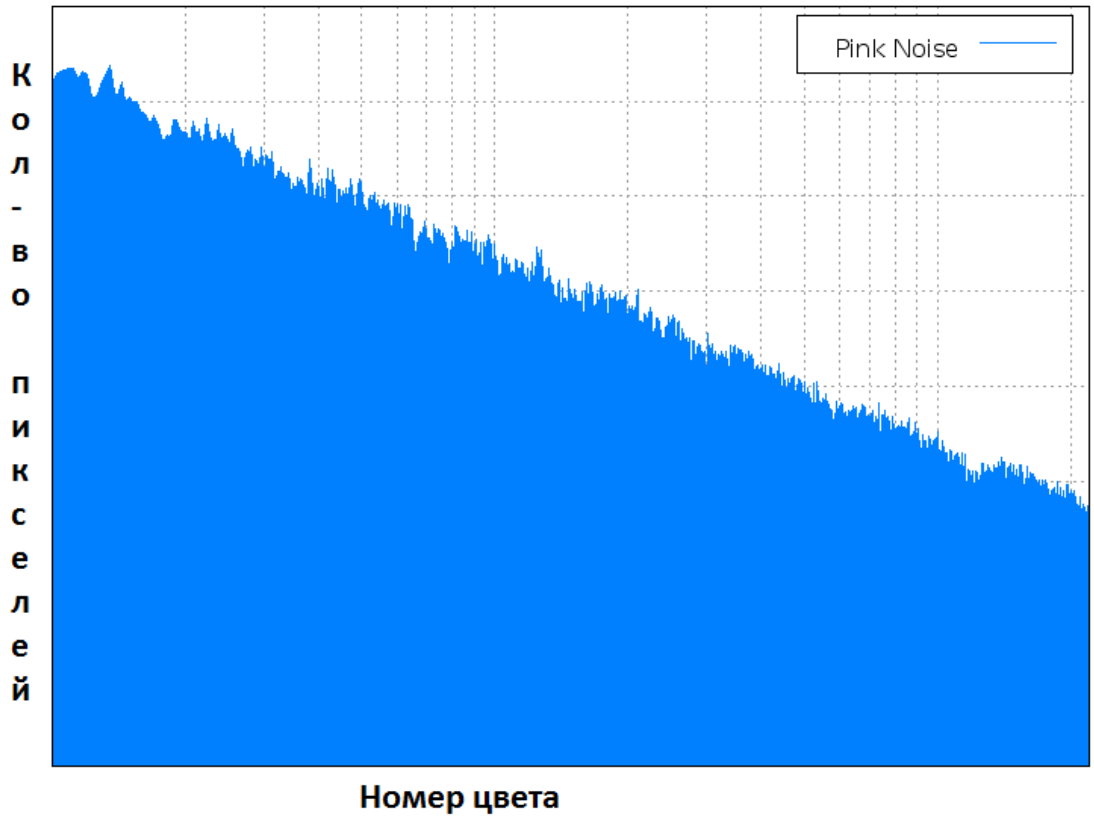


Рисунок 2.3 — Диаграмма розового шума

Листинг 2.3 — Получение розового шума

```
1 def rougher(noise):
2     output = []
3     for i in range(len(noise) - 1):
4     output.append(0.5 * (noise[i] - noise[i+1]))
5 return output
```

2.2.5 Розовый и синий шумы

Розовый и синий шумы представляют собой "промежуточные"шумы.Изобржаение с розовым шумом темнее изображения с белым шумом, но светлее изображения с коричневым шумом. Изображение с синим шумом светлее изображения с белым шумом, но темнее чем изображение с фиолетовым шумом. Их получение аналогично получению со коричневого и фиолетового шумов.

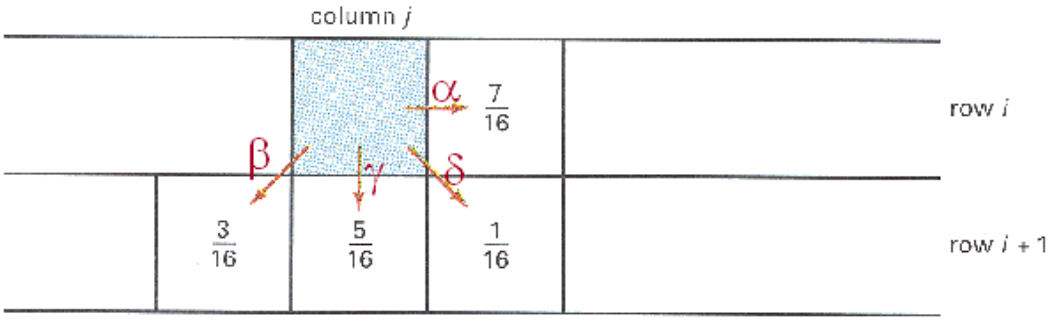
2.3 Алгоритм Байера

Листинг 2.4 — Алгоритм Флойда-Стейнберга

```
For each pixel, Input,:
    Factor  = ThresholdMatrix[xcoordinate % X][ycoordinate % Y];
    Attempt = Input + Factor * Threshold
    Color = FindClosestColorFrom(Palette, Attempt)
```

2.4 Алгоритм Флойда-Стейнберга

Рассмотрим более детально алгоритм Флойда-Стейнберга. P(x,y) - цвет пикселя в точке x,y I(x,y) - предполгаемый цвет пикселя с учетом ошибки(действительное число) Следует отметить, что сумма "долей"рассевания ошибки



$$\alpha + \beta + \gamma + \delta = 1.0$$

Рисунок 2.4 — Схема рассеивания ошибки

и для других алгоритмов упорядоченного дизеринга будет равна единице.

Листинг 2.5 — Алгоритм Флойда-Стейнберга

```
for x in range(width):
    for y in range(height):
        P(x,y) = trunc(I(x,y)+0.5)
        e = I(x,y) - P(x,y)
        I(x,y-1) += 7/16*e
        I(x+1, y-1) += 3/16*e
        I(x+1, y) += 5/16*e
        I(x+1, y+1) += 1/16*e
```

2.5 Ложный алгоритм Флойда-Стейнберга

Отличительной особенностью ложного алгоритма Флойда-Стейнберга является по пикселям изображения справа-налево, а не слева-направо как в большинстве других алгоритмов.

Листинг 2.6 — Ложный алгоритм Флойда-Стейнберга

```
for x in range(width):
    for y in range(height):
        P(x,y) = trunc(I(x,y)+0.5)
        e = I(x,y) - P(x,y)
        P(x,y) = trunc(I(x,y)+0.5)
        e = I(x,y) - P(x,y)
        I(x,y-1) += 3/8*e
        I(x-1, y-1) += 2/8*e
        I(x-1, y) += 3/8*e
```

2.6 Алгоритм Джарвиса,Джунка и Нинка

Листинг 2.7 — Алгоритм Джарвиса,Джунка и Нинка

```
for x in range(width):
    for y in range(height):
        P(x,y) = trunc(I(x,y)+0.5)
        e = I(x,y) - P(x,y)
        P(x,y) = trunc(I(x,y)+0.5)
        e = I(x,y) - P(x,y)
        I(x,y-1) += 3/48*e
        I(x+1,y+1) += 5/48*e
        I(x,y+1) += 7/48*e
        I(x-1,y+1) += 5/48*e
        I(x-2,y+1) += 3/48*e
        I(x-1,y+1) += 5/48*e
        I(x+2,y+2) += 1/48*e
        I(x+1,y+2) += 3/48*e
        I(x, y+2) += 5/48*e
        I(x-1,y+2) += 3/48*e
        I(x-2,y+2) += 1/48*e
        I(x+1,y) += 7/48*e
        I(x+2,y) += 5/48*e
```

2.7 Первый алгоритм Юлиомы

Листинг 2.8 — Первый алгоритм Юлиомы

```
For each pixel, Input, in the original picture:
    Factor  = ThresholdMatrix[xcoordinate % X][ycoordinate % Y];
    Make a Plan, based on Input and the Palette.

    If Factor < Plan.ratio,
        Draw pixel using Plan.color2
    else,
        Draw pixel using Plan.color1
```

Листинг 2.9 — План нахождения цвета пикселя

```
SmallestPenalty = 10^99
For each unique combination of two colors from the palette, Color1
   and Color2:
    For each possible Ratio, 0 .. (X*Y-1):
        Mixed = Color1 + Ratio * (Color2 - Color1) / (X*Y)
        Penalty = Evaluate the difference of Input and Mixed.

        If Penalty < SmallestPenalty,
        SmallestPenalty = Penalty
        Plan = { Color1, Color2, Ratio / (X*Y) }1
```

Листинг 2.10 — Вспомогательная матрица перого Алгоритма Юлиомы

```
    [0/64, 48/64, 12/64, 60/64, 3/64, 51/64, 15/64, 63/64,
    32/64, 16/64, 44/64, 28/64, 35/64, 19/64, 47/64, 31/64,
    8/64, 56/64, 4/64, 52/64, 11/64, 59/64, 7/64, 55/64,
    40/64, 24/64, 36/64, 20/64, 43/64, 27/64, 39/64, 23/64,
    2/64, 50/64, 14/64, 62/64, 1/64, 49/64, 13/64, 61/64,
    34/64, 18/64, 46/64, 30/64, 33/64, 17/64, 54/64, 29/64,
    10/64, 58/64, 6/64, 54/64, 9/64, 57/64, 5/64, 53/64,
    42/64, 26/64, 38/64, 22/64, 41/64, 25/64, 37/64, 21/64]
```

3 Технологический раздел

3.1 Выбор языка программирования

Для реализации данных алгоритмов был выбран язык C++. Данный язык был обоснован следующими причинами: Причины:

а) Компилируемый язык со статической типизацией.
б) Сочетание высокоуровневых и низкоуровневых средств.
в) Реализация ООП.
г) Наличие удобной стандартной библиотеки шаблонов

3.2 Выбор вспомогательных библиотек

Для реализации программы была выбрана библиотека Qt.

а) Широкие возможности работы с изображениями, в том числе и попиксельно
б) Наличии более функциональных аналогов стандартной библиотеки шаблонов в том числе для разнообразных структур данных

Так же были были использованы библиотеки ImageMagick(для конвертации изображения в ограниченную цветовую палитру), OpenMP(многопоточность), OpenGL(работа с шейдерами)

3.2.1 Диаграмма классов

Для реализации различных алгоритмов была разработана следующая структура классов. Был создан абстракный класс дизеринга Dithering с интерфейсом, наследуюмом в дочерних классах. Так же была введена система менджеров: DitherManager, MetricsManager, DataManager и MainManager.

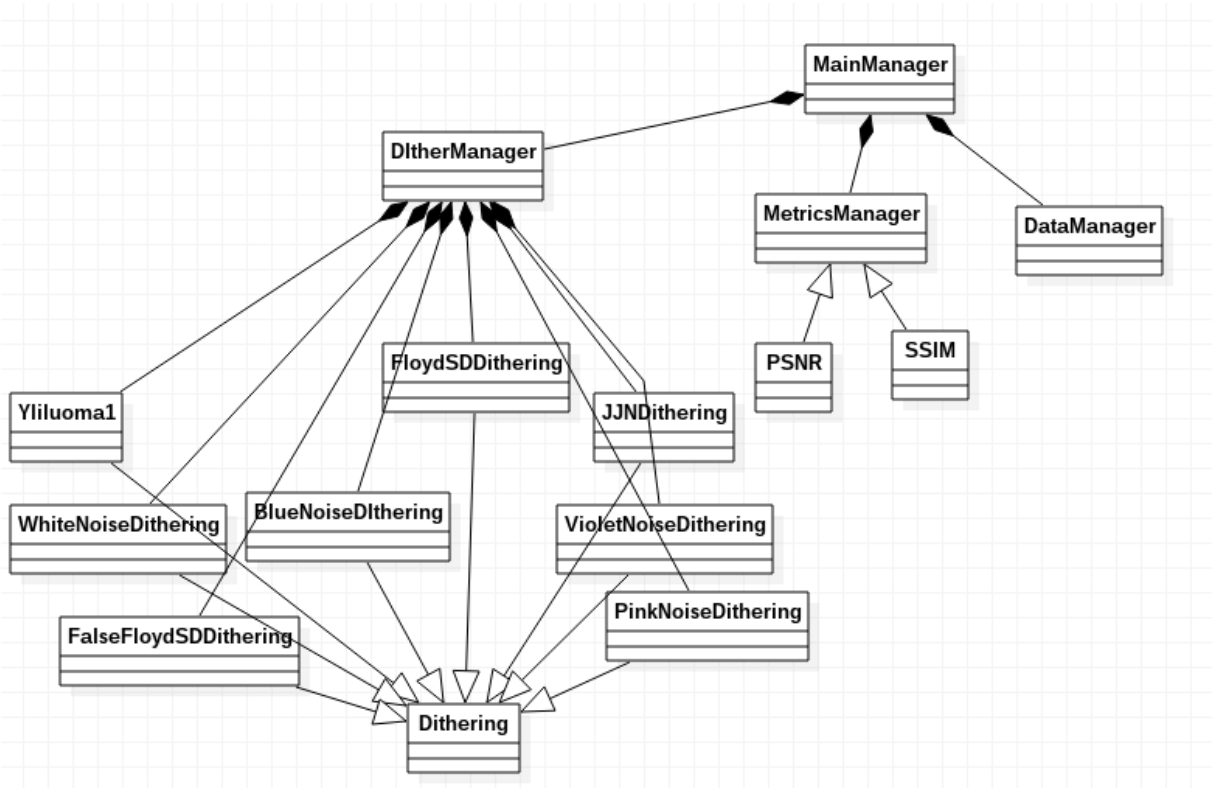


Рисунок 3.1 — Диаграмма классов

4 Исследовательский раздел

4.1 Время дизеринга раличных алгоритмов

Рассмотрим время работы различных алгоритмов для различных размеров изображения.

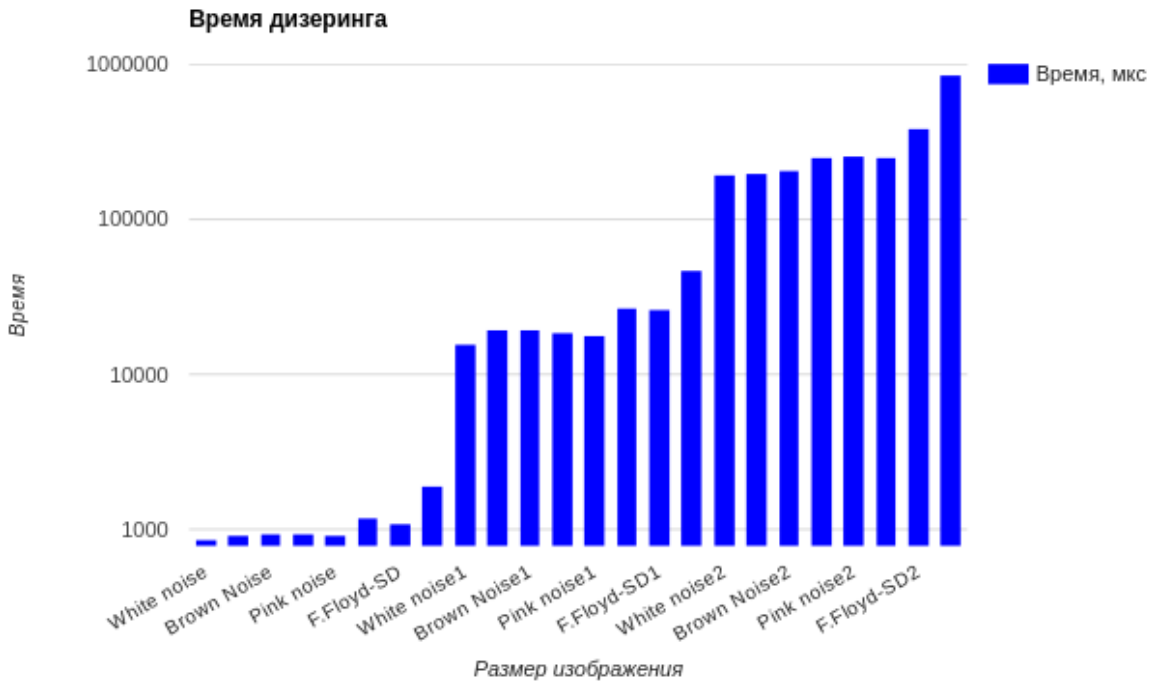


Рисунок 4.1 — Диаграмма времени дизеринга(логарифмическая шкала)

	Размер, пиксели	Время, мкс
White noise	133x90	862
Blue noise	133x90	930
Brown noise	133x90	934
Violet noise	133x90	937
Pink noise noise	133x90	930
Floyd-SD	133x90	1200
F. Floyd-SDe	133x90	1093
JJN	133x90	1909
White noise	458x458	15735
Blue noise	458x458	19374
Brown noise	458x458	19432
Violet noise	458x458	18787
Pink noise noise	458x458	18129
Floyd-SD	458x458	27173
F. Floyd-SDe	458x458	26424
JJN	458x458	47201
White noise	458x458	194376
Blue noise	458x458	200577
Brown noise	458x458	208400
Violet noise	458x458	251294
Pink noise noise	458x458	258775
Floyd-SD	458x458	251294
F. Floyd-SDe	458x458	387104
JJN	458x458	857481

Из рассмотрения вынесены алгоритм Юлиомы в вследствие того, что он значительно медленней других алгоритмов(2732568 мкс для изображения 113x90) в и алгоритм Байера, реализованный при помощи шейдеров, вследствии того, что он не не укладывается в рамки требуемой палитры (при этом он работает очень быстро 64 мс для изображени 640x480).

4.2 Качество получаемого изображения

	PSNR	SSIM
White noise	33.2894	0.914778
Blue noise	36.1756	0.971626
Brown noise	33.32370	0.915767
Violet noise	37.63480	0.984574
Pink noise	36.4484	0.974718
Floyd-SD	37.0553	0.979173
F. Floyd-SDe	36.8401	0.976452
JJN	37.30740	0.981688
Yliouma	36.2359	0.967796
Without dithering	37.6348	0.984574

Несмотря на то, что некоторые сложные алгоритмы дизеринга диффузии ошибок обещат получения хорошего качества изображений,некоторые алгоритмы случайного дизеринга на конкретных изображениях дают лучший результат. Для того чтобы получить наилучший результат дизеринга, следует проанализировать результаты дизеринга нескольких изображений и выбрать среди них наилучшее. Так же следует отметить некоторую необъективность метрик: результат метрик не всегда совпадает с человеческим восприятием картинки.

4.3 Размер получаемого изображения

	Разрешение, пикс	Размер, кб	Исх. раз., кб
White noise	900x675	186	2373 bmp,1779 png
Blue noise	900x675	135	
Brown noise	900x6750	186	
Violet noise	900x675	98	
Pink noise	900x675	1158	
Floyd-SD	900x675	1273	
F. Floyd-SDe	900x675	143	
JJN	900x675	117	
White noise	3984x32355	3431	50344 bmp,37758 png
Blue noise	3984x3235	2570	
Brown noise	3984x3235	3432	
Violet noise	3984x3235	1950	
Pink noise	3984x3235	2406	
Floyd-SD	3984x32355	3605	
F. Floyd-SDe	3984x3235	4269	
JJN	3984x3235	3716	

Рисунок 4.2 — Диаграмма размера изображения(логарифмическая шкала)

Из вышеприведенной таблицы, можно заметить, размер изображения после дизеринга значительно уменьшиается,достирается выигрыш в размере изображение до 15 раз, в зависимости от иходного контейнера изображения и выбранного способра дизеринга.

Заключение

В данной работе были реализован различные алгоритмы дизеринга, был произведен сравнения и анализ этих алгоритмов. Программа позволяет получить изображение схожего визуального качества при значительном уменьшении размера. Был получен вывод о том, что для различных целей следует использовать различные алгоритмы дизеринга, универсального алгоритма дизеринга не существует. Программа не привязана к какой-то конкретной операционной системе и может быть скомпилирована и запущена на всех популярных ОС.

Список использованных источников

1. Википедия. — https://en.wikipedia.org/wiki/Artifact_(error).
2. Википедия. — https://en.wikipedia.org/wiki/Image_noise.
3. *Crocker, Lee Daniel.* Digital Halftoning / Lee Daniel Crocker. — 1989.
4. Википедия. — https://en.wikipedia.org/wiki/Colorfulness.
5. Википедия. — https://en.wikipedia.org/wiki/Color.
6. Википедия. — https://en.wikipedia.org/wiki/Colors_of_noise.
7. Сайт ВГТУ. — http://www.zdo.vstu.edu.ru/html/L9P3.html.
8. *Ulichney, R.* Digital Halftoning / R. Ulichney. — The MIT Press, 1987.
9. *Bayer, B.E.* An Optimum Method for Two-Level Rendition of Continuous Tone Pictures / B.E. Bayer. — IEEE International Conference on Communications, Conference Records, 1973.
10. *Yliluoma, Joel.* Joel Yliluoma's arbitrary-palette positional dithering algorithm / Joel Yliluoma.
11. *Инсаф, Ашрапов.* PSNR и SSIM или как работать с изображениями под C / Ашрапов Инсаф. — https://habrahabr.ru/post/126848/, 2011.