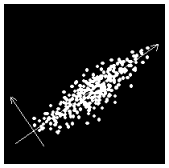




**Практическое задание 1**

## Тема: Создание тестовой оболочки. Измерение PSNR. Преобразование цветовых пространств. Квантование.

1. Пользуясь предпочитаемыми вами средствами программирования, создайте программу, отображающую панель с двумя полноцветными изображениями стандартного разрешения 512 x 512 пикселей (глубина каждого цветового канала *Red, Green, Blue* по 8 бит, в сумме 24 бит). Оставьте на панели достаточно много свободного места для размещения различных кнопок и текстовых полей, которые вы добавите при выполнении последующих заданий.
   Научитесь загружать (и отображать) на каждом из двух мест на панели изображения из файлов стандартных форматов BMP, PNG, TIFF, JPEG и сохранять каждое изображение в файл формата BMP (или PNG, TIFF).

2. Реализуйте алгоритм, вычисляющий разницу в метрике *PSNR* между двумя открытыми изображениями стандартного разрешения. Добавьте кнопку «Вычислить PSNR» на панель, созданную вами в пункте 1.

   Для монохроматических (одноканальных) изображений *PSNR* вычисляется следующим образом (в указанной ниже формуле $N$ – это количество пикселей по одной из сторон растра, в данном случае $N = 512$, $x_{ij}, y_{ij}$ это значения светимости пикселей соответственно первого и второго изображений):

$$PSNR = 10 \cdot \log_{10} \frac{255^2 \cdot N^2}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left( x_{ij} - y_{ij} \right)^2}$$

   Для полноцветных RGB-изображений *PSNR* вычисляется аналогично с той лишь разницей, что суммировать нужно по всем пикселям всех трех каналов, и, соответственно, при усреднении делить на утроенное количество пикселей.

$$PSNR = 10 \cdot \log_{10} \frac{3 \cdot 255^2 \cdot N^2}{\sum_{k \in \{R,G,B\}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left( x_{ijk} - y_{ijk} \right)^2}$$

   Проверьте корректность работы алгоритма (в частности, убедитесь, что корректно обрабатывается случай сравнения одинаковых изображений). Попробуйте вычислить *PSNR* между полностью черным и полностью белым изображениями (должен получиться ноль dB). Заметим, что величина *PSNR* не обязательно является целым числом, чаще дробным (но всегда неотрицательным). Желающие могут сравнить свой результат со значением *PSNR*, вычисленным в MatLab http://www.mathworks.com/help/images/ref/psnr.html

3. Научитесь преобразовывать изображение из цветного в чёрно-белое (точнее, в оттенках серого, англ. *Grayscale*). Сделайте это двумя способами:

   a) с равными весами (то есть просто для каждого пикселя взять среднее арифметическое по трем цветовым каналам)

```
Y = (R + G + B) / 3                    (a)
```

   b) с весами по стандарту CCIR 601-1:

```
Y = (77/256)*R + (150/256)*G + (29/256)*B    (b)
```

Убедитесь, что вы используете только целочисленную арифметику, минимизируете количество операций, и, по возможности, умножение/деление заменяете побитовым сдвигом. Визуально оцените результат преобразования с использованием формул (a), (b), одновременно отображая оба полученных черно-белых (точнее, в оттенках серого) изображения в программе, созданной вами в пункте 1. Тестирование следует выполнить для нескольких тестовых изображений, но среди них обязательно должно быть image_Peppers512rgb.png

Какой вариант преобразования вам показался более «правдоподобным» ?

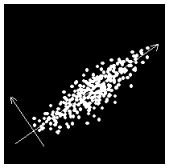
4. Реализуйте преобразование изображения из цветового пространства *RGB* в $YC_bC_r$ и обратно. Для каждого пикселя значения $Y$, $C_b$, $C_r$ следует хранить как целые числа (выясните, сколько бит потребуется для хранения таких чисел). Убедитесь теоретически (на бумаге) и практически, что (при правильном округлении) если $R,G,B \in [0;255] \cap \mathbb{Z}$, то и значения в каналах $Y$, $C_b$, $C_r$ всегда будут из множества $[0;255] \cap \mathbb{Z}$.

   Канал Y называется яркостным, а каналы $C_b$ и $C_r$ – цветоразностными (поскольку с точностью до константы получаются из разностей $(B–Y)$ и $(R–Y)$). Преобразованное в пространство $YC_bC_r$ изображение следует отображать в созданной вами программе поканально, каждый канал черно-белым (точнее, в оттенках серого). Добавьте кнопки $Y$, $C_b$, $C_r$ на панель, созданную в пункте 1.

$$Y = (77/256)*R + (150/256)*G + (29/256)*B$$
$$C_b = -(43/256)*R - (85/256)*G + (128/256)*B + 128 = (144/256)*(B-Y) + 128$$
$$C_r = (128/256)*R - (107/256)*G - (21/256)*B + 128 = (183/256)*(R-Y) + 128$$

$$R = Y + (256/183)*(C_r - 128)$$
$$G = Y - (5329/15481)*(C_b - 128) - (11103/15481)*(C_r - 128)$$
$$B = Y + (256/144)*(C_b - 128)$$

   Убедитесь теоретически (на бумаге) и практически, что преобразование *RGB* в $YC_bC_r$ и обратно по указанным формулам является (почти) тождественным. Убедитесь, что при реализации алгоритма вы используете только целочисленную арифметику, минимизируете количество операций, и, по возможности, умножение/деление заменяете побитовым сдвигом. Для нескольких тестовых изображений посчитайте разницу в метрике $PSNR$ между исходным изображением и изображением, полученным после преобразования цветовых пространств из *RGB* в $YC_bC_r$ и обратно. Потери возможны из-за округления до целого, а также в связи с тем, что перемножение матриц преобразования, быть может, дает лишь приблизительно единичную матрицу.





5. Операции, описанные в этом задании, применяются к полноцветному изображению в модели *RGB* (глубина цвета не менее 24 бит на пиксель). На ранее созданной вами тестовой оболочке должно быть место для двух растров. Слева отображайте исходное изображение, а справа – преобразованное. Для оценки потерь сравнивайте изображения визуально (субъективно) и с использованием метрики *PSNR* (объективно).

   Реализуйте квантование, построив палитру из 1024 цветов (разбив цветовое пространство на 1024 ячейки) предложенными ниже способами. Для выбранного тестового изображения проведите серию экспериментов и заполните таблицу.

| Тестовое изображение | | image_Baboon512rgb.png | |
|---|---|---|---|
| Метод квантования | | Потери в метрике PSNR, dB | Место в субъективной ранговой шкале |
| Равномерное, *RGB* | 3+4+3 bit | | 9 |
| | 4+4+2 bit | | 8 |
| Равномерное, $YC_bC_r$ | 2+4+4 bit | | 7 |
| | 5+2+3 bit | | 6 |
| | 5+3+2 bit | | 5 |
| Алгоритм Median Cut, 256 цветов | | | 4 |
| Алгоритм LBG, 256 цветов | | | 3 |
| Алгоритм Median Cut, 1024 цветов | | | 2 |
| Алгоритм LBG, 1024 цветов | | | 1 |

5a. Простейший и быстрейший метод квантования – равномерное квантование (англ. *uniform quantization*). Для каждого канала оставляйте указанное количество старших битов, а младшие биты просто игнорируйте.

Проведите эксперименты, оставив следующее количество старших битов:

В модели *RGB* *Red* 3 bit, *Green* 4 bit, *Blue* 3 bit.
*Red* 4 bit, *Green* 4 bit, *Blue* 2 bit.

В модели $YC_bC_r$ $Y$ 2 bit, $C_b$ 4 bit, $C_r$ 4 bit.
$Y$ 5 bit, $C_b$ 2 bit, $C_r$ 3 bit.
$Y$ 5 bit, $C_b$ 3 bit, $C_r$ 2 bit.

Для квантования в модели $YC_bC_r$ используйте преобразование изображения в это пространство и обратно в *RGB*, реализованное вами в части 4 этого задания.

5b. Палитра, близкая к оптимальной, может быть построена с помощью классического алгоритма векторного квантования LBG (Linde-Buzo-Gray), он же GLA (Generalized Lloyd Algorithm). Реализуйте этот алгоритм в вашей программе.

5c. Реализуйте в вашей программе достаточно простой и быстрый, но в то же время дающий приемлемые результаты метод квантования Median Cut. Классический вариант Median Cut – это итеративный алгоритм. После $k$ итераций будет построена палитра (кодовая книга) из $2^k$ цветов.

Описание алгоритма:

**A**. Внутри цветового куба найдите минимальный прямоугольный параллелепипед с ребрами, параллельными осям координат, который полностью вместит облако точек, соответствующих цветам, используемым в данном изображении. Обратите внимание, что у параллелепипеда 12 рёбер, по 4 ребра равной длины на каждое из трех измерений, а каждое измерение соответствует одному из цветовых каналов *R, G, B*. Инициализируйте разбиение этим прямоугольным параллелепипедом. В дальнейшем будем итеративно удваивать количество элементов разбиения. Переходим к первой итерации алгоритма.

**B**. Среди нерассмотренных элементов разбиения найдите элемент, имеющий самое длинное ребро. Для точек, оказавшихся внутри найденного элемента разбиения, рассматривается только координата (значения цветового канала), соответствующая измерению длиннейшего ребра. Например, если длиннейшее ребро это *G*, то для точек (124, 84, 21), (59, 16, 78), (97, 30, 44), (17, 14, 33) будут рассмотрены только значения 84, 16, 30, 14. Для множества этих значений найдите медиану (см. курс матем. статистики). Разделите ребро по медиане на две части. Если провести через точку медианы плоскость, перпендикулярную ребру, то она разделит рассматриваемый элемент разбиения на две части. Сохраните эти две части как элементы нового разбиения, которое будет рассматриваться на следующей итерации алгоритма.

**C**. Если на текущей итерации остались нерассмотренные элементов разбиения, то переходим к пункту **B**. Если же все элементы разбиения уже рассмотрены, то значит, каждый из них был разбит на две части, и, таким образом, количество элементов разбиения удвоилось, итерация алгоритма завершена.

**D**. Если желаемое количество элементов разбиения не достигнуто, то начинаем новую итерацию алгоритма, переходим к пункту **B**. Если желаемое количество элементов разбиения достигнуто, то фиксируем построенное разбиение цветового куба. Элемент палитры – это точка, координаты которой выбираются, как медианы соответствующего множества значений по каждому измерению для каждого элемента разбиения. Цветов палитры столько же, сколько элементов разбиения, цвета палитры и элементы разбиения связаны взаимно однозначно.

**E**. Квантование происходит следующим образом: каждый цвет рассматриваем как точку внутри цветового куба, находим элемент разбиения, в котором она находится. Цвет пикселя заменяем цветом палитры, соответствующим этому элементу разбиения.

Для эффективной реализации алгоритма рекомендуется работать не с облаком точек на каждой итерации, а заранее построить гистограмму (таблицу частот), и в дальнейшем работать с ней. Поскольку, вообще говоря, в этой таблице может быть достаточно много элементов, то в случае медленной работы алгоритма рекомендуемся перед построением гистограммы сделать равномерное пре-квантование. В этом случае размер гистограммы уменьшится, скорость работы алгоритма возрастет, а качество ухудшится незначительно.