

# Лабораторная работа №1

## Форматы изображений

### Введение

Развитие цифрового телевидения, цифровой фотографии, систем цифровой передачи видеосигналов по стандартным телефонным каналам делают все более актуальной задачу компактного представления изображений в цифровой форме. Количество бит, необходимое для хранения стандартного изображения весьма велико и ограничивает возможность использования графической информации в системах обработки и передачи информации.

Рассмотрим, например, передачу данных по телефонному каналу с помощью стандартного модема со скоростью 33600 бит/с. Если необходимо передать изображение размером 240 на 352 точек, и для представления каждой точки затрачивается 3 байта, то, нетрудно подсчитать, что на передачу одного изображения потребуется около минуты. В то же время, уже существуют и зарегистрированы в качестве международных стандартов алгоритмы сжатия изображений, позволяющие использовать телефонный канал для проведения видеоконференции.

В качестве второго примера рассмотрим запись подвижного изображения (видеосигнала) на компакт-диск. Одна секунда полноэкранный видеofilm с разрешением 480 на 720 точек при записи со скоростью 30 кадров в секунду потребует 20,736 Мбайт. Таким образом, только 31с фильма может быть записана на диск емкости 650 Мбайт. Использование современных алгоритмов сжатия подвижных изображений позволяет записать на такой компакт-диск 74-минутный фильм.

Системы сжатия изображений можно разделить на два больших класса: сжатие изображений без потери качества и сжатие с потерей качества.

При сжатии без потерь восстановленные на приемной стороне данные в точности совпадают с входным сигналом кодера. Для сжатия изображений без потерь можно использовать известные из теории информации алгоритмы кодирования источников, в частности,

- кодирование кодом Хаффмена,
- арифметическое кодирование,
- кодирование на основе алгоритмов Зива-Лемпела и другие алгоритмы.

Кодирование с потерей качества предполагает устранение из изображения некоторых избыточных компонент, которое не ведет к существенному ухудшению качества по сравнению с исходным сигналом. Сюда относятся

- методы сжатия изображения, основанные на кодировании коэффициентов различных преобразований (дискретное преобразование Фурье, дискретное косинусное преобразование, вейвлетная фильтрация и т.д.),
- методы сжатия во временной области (дифференциальное кодирование).

Для оценивания качества алгоритма сжатия необходимо принимать во внимание следующие его характеристики:

- эффективность кодирования (коэффициент сжатия),
- качество восстановленного изображения,
- сложность кодирования,
- задержка кодирования.

Эффективность кодирования обычно оценивается в битах на отсчет или в битах в секунду. Распространенным критерием качества при кодировании с потерями является отношение сигнал/шум (SNR) на выходе декодера

$$SNR = 10 \log_{10}(E_{inp} / E_n) \text{ (дБ)},$$

где  $E_{inp}$  - это энергия входного сигнала кодера, а  $E_n$  - это энергия шумового сигнала. Под энергией шумового сигнала здесь понимается энергия разности между входным сигналом кодера и выходным сигналом декодера. Отношение сигнал/шум измеряется в децибелах. В случае обработки изображений вместо SNR часто используется пиковое отношение сигнал/шум, которое определяется по формуле

$$PSNR = 10 \log_{10}((255)^2 / E_n),$$

где 255 - это максимальное десятичное значение беззнакового 8-битового числа, соответствующего одной точке черно-белого изображения.

Рассмотрим стандартные форматы, используемые для представления цветных изображений. В цифровой форме любое изображение представляется в виде двумерного поля отсчетов (точек), называемых пикселями. Большинство цветных сканеров генерируют изображения с красной, зеленой и голубой цветовыми компонентами или так называемые изображения в формате RGB. В простейшем варианте каждому пикселю сопоставляется 3 числа, характеризующие интенсивность красной, зеленой и голубой составляющих изображения в данной точке. Таким образом, изображение описывается тремя цветовыми компонентами, каждая из которых представляет собой прямоугольный массив чисел. Значение интенсивности компоненты в одной точке характеризуется целым числом из интервала 0...255 и для ее хранения отводится 1 байт. Всего изображение размера  $M \times N$  в формате RGB занимает  $3MN$  байт в памяти компьютера.

В типичных изображениях в формате RGB имеется существенная корреляция между цветными компонентами и с точки зрения сжатия изображений формат RGB является заведомо избыточным. Как известно, в стандартах телевизионного вещания используется другое представление

изображений, при котором также используются 3 компоненты сигнала, но эти три компоненты почти некоррелированы друг с другом. Компоненты R, G и B преобразуются в яркостную компоненту Y и две цветоразностных компоненты U и V, формата YUV.

Преобразование формата RGB в формат YUV выполняется по формулам

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B, \\ U &= (B - Y)0.5643 + 128, \\ V &= (R - Y)0.7132 + 128. \end{aligned}$$

Обратное преобразование выполняется по формулам

$$\begin{aligned} G &= Y - 0.714(V - 128) - 0.334(U - 128), \\ R &= Y + 1.402(V - 128), \\ B &= Y + 1.772(U - 128). \end{aligned}$$

В формате YUV компоненты слабо коррелированы. Более того, так как большая часть информации сосредоточена в яркостной компоненте, то мы теряем мало информации, если выполним децимацию (прореживание) компонент U и V с коэффициентом 2. При таком прореживании 4 соседние точки изображения, образующие квадрат размера 2\*2, описываются 4 значениями яркостных компонент Y, одним значением компоненты U и одним значением компоненты V. Каждая из цветоразностных компонент вычисляется как округленное до ближайшего целого среднее арифметическое соответствующих четырех значений рассматриваемого квадрата. Результатом является стандартный формат YUV 4:1:1, который, как правило, является входным для большинства видеокодеров. Нетрудно подсчитать, что на квадрат 2\*2 будет затрачено уже не 12, а 6 байт. Таким образом, получается сжатие в 2 раза без сколько-нибудь заметного искажения изображения.

Описанное выше представление цветных изображений в виде RGB является фактически вариантом более общей конструкции - формата BMP (от Bitmap). Этот формат, будучи одним из самых распространенных форматов хранения растровой графической информации, является стандартным для операционных систем Windows 3.x, Windows 95 и Windows NT. В формате BMP изображение может храниться как без сжатия, так и со сжатием без потерь с использованием метода кодирования длинами серий. Изображения могут быть монохромными (1 бит/пиксел) или цветными (4, 8, 16, 24 или 32 бита/пиксел).

Файл в формате BMP состоит из четырех частей:

BITMAPFILEHEADER
BITMAPINFOHEADER
RGBQUADS
Pixels

В BITMAPFILEHEADER и BITMAPINFOHEADER содержатся параметры файла и изображения, в RGBQUADS записывается

цветовая палитра, а затем хранятся собственно пикселы изображения (как индексы палитры или как величины красной, зеленой и голубой составляющей цвета).

Формат BITMAPFILEHEADER:

Название поля	Число байт	Комментарий
bfType	2	Тип файла. Должен быть "BM".
bfSize	4	Размер файла в байтах
bfReserved1	2	Зарезервировано. Должно быть 0.
bfReserved2	2	Зарезервировано. Должно быть 0.
BfOffBits	4	Расстояние в байтах от BITMAPFILEHEADER до пикселов изображения

Формат BITMAPINFOHEADER:

Название поля	Число байт	Комментарий
biSize	4	Размер структуры BITMAPINFOHEADER в байтах
biWidth	4	Ширина изображения в пикселах
biHeight	4	Высота изображения в пикселах
biPlanes	2	Число плоскостей на устройстве вывода. Должно быть 1
biBitCount	2	Число бит на пиксел (1, 4, 8, 16, 24, 32)
biCompression	4	Метод хранения пикселов (BI_RGB, BI_RLE8, BI_RLE4, BI_BITFIELDS)
biSizeImage	4	Размер изображения в байтах (Может быть 0, если biCompression=BI_RGB)
biXPelsPerMeter	4	Горизонтальное разрешение устройства вывода (в пикселах/метр)
biYPelsPerMeter	4	Вертикальное разрешение устройства вывода (в пикселах/метр)
biClrUsed	4	Число цветовых индексов в таблице цветов, которые используются в изображении
biClrImportant	4	Число цветовых индексов, которые считаются важными при выводе изображения

Если величина biHeight положительна, то изображение записано снизу-вверх и начало изображения - левый нижний угол. Если величина biHeight отрицательна, то изображение

записано сверху-вниз и начало изображения в левом нижнем углу.

Поле `biCompression` может принимать следующие значения:

`BI_RGB` (0) – формат без сжатия.

`BI_RLE8` (1) – сжатие длинами серий, 8 бит/пиксел. Каждая запись состоит из 2х байтов, в первом байте хранится число цветовых индексов в серии, во втором байте цветовой индекс.

`BI_RLE4` (2) – сжатие длинами серий, 4 бит/пиксел.

`BI_BITFIELDS` (3) – изображение хранится без сжатия, цветовая таблица состоит из трех четырехбайтовых масок для выделения красной, зеленой и голубой составляющей каждого пиксела. Этот режим используется при 16 и 32 битах/пиксел

`RGBQUADS` состоит из четверок байт `rgbBlue, rgbGreen, rgbRed, rgbReserved`, которые определяют голубую, зеленую и красную составляющую цвета. Размер массива `RGBQUADS` зависит от числа бит на пиксел и метода сжатия.

<code>biBitPerPixel</code>	Значение
1	Изображение монохромное. <code>RGBQUADS</code> содержит две четверки, определяющие цветовые компоненты "черных" и "белых" пикселов. В этом случае каждый бит массива задает один пиксел
4	Изображение содержит до 16 цветов, в <code>RGBQUADS</code> записано до 256 четверок, определяющих палитру изображения.
8	В изображении до 256 цветов. <code>RGBQUADS</code> содержит до 256 четверок, определяющих палитру изображения.
16	До $2^{16}$ цветов. Если <code>biCompression=BI_RGB</code> , то массив <code>RGBQUADS</code> пуст, на каждый пиксел изображения отводится 2 байта, в которых записаны <code>B, G, R</code> цветовые компоненты (5 бит/компоненту, старший бит двухбайтового слова не используется.) Если <code>biCompression=BI_BITFIELDS</code> , то <code>RGBQUADS</code> состоит из трех четырехбайтовых масок, определяющих <code>R, G, B</code> компоненты.
24	Если <code>biBitPerPixel=24</code> , то массив <code>RGBQUADS</code> пуст и пикселы изображения хранятся в виде троек байт <code>Blue, Green, Red</code> .
32	Аналогично <code>biBitPerPixel=16</code> , только на пиксел отводится 4 байта, три байта на <code>R, G</code> и <code>B</code> , старший байт не используется.

В настоящее время в файлах BMP изображения обычно хранятся без сжатия в формате либо 8 бит/пиксел (с палитрой) либо 24 бит/пиксел.

Пикселы изображения хранятся в файле строка за строкой. Представление каждой строки должно быть выравнено на четырехбайтовую границу. Недостающие байты заполняются нулями.

#### **ЗАДАНИЕ 1**

1. Дан файл с изображением размером 160X120 точек в формате RGB. Преобразовать его в формат YUV.
2. Выполнить децимацию компонент U и V.
3. Выполнить обратное преобразование YUV в RGB.
4. Вывести изображение на экран.
5. Оценить пиковое отношение сигнал/шум для всех компонент изображения.
6. Выполнить статистический анализ изображений в форматах YUV и RGB. Оценить энтропию значений компонент и достижимое сжатие при одномерном побуквенном кодировании компонент изображения без потерь.

#### **ЗАДАНИЕ 2**

1. Дан файл в формате BMP, 8 бит/пиксел. Прочитать файл, вывести размеры, число цветов и другие поля в заголовке изображения, выделить и напечатать массив цветов-палитру.
2. Дан файл в формате BMP, 24 бит/пиксел. Прочитать исходное изображение, растянуть его в 2 раза по высоте и ширине и записать в новый BMP-файл.
3. Дан файл в формате BMP, 24 бит/пиксел. Прочитать изображение, в каждом пикселе поменять между собой красную и голубую компоненту и записать результат в новый BMP-файл.
4. Создать 24-битовый BMP-файл, записать в него изображение, в котором зеленая компонента каждого пиксела не изменяется, красная компонента растёт снизу вверх, а зеленая - слева направо.