Кодирование изображений с преобразованием. Стандарт JPEG

5 января 2023 г. [illegible]

Кодирование с преобразованием используется, когда нужно не без потерь сжать изображение, а с потерями.

Основная идея:

1. Изображение делится на блоки n x n (8x8, 16x16 и т.д.)
2. Для каждого блока выполняется декоррелирующее преобразование. Т.е. преобразовываем блок так, чтобы уменьшить зависимости между его пикселями. Искажение в области коэффициентов преобразования должно быть то же, что и искажение в области исходного сигнала. Должна быть небольшая вычислительная сложность.
3. Квантование устраняет наименее информативные коэффициенты преобразования
4. Энтропийное кодирование применяется к квантованным коэффициентам преобразования

Разные виды преобразования:

- Прямое преобразование

$$T(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)g(x,y,u,v),$$

$$u, v = 0, 1, 2, \ldots, N-1.$$

- Обратное преобразование

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(x,y)h(x,y,u,v).$$

- Сепарабельное преобразование:

$$g(x,y,u,v) = g_1(x,u)\cdot g_2(y,v).$$

- Симметричное преобразование:

$$g(x,y,u,v) = g_1(x,y)\cdot g_1(u,v).$$

Возможные кандидаты:

- Преобразование Карунена-Лоева (Karhunen-Loeve Transform, KLT);
  - Гарантирует, что коэффициенты не коррелированы.
  - Оптимальный базис зависит от входных данных, т.е., его нужно передавать декодеру.
  - Высокая вычислительная сложность;
- Дискретное преобразование Фурье (Discrete Fourier Transform, DFT);
  - Имеет избыточные (мнимые) коэффициенты;
- Дискретное косинусное преобразование (Discrete Cosine Transform, DCT);
- Дискретное преобразование Уолша-Адамара (Walsh-Hadamard transform, WHT);

Преобразование Фурье избыточное. Еще оно дает мнимую часть, с которой не ясно, что делать, т.к. кол-во коэффициентов дублируется, их становится в 2 раза больше

Преобразование Уолша-Адамара, условно говоря, раскладывает входной блок на сумму блоков, умноженных на какой-то коэффициент.

В области преобразования коэффициент остается только на какой-то одной позиции, остальные обнуляются, и производится обратное преобразование

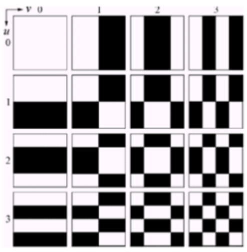
Для блока 4x4:

Преобразование Уолша-Адамара

$$g(x,y,u,v) = h(x,y,u,v) = \frac{1}{N}(-1)^{(b_i(x)p_i(u) + b_i(y)p_i(v))}, N = 2^m,$$

where $b_i(x)$ – значение бита в x на позиции i.

$$p_0(u) = b_{n-1}(u),$$
$$p_1(u) = b_{n-1}(u) + b_{n-2}(u),$$
$$p_2(u) = b_{n-2}(u) + b_{n-3}(u),$$
$$\ldots$$
$$p_{n-1}(u) = b_1(u) + b_0(u).$$



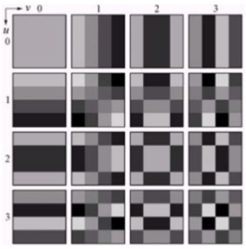
Дискретное косинусное преобразование

В нем левый верхний коэффициент - коэффициент постоянного тока, он же по факту средняя яркость

Дискретное косинусное преобразование

$$g(x,y,u,v) = h(x,y,u,v) = a(u)a(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

$$a(u) = \begin{cases} \dfrac{1}{\sqrt{N}}, u = 0 \\ \dfrac{2}{\sqrt{N}}, u \neq 0. \end{cases}$$



Стандарт JPEG

Основные этапы [7]

1. Преобразование цветового пространства из RGB24 в YCbCr 4:2:0;
2. Разбиение яркостной и цветоразностной компоненты на блоки 8×8;
3. Примерение 2-D DCT для каждого блока;
4. Квантование DCT коэффициентов;
5. DC коэффициент из текущего блока предсказывается при помощи DC коэффициента предыдущего блока и кодирование разности кодом Левенштейна с кодом Хаффмана в первой части кодового слова.
6. Сканирование AC коэффициентов в 'зигзагообразном' порядке.
7. Одномерный вектор AC кодируется длинами серий и кодом Хаффмана.

Исходный блок 8×8:

$$X = \begin{pmatrix} 168 & 161 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{pmatrix}$$

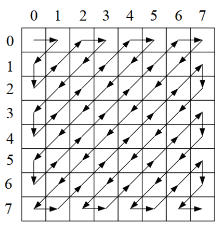
Блок после вычитания 128 и выполнения 2-D DCT:

$$Y = \begin{pmatrix} 214 & 49 & -3 & 20 & -10 & -1 & 1 & -6 \\ 34 & -25 & 11 & 13 & 5 & -3 & 15 & -6 \\ -6 & -4 & 8 & -9 & 3 & -3 & 5 & 10 \\ 8 & -10 & 4 & 4 & -15 & 10 & 6 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -5 & -1 \\ 5 & 9 & -8 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 3 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{pmatrix}$$

Блок после скалярного квантования:

$$Z = \begin{pmatrix} 13 & 4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Коэффициенты сканируются в "зигзагообразном" порядке:



- После сканирования
  $Z = \{13, 4, 3, 0, -2, 0, 1, 1, 0, 1, -1, -1, 1, 1, 0, ..., 0\}$;

- $Z = \{13, 4, 3, 0, -2, 0, 1, 1, 0, 1, -1, -1, 1, 1, 0, ..., 0\}$;
- Первый коэффициент 13 (DC coefficient) предсказывается по DC коэффициенту из предыдущего (слева) блока, затем амплитуда ошибки предсказания кодируется монотонным кодом, в котором первая часть кодируется кодом Хаффмана. Дополнительный бит используется для передачи знака для ненулевых значений.

$$n \rightarrow \underbrace{\texttt{huff(DC bits)}}_{\texttt{DC bit size}} \; \underbrace{\texttt{bin(DC)}}_{\texttt{DC}} \; \underbrace{\texttt{one bit}}_{sign}.$$

- $Z = \{13, 4, 3, 0, -2, 0, 1, 1, 0, 1, -1, -1, 1, 1, 0, ..., 0\}$;

- $Z = \{13, 4, 3, 0, -2, 0, 1, 1, 0, 1, -1, -1, 1, 1, 0, \ldots, 0\}$,
- Оставшиеся коэффициенты (AC coefficients) представляются парами [*run*, *level*], где *run* – число нулей перед ненулевым коэффициентом, *level* – значение ненулевого коэффициента. В нашем случае: $[0,4]$,$[0,3]$,$[1,-2]$,$[1,1]$,$[0,1]$,$[1,1]$,$[0,-1]$,$[0,-1]$,$[0,1]$,$[0,1]$.
- $$[run, level] \to \underbrace{\texttt{Huffman}}_{[run,\,\texttt{level bit size}]} \quad \underbrace{\texttt{bin(|level|)}}_{\texttt{level}} \quad \underbrace{\texttt{one bit}}_{sign}.$$
- Если пара [*run*,*level*] не присутвует в кодовой таблице Хаффмана, то передаётся ESC-символ, после чего *run* и *level* передаются равномерным кодом.
- End-of-block (EOB) символ передаётся кодом Хаффмана, если в "зиг-заге" далее следуют только нули.

Блок после декодирования кодом Хаффмана, обратного прохода по "зиг-загу", деквантования, обратного преобразования и прибавления 128:

$$\hat{X} = \begin{pmatrix} 171 & 160 & 149 & 149 & 158 & 166 & 166 & 162 \\ 174 & 164 & 155 & 154 & 160 & 164 & 161 & 156 \\ 171 & 164 & 157 & 156 & 158 & 158 & 151 & 145 \\ 161 & 157 & 154 & 154 & 155 & 151 & 144 & 137 \\ 156 & 155 & 155 & 156 & 156 & 152 & 145 & 140 \\ 159 & 160 & 160 & 160 & 157 & 153 & 148 & 145 \\ 161 & 161 & 160 & 156 & 150 & 144 & 141 & 139 \\ 159 & 158 & 155 & 148 & 139 & 132 & 129 & 128 \end{pmatrix}$$

При больших значенях Δ возникают хорошо видимые границы блоков:

Δ - шаг квантования

Из-за поблочного квантования появляются блоковые артефакты: возникают расхождения на границах блоков. Это можно учесть.
Известно, что расхождения возникают на границах блоков, размер блоков известен. Можно применить алгоритм, который будет усреднять граничные пиксели