МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. М.В. ЛОМОНОСОВА
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ОБЩИХ ПРОБЛЕМ УПРАВЛЕНИЯ

Дипломная работа
студента 632 группы
Садыкова И.И.

Об оптимальном восстановлении изображения по неточным данным.

Научные руководители:
доцент Демидович В.Б.
профессор Осипенко К.Ю.

Москва – 2017

Содержание

1 Введение **3**

2 Постановка задачи **5**

3 Результат **5**

4 Приложение **12**

5 Заключение **26**

6 Литература **27**

1 Введение

Одной из наиболее актуальных проблем в мире информационных технологий является проблема обработки, передачи и хранения различной информации, в том числе визуальной. В последние два десятилетия изображения стали использоваться повсеместно. Но чем качественнее картинка, тем больший объем информации для неё нужен, что усложняет процесс её хранения и передачи. Решению данной проблемы уделяется серьезное внимание. Разработано большое количество алгоритмов архивации: как видоизмененные универсальные, так и абсолютно новые алгоритмы, ориентированные только на изображения. Более того, были разработаны алгоритмы, ориентированные только на конкретный класс изображений.

В данной дипломной работе был проанализирован алгоритм сжатия JPEG, который и на сегодняшний день является одним из самых популярных и достаточно мощных алгоритмов в области сжатия изображений. При таком сжатии, изображение преобразуется из цветового пространства RGB в YCbCr. После преобразования для каналов изображения Cb и Cr, выполняется "прореживание". Далее яркостный компонент Y и отвечающие за цвет компоненты Cb и Cr разбиваются на блоки 8x8 пикселов. Каждый такой блок подвергается дискретному косинусному преобразованию (ДКП). Полученные коэффициенты ДКП квантуются и пакуются с использованием кодирования серий и кодов Хаффмана.

Процесс квантования играет ключевую роль в сжатии JPEG. Это процесс, который удаляет высокие частоты, представленные в исходном изображении. Это делается из-за того, что человеческий глаз, благодаря своему строению, более чувствителен к низким частотам, чем к высоким. То есть при удалении высоких частот убытки небольшие. Таким образом, квантование - это просто деление рабочей матрицы на Матрицу Квантования(МК) поэлементно. В стандарт JPEG включены рекомендованные МК, построенные опытным путем, но никто не гарантирует, что нельзя найти более удачные матрицы.

В данной работе мы рассмотрим множество матриц, элементами которой являются коэффициенты Фурье (матрицы, полученные после ДКП.). Мы не будем использовать МК, вместо этого мы покажем, следуя какому алгоритму можно отбросить несущественные элементы матрицы(тем самым сжать изображение) и сгладить

используемые коэффициенты. Мы также получим метод восстановления изначальной матрицы по декодированной матрице, докажем оптимальность метода и покажем, почему для данного метода оказывается достаточно использовать не все доступные для измерения коэффициенты Фурье.

Представленный подход основан на теории оптимального восстановления. Задача оптимального восстановления берет свое начало от работ самого А.Н. Колмогорова и впервые появилась в кандидатской диссертации С.А. Смоляка и получила развитие в работах C.A. Michelli, T.J. Rivlin, К.Ю. Осипенко, М.И. Стесина, Г.Г. Магарил-Ильяева. Методика иследования подобных данной работе задач развивалась в таких работах как [1]-[5]

Важная особенность подхода теории оптимального востановления заключается в том, что ставится задача о нахождении на данном классе элементов метода восстановления, являющегося наилучшим среди всех возможных.

2 Постановка задачи

Обозначим через $\mathcal{X}$ множество матриц $N * N$ вида $\begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \cdots & \vdots \\ x_{N1} & \cdots & x_{NN} \end{pmatrix} = \{x_{kj}\}$, где $N \in \mathbb{N}; x_{kj} \in \mathbb{R}; k, j = 1...N$.

Рассматривается класс матриц $W = \{\{x_{kj}\} \in \mathcal{X} : \sum\limits_{k=1}^{N}\sum\limits_{j=1}^{N}(k^{2r} + j^{2r})x_{kj}^2 \leqslant 1\}$, где $||\{x_{kj}\}||_{l_2} = \sqrt{\sum\limits_{k=1}^{N}\sum\limits_{j=1}^{N}\{x^2{}_{kj}\}}, r \in \mathbb{R}_+$.

Задача заключается в том, что на классе W необходимо восстановить матрицу $\{x_{kj}\}$ по её неточно заданным элементам $y_{kj} : |x_{kj} - y_{kj}| \leqslant \delta$, где $\delta > 0$.

Метод восстановления - это любое отображение $m : \mathcal{X} \to \mathcal{X}$.

Погрешностью метода и погрешностью оптимального восстановления соответственно назовем следующие величины:

$$e(m, \delta) = \sup_{\substack{\{x_{kj}\}\in W, \{y_{kj}\}\in\mathcal{X}, \\ |x_{kj}-y_{kj}|\leqslant\delta}} ||x_{kj} - m(y_{kj})||_{l_2}, \qquad E(\delta) = \inf_{m:\mathcal{X}\to\mathcal{X}} e(m, \delta).$$

Задача состоит в нахождении величины $E(\delta)$ и соответствующего ей оптимального метода, на котором достигается нижняя грань.

3 Результат

Расположим $k^{2r}+j^{2r}$ в порядке возрастания $k_i^{2r}+j_i^{2r} \leqslant k_{i+1}^{2r}+j_{i+1}^{2r}$ где $i = 1, ..., N^2$. В аналогичном порядке расположим элементы x_{kj} и y_{kj}, а также введем новые обозначения $y_{kj} = \widetilde{y_i}$ и $x_{kj}^2 = \widetilde{x_i}^2 = c_i$.

Заметим, что для каждого фиксированного r порядок будет свой. Для наглядности выпишем порядок первых восемьнадцати элементов для $r = 1$, а также их сумму $k^{2r}+j^{2r}$

.

$r = 1$:

k	1	1	2	2	1	3	2	3	1	4	3	2	4	3	4	1	5	2	$\dots N$
j	1	2	1	2	3	1	3	2	4	1	3	4	2	4	3	5	1	5	$\dots N$
Σ	2	5	5	8	10	10	13	13	17	17	18	20	20	25	25	26	26	29	$\dots 2N^2$
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	$\dots N^2$

Теорема 1 *Пусть*

$$p_0 = \begin{cases} \max\left(p \in [1; N^2] : \delta^2 \sum\limits_{i=1}^{p} (k_i^{2r} + j_i^{2r}) < 1\right), & \text{если } \delta^2(k_1^{2r} + j_1^{2r}) < 1 \\ 0, & \text{иначе} \end{cases},$$

и обозначим $p_0 + 1 = q_0$.

Тогда:

1) Для погрешности оптимального восстановления имеет место равенство:

$E(\delta) = \left[\frac{1}{k_{q_o}^{2r} + j_{q_o}^{2r}} \left(1 + \delta^2 \sum\limits_{i=1}^{p_o} \left((k_{q_0}^{2r} - k_i^{2r}) + (j_{q_0}^{2r} - j_i^{2r})\right)\right)\right]^{\frac{1}{2}}$.

2) Оптимальный метод восстановления имеет вид: $m(\widetilde{y_i}) = \sum\limits_{i=1}^{p_0} \frac{\left((k_{q_0}^{2r} - k_i^{2r}) + (j_{q_0}^{2r} - j_i^{2r})\right)\widetilde{y_i} e_i}{k_{q_o}^{2r} + j_{q_o}^{2r}}$

, где $e_i, i = 1, ...N^2$ *- стандартный базис в* l_2

$$(e_i, e_s) = \begin{cases} 1, s = i \\ 0, s \neq i \end{cases}, s = 1, ...N^2$$

Доказательство.

Для величины $e(m, \delta) = \sup\limits_{\substack{\{x_{kj}\} \in W, \{y_{kj}\} \in \mathcal{X}, \\ |x_{kj} - y_{kj}| \leqslant \delta}} ||x_{kj} - m(y_{kj})||_{l_2}$ получим оценку снизу. Пусть $\{x_{kj}\} \in W$ и $|x_{kj}| \leqslant \delta$, а m - произвольный метод. Тогда

$$2||x_{kj}||_{l_2} = ||x_{kj} - m(y_{kj})(0) - (-x_{kj} - m(y_{kj})(0))||_{l_2} \leqslant$$
$$\leqslant ||x_{kj} - m(y_{kj})(0)||_{l_2} + || - x_{kj} - m(y_{kj})(0)||_{l_2} \leqslant 2 \sup ||x_{kj} - m(y_{kj})(0)||_{l_2} \leqslant 2e(\delta, m) \Rightarrow$$
$$\Rightarrow ||x_{kj}||_{l_2} \leqslant e(\delta, m) \qquad (1)$$

Неравенство (1) верно $\forall m$, значит, верно и следующее:

$$\begin{cases} E(\delta) \geqslant \sup\limits_{\substack{\{x_{kj}\} \in W, |x_{kj}| \leqslant \delta, \\ k,j = 1,...,N}} ||x_{kj}||_{l_2} \end{cases} \qquad (2)$$

преобразуем правую часть (2) с помощью равенства Парсеваля и возведения в квадрат, чем получим следующую экстремальную задачу

$$\begin{cases} \sum\limits_{j=1}^{N} \sum\limits_{k=1}^{N} x_{jk}^2 \to \max, \\ \sum\limits_{j=1}^{N} \sum\limits_{k=1}^{N} (j^{2r} + k^{2r}) x_{jk}^2 \leqslant 1, \\ |x_{jk}| \leqslant \delta, \ j, k = 1, \ldots, N. \end{cases} \qquad (3)$$

Упорядочим $j^{2r} + k^{2r}$ по возрастанию: $j_i^{2r} + k_i^{2r} \leqslant j_{i+1}^{2r} + k_{i+1}^{2r}$, где $i = 1, \ldots, N^2$. В аналогичном порядке расположим x_{kj} и переименуем $x_{kj}^2 = c_i$.

Тогда задача (3) перепишется в следующем виде:

$$\begin{cases} \sum\limits_{i=1}^{N^2} c_i \to \max, \\ \sum\limits_{i=1}^{N^2} (j_i^{2r} + k_i^{2r}) c_i \leqslant 1, \ i = 1, \ldots, N^2, \\ 0 \leqslant |c_i| \leqslant \delta^2. \end{cases} \tag{4}$$

Для решения задачи (4) запишем и докажем лемму Куна-Таккера.

Для этого рассмотрим задачу минимизации: $f_0(x) \to \max$ с следующими ограничениями $f_i(x) \leqslant 0, j = 1, \ldots, n \qquad (*)$

Запишем функцию Лагранжа $\mathcal{L}(x, \lambda_1, \ldots, \lambda_n) = -f_0(x) + \sum\limits_{i=1}^{n} \lambda_i f_i(x)$

Нас интересует только условие достаточности, следовательно нам не нужна выпуклость.

Лемма Куна-Таккера 1 *Пусть $\widehat{x}$ - допустимый элемент (т.е. удовл.) задаче $(*)$, $\lambda \in \mathbb{R}^n, \widehat{\lambda_i} \geqslant 0, i = 1, ..., n$ и выполнены условия*

1) $\min \mathcal{L}(x, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) = \mathcal{L}(\widehat{x}, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n})$, (стационарности)

2) $\widehat{\lambda_i} f_i(\widehat{x}) = 0, i = 1, \ldots, n$. (дополняющей нежесткости)

Тогда $\widehat{x}$– решение задачи $(*)$

□

Пусть $\widehat{x}$ – допустимое решение, тогда $-f_0(x) \overset{(a)}{\geqslant} \mathcal{L}(x, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) \overset{(b)}{\geqslant} \min \mathcal{L}(x, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) \overset{(b)}{=} \mathcal{L}(\widehat{x}, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) \overset{(c)}{=} -f_0(\widehat{x})$

a)$f_i(x) \leqslant 0$ и $\widehat{\lambda_i} \Rightarrow \sum\limits_{i=1}^{n} \widehat{\lambda_i} f_i(x) \leqslant 0 \Rightarrow \mathcal{L}(x, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) = -f_0(x) + \sum\limits_{i=1}^{n} \widehat{\lambda_i} f_i(x) \Rightarrow \mathcal{L}(x, \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}) \leqslant -f_0(x)$.

b) По условию стационарности.

c) По условию дополнящей нежесткости.

■

Из Леммы Кунна-Таккера следует, что для решения задачи (4) достаточно найти такие $\widehat{\lambda} \geqslant 0, \widehat{\lambda}_i \geqslant 0$ и допустимую последовательность $\widehat{c} = \{\widehat{c}_i\}_{i=1}^{N^2}$ для которых $\forall c \geqslant 0$ выполняются условия стационарности и дополняющей нежесткости. Т.е. в нашем случае

$\mathcal{L}(c, \lambda, \lambda_1, \lambda_2, \ldots, \lambda_{N^2}) = \sum\limits_{i=1}^{N^2}(-1 + \lambda_i + \lambda(k_i^{2r} + j_i^{2r}))c_i$ - функция Лагранжа

а) $\sum\limits_{i=1}^{N^2}(-1+\widehat{\lambda}_i+\widehat{\lambda}(k_i^{2r} + j_i^{2r}))c_i \geqslant \sum\limits_{i=1}^{N^2}(-1+\widehat{\lambda}_i+\widehat{\lambda}(k_i^{2r} + j_i^{2r}))\widehat{c}_i$ (условие стационарности)

б) $\sum\limits_{i=1}^{N^2} \widehat{\lambda}_i(\widehat{c}_i - \delta^2) = 0$ и $\widehat{\lambda}\sum\limits_{i=1}^{N^2}((k_i^{2r} + j_i^{2r})\widehat{c}_i - 1) = 0$, (условие дополнящей нежесткости)

Очевидно, что в задаче (4) максимум достигается при $|c_i| = \delta^2$. Но, т.к. имеется ограничение $\sum\limits_{i=1}^{N^2}(j_i^{2r} + k_i^{2r})c_i \leqslant 1,\ i = 1, \ldots, N^2$, то положим

$$p_0 = \begin{cases} \max\left(p \in [1; N^2] : \delta^2 \sum\limits_{i=1}^{p}(k_i^{2r} + j_i^{2r}) < 1,\right), \text{ если } \delta^2(k_1^{2r} + j_1^{2r}) < 1 \\ 0, \text{ иначе} \end{cases},$$

и обозначим $p_0 + 1 = q_0$

Далее рассмотрим 2 случая:

1) При $p_0 > 0$ положим $\widehat{\lambda} = \frac{1}{k_{q_0}^{2r} + j_{q_0}^{2r}}$, $\widehat{\lambda}_i = \begin{cases} 1 - \frac{k_i^{2r} + j_i^{2r}}{k_{q_0}^{2r} + j_{q_0}^{2r}}, & i \in [1; p_0] \\ 0, & i \in (p_0; N^2] \end{cases}$,

$$\widehat{c}_i = \begin{cases} \delta^2, & i \in [1; p_0], \\ \dfrac{\left(1 - \delta^2 \sum\limits_{i=1}^{p_0}(k_i^{2r} + j_i^{2r})\right)}{k_{q_0}^{2r} + j_{q_0}^{2r}}, & i = q_0, \\ 0, & i \in (p_0; N^2], i \neq q_0. \end{cases}$$

Здесь $\widehat{\lambda}, \widehat{\lambda}_i$ и $\widehat{c}_i$ - неотрицательны в силу выбора q_0 и p_0.

Проверим условия стационарности и дополняющей нежесткости:

а) $\sum\limits_{i=1}^{N^2}(-1+\widehat{\lambda}_i+\widehat{\lambda}(k_i^{2r}+j_i^{2r}))c_i = \sum\limits_{i=1}^{p_0}\left(-1 + 1 - \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}} + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)c_i + \sum\limits_{i=p_0+1}^{N^2}\left(-1 + 0 + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)c_i =$

$\sum\limits_{i=p_0+1}^{N^2}\left(-1 + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)c_i \geqslant \sum\limits_{i=p_0+1}^{N^2}\left(-1 + \frac{k_i^{2r}+j_i^{2r}}{k_i^{2r}+j_i^{2r}}\right)c_i = 0$, так как $k_i^{2r} + j_i^{2r} \geqslant k_{q_0}^{2r} + j_{q_0}^{2r},\ i > p_0$.

При этом $\sum\limits_{i=1}^{N^2}(-1+\widehat{\lambda}_i+\widehat{\lambda}(k_i^{2r}+j_i^{2r}))\widehat{c}_i = \sum\limits_{i=1}^{p_o}\left(-1 + 1 - \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}} + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)\delta^2 + \sum\limits_{i=p_0+1}^{N^2}\left(-1 + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)\cdot$

$0 + \left(-1 + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)\dfrac{1 - \delta^2 \sum\limits_{i=1}^{p_0}(k_i^{2r} + j_i^{2r})}{k_{q_0}^{2r} + j_{q_0}^{2r}} = 0.$

б) 1. $\sum\limits_{i=1}^{N^2} \widehat{\lambda}_i(\widehat{c}_i - \delta^2) = 0$

$\sum\limits_{i=1}^{p_0} \widehat{\lambda}_i(\delta^2 - \delta^2) + \sum\limits_{i=p_0+1}^{N^2} 0 \cdot (\widehat{c}_i - \delta) + \left(-1 + \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)\left(\frac{1-\delta^2\sum(k_i^{2r}+j_i^{2r})}{k_{q_0}^{2r}+j_{q_0}^{2r}} - \delta^2\right) = 0$

2. $\widehat{\lambda}\sum\limits_{i=1}^{N^2}((k_i^{2r}+j_i^{2r})\widehat{c}_i - 1) = 0$

$\frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}}\left(\left(\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r})\delta^2\right) + \left(\sum\limits_{i=p_0+1}^{N^2}(k_i^{2r}+j_i^{2r})\cdot 0\right) + \left((k_{q_0}^{2r}+j_{q_0}^{2r})\frac{1-\delta^2\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r})}{k_{q_0}^{2r}+j_{q_0}^{2r}} - 1\right)\right) =$

$\frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}}\left(\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r})\delta^2 - \delta^2\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r})\right) = 0.$

Причем покажем, что $\widehat{c}_i \leqslant \delta^2$ при $q_0 \leqslant N$ В этом случае: $i = q_0 \Rightarrow q_0 = p_0 + 1$.

Предположим обратное $\frac{1-\delta^2\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r})}{k_{p_0+1}^{2r}+j_{p_0+1}^{2r}} > \delta^2 \Rightarrow 1 - \delta^2\sum\limits_{i=1}^{p_0}(k_i^{2r}+j_i^{2r}) > \delta^2(k_{p_0+1}^{2r}+j_{p_0+1}^{2r}) \Rightarrow$

$1 > \delta^2\sum\limits_{i=1}^{p_0+1}(k_i^{2r}+j_i^{2r}) \Rightarrow$ противоречие определению p_o.

2) При $p_0 = 0$, то есть $\delta^2(k_1^{2r}+j_1^{2r}) \geqslant 1$,

Положим $\widehat{\lambda} = \frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}}, \widehat{\lambda}_i = 0,\ 1 \leqslant i \leqslant N^2$

$$\widehat{c}_i = \begin{cases} \frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}},\ i = q_0 \\ 0, \quad i \neq q_0. \end{cases}$$

Для $p_0 = 0$, условия а) - б) выполнены очевидным образом

Соответственно, нами представленны $\widehat{\lambda} \geqslant 0, \widehat{\lambda}_i \geqslant 0$, допустимая $\widehat{c}$ и проверены условия стационарности и дополняющей нежесткости $\Rightarrow \widehat{c}$ – решение задачи (4) . Из чего получаем оценку снизу

$$E(\delta) \geqslant \sqrt{\widehat{\lambda} + \delta^2\sum_{i=1}^{N^2}\widehat{\lambda}_i} = \left[\frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}} + \delta^2\sum_{i=1}^{p_0}\left(1 - \frac{k_i^{2r}+j_i^{2r}}{k_{q_0}^{2r}+j_{q_0}^{2r}}\right)\right]^{\frac{1}{2}} =$$

$$= \left[\frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}} + \frac{\delta^2}{k_{q_0}^{2r}+j_{q_0}^{2r}}\sum_{i=1}^{p_0}((k_{q_0}^{2r}-k_i^{2r}) + (j_{q_0}^{2r}-j_i^{2r}))\right]^{\frac{1}{2}} =$$

$$= \left[\frac{1}{k_{q_0}^{2r}+j_{q_0}^{2r}}(1 + \delta^2\sum_{i=1}^{p_0}((k_{q_0}^{2r}-k_i^{2r}) + (j_{q_0}^{2r}-j_i^{2r})))\right]^{\frac{1}{2}}.$$

Оптимальный метод будем искать в виде $m(\widetilde{y}_i) = \sum\limits_{i=1}^{N^2} a_i\widetilde{y}_i e^i$.

Далее в целях сокращениях выкладок положим $k_i^{2r} + j_i^{2r} = \eta_i$

Тогда с помощью равенства Парсеваля квадрат задачи

$$e(m,\delta) = \sup_{\substack{\{x_{kj}\}\in W, \{y_{kj}\}\in\mathcal{X}, \\ |x_{kj}-y_{kj}|\leqslant\delta}} ||x_{kj} - m(y_{kj})||_{l_2}.$$

переписывается в виде:

$$\begin{cases} \sum\limits_{i=1}^{N^2} |\widetilde{x}_i - a_i\widetilde{y}_i|^2 \to \max \\ |\widetilde{x}_i - \widetilde{y}_i| \leqslant \delta \\ \sum\limits_{i=1}^{N^2} (k_i^{2r} + j_i^{2r})\widetilde{x}_i^{\,2} \leqslant 1. \end{cases}$$

Положим $\widetilde{z}_i = \widetilde{x}_i - \widetilde{y}_i$ и воспользуемся неравенством Коши-Буняковского (для $p_0+1 \leqslant i \leqslant N^2$ положим $a_i = 0$).

Для $i : 1 \leqslant i \leqslant p_0$:

$$|\widetilde{x}_i(1-a_i) + a_i\widetilde{z}_i|^2 = \left| \frac{\widetilde{x}_i(1-a_i)\sqrt{\widehat{\lambda}\eta_i}}{\sqrt{\widehat{\lambda}\eta_i}} + \frac{a_i\widetilde{z}_i\sqrt{\widehat{\lambda}}}{\sqrt{\widehat{\lambda}}} \right| \leqslant \left(\frac{|1-a_i|^2}{\eta_i\widehat{\lambda}} + \frac{|a_i|^2}{\widehat{\lambda}_i} \right) (|\widetilde{x}_i|^2\ \widehat{\lambda}\ \eta_i + |\widetilde{z}_i|^2\ \widehat{\lambda}_i).$$

Оценим весь функционал

$$\sum_{i=1}^{N^2} |\widetilde{x}_i - a_i\widetilde{y}_i|^2 \leqslant \sum_{i=1}^{N^2} \left(\frac{|1-a_i|^2}{\eta_i\widehat{\lambda}} + \frac{|a_i|^2}{\widehat{\lambda}_i} \right) (|\widetilde{x}_i|^2\widehat{\lambda}\eta_i + |\widetilde{z}_i|^2\widehat{\lambda}_i) \leqslant \sup_{i=1,\ldots,N^2} S_i \left(\widehat{\lambda} + \delta^2 \sum_{i=1}^{N^2} \lambda_i \right),$$

где $S_i = \left(\frac{|1-a_i|^2}{\eta_i\widehat{\lambda}} + \frac{|a_i|^2}{\xi_i} \right)$, где $\xi_i = \begin{cases} \widehat{\lambda}_i, & i \leqslant p_0 \\ 1, & N^2 \geqslant i > p_0 \end{cases}$.

Значит, метод $m(\widetilde{y}_i)$ оптимальный, если $\forall i, a_i$ удовлетворяет условию $S_i \leqslant 1$.

Для начала покажем, что a_i непусто при $i \leqslant p_0$.

$$\left| a_i - \frac{\widehat{\lambda}_i}{\widehat{\lambda}_i + \eta_i\widehat{\lambda}} \right|^2 \leqslant \frac{\eta_i\widehat{\lambda}\widehat{\lambda}_i}{\widehat{\lambda}_i + \eta_i\widehat{\lambda}} + \left(\left(\frac{\widehat{\lambda}_i}{\widehat{\lambda}_i + \eta_i\widehat{\lambda}} \right)^2 - \left(\frac{\widehat{\lambda}_i}{\lambda_i + \eta_i\widehat{\lambda}} \right) \right) =$$

$$= \frac{\widehat{\lambda}_i}{\widehat{\lambda}_i + \eta_i\widehat{\lambda}} \left(\eta_i\widehat{\lambda} + \left(\left(\frac{\widehat{\lambda}_i}{\widehat{\lambda}_i + \eta_i\widehat{\lambda}} \right) - 1 \right) \right) =$$

$$= \frac{\widehat{\lambda}_i}{(\widehat{\lambda}_i + \eta_i\widehat{\lambda})^2} (\eta_i\widehat{\lambda}(\widehat{\lambda}_i + \eta_i\widehat{\lambda}) + (\lambda_i - \lambda_i - \eta_i\widehat{\lambda})) = \frac{\widehat{\lambda}_i\eta_i\widehat{\lambda}}{(\widehat{\lambda}_i + \eta_i\widehat{\lambda})^2} (-1 + \widehat{\lambda}_i + \eta_i\lambda) = 0.$$

Следовательно $a_i = \frac{\widehat{\lambda_i}}{\widehat{\lambda_i}+\eta_i\widehat{\lambda}}$ при $i \leqslant p_0$.

Проверим, что $S_i \leqslant 1$ на $i \in (1, N^2]$

Для $i \in (p_0, N^2]$ положим $a_i = 0$. Тогда $S_i = \frac{1}{\eta_i\widehat{\lambda}} = \dfrac{1}{\eta_i\frac{1}{\eta_q}} = \frac{\eta_q}{\eta_i} \leqslant 1$, так как $\eta_i \geqslant \eta_q$ при $i \in (p_0, N^2]$.

Для $i \in (1, p_0]$

$$\frac{\left|1 - \frac{\widehat{\lambda_i}}{\widehat{\lambda_i}+\eta_i\widehat{\lambda}}\right|^2}{\eta_i\widehat{\lambda}} + \frac{\left|\frac{\widehat{\lambda_i}}{\widehat{\lambda_i}+\eta_i\widehat{\lambda}}\right|^2}{\widehat{\lambda_i}} = \frac{\left|\frac{\eta_i\widehat{\lambda_i}}{\widehat{\lambda_i}+\eta_i\widehat{\lambda}}\right|^2}{\eta_i\widehat{\lambda}} + \frac{\widehat{\lambda_i}}{(\widehat{\lambda_i}+\eta_i\widehat{\lambda})^2} =$$

$$= \frac{\eta_i\widehat{\lambda}+\widehat{\lambda_i}}{(\widehat{\lambda_i}+\eta_i\widehat{\lambda})^2} = \frac{1}{\widehat{\lambda_i}+\eta_i\widehat{\lambda}} = \frac{1}{\eta_i\frac{1}{\eta_{q_0}}+1-\frac{1}{\eta_{q_0}}\eta_i} = 1.$$

■

Следовательно, мы предоставили метод восстановления $m(\widetilde{y_i})$, при котором $E(\delta) \leqslant \sqrt{\widehat{\lambda}+\delta^2\sum\limits_{i=1}^{N^2}}$. В то же время $E(\delta) \geqslant \sqrt{\widehat{\lambda}+\delta^2\sum\limits_{i=1}^{N^2}} \Rightarrow E(\delta) = \sqrt{\widehat{\lambda}+\delta^2\sum\limits_{i=1}^{N^2}}$ и метод $m(\widetilde{y_i}) = \sum\limits_{i=1}^{N^2} a_i\widetilde{y_i}e^i$ – оптимальный.

$$m(\widetilde{y_i}) = \sum_{i=1}^{p_0}\frac{\widehat{\lambda_i}}{\widehat{\lambda_i}+(k_i^{2r}+j_i^{2r})\widehat{\lambda}}\widetilde{y_i}e_i = \sum_{i=1}^{p_0}\frac{\left(1-\frac{\eta_i}{\eta_{q_0}}\right)\widetilde{y_i}e_i}{(1-\frac{\eta_i}{\eta_{q_0}})+\eta_i\frac{1}{\eta_{q_0}}} =$$

$$= \sum_{i=1}^{p_0}\frac{-(-\eta_{q_0}+\eta_i)\frac{1}{\eta_{q_0}}\widetilde{y_i}e_i}{\frac{1}{\eta_{q_0}}(\eta_{q_0}-\eta_i+\eta_i)} = \sum_{i=1}^{p_0}\frac{\left((k_{q_0}^{2r}-k_i^{2r})+(j_{q_0}^{2r}-j_i^{2r})\right)\widetilde{y_i}e_i}{k_{q_o}^{2r}+j_{q_o}^{2r}}.$$

Важным моментом является то, что оптимальный метод восстановления, как мы видим из полученного уравнения, не использует все N^2 коэффициентов. То есть для оптимального восстановления достаточно использовать лишь часть предоставленных коэффициентов, а это значит, что остальные коэффициенты можно отбросить(занулить), так как они не несут в себе значимой информации. Это и есть сжатие с потерями.

4 Приложение

Глава 1 Подробное описание со схемами и численными примерами работы стандарта JPEG.

Глава 2 Примеры необходимых вычислений в Excel и Python для сравнения результатов полученных найденным выше оптимальным методов.

Глава 1. Компрессия изображения

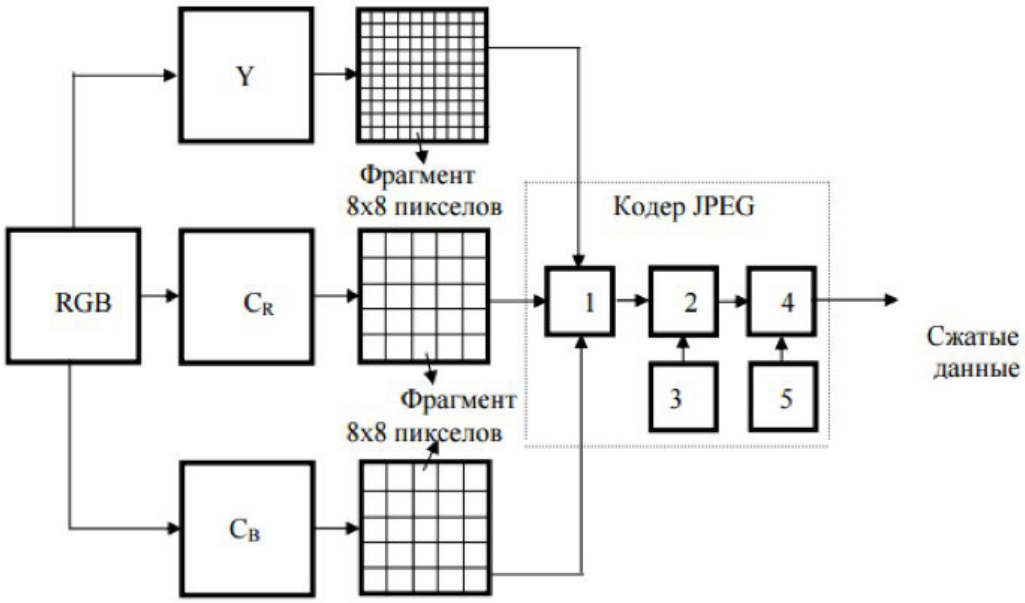
Существуют два вида алгоритмов сжатия изображений: 1) сжатие без потерь информации, 2) сжатие с потерями. Сначала для архивации изображений применялись алгоритмы без потерь, которые из-за низкой степени сжатия в дальнейшем, очевидно, перестали удовлетворять пожеланиям потребителей. В настоящее время в основном применяются алгоритмы с потерями информации. При этом в каждом конкретном случае задача сжатия решается в результате компромисса между степенью сжатия и качеством сжатых изображений. В большинстве случаев, качество сжатых изображений-это человеческая оценка. Отсюда следует, что работа над сжатием в современных алгоритмах ведется таким образом, чтобы потери в качестве результирующей картинки не были заметны невооруженному человеческому глазу. Ниже будет описан один из таких стандартов современного сжатия изображений - стандарт JPEG.

Алгоритм JPEG оперирует блоками изображения размерами 8x8, на которых яркость и цвет меняются достаточно плавно. Изображение разбивается на блоки 8x8 пикселов, которые далее кодируются один за другим. Цветное изображение может быть представлено как в формате его цветовых составляющих R, G и B, так и в формате яркостного Y и двух цветоразностных составляющих CB и CR.

В первом случае для каждого пиксела задаются значения трёх основных цветов (красного, зелёного и синего) и каждый блок 8x8 пикселов представляются тремя блоками 8x8 чисел. Кодирование данных каждого из трёх цветов выполняется аналогично, как для чёрно-белого изображения. Следует отметить, что предпочтительным является второй вариант представления цветного изображения, когда для каждого пиксела задаются значения яркости и цветоразностных сигналов. Дело в том, что в этом случае возможно уменьшение числа блоков цветоразностных составляющих путём уменьшения числа их отсчетов (пикселов) как по горизонтали, так и по вертикали в два раза. Следовательно, на каждые четыре блока пикселов яркостного сигнала будет приходиться

по одному блоку пикселов цветоразностных сигналов Cb и Cr . Поэтому по сравнению с форматом RGB полное число кодируемых блоков уменьшится в два раза, но заметного ухудшения качества изображения при этом не произойдёт, так как человеческое зрение не воспринимает искажения цвета мелких деталей изображения. Таким образом, в стандарте JPEG цветоразностные составляющие кодируются с большей погрешностью, чем яркостная составляющая. При разложении блоков изображения на основе дискретного косинусного преобразования значимыми оказываются лишь небольшое количество коэффициентов преобразованного пространства, сохранение которых требует небольшой объем по сравнению с исходным изображением. Последовательность этапов алгоритма JPEG, можно пояснить структурной схемой, представленной на рисунке

Рис. 1: Схема кодирования цветных изображений по стандарту JPEG. 1-Дискретное косинусное преобразование; 2-Квантователь; 3-Таблица квантования Q; 4-Кодер Хаффмена; 5-Таблица кодов



Этап 1. Как отмечалось выше, для цветных изображений наилучшее сжатие может быть достигнуто при раздельном кодировании его яркостного (черно-белого) Y и двух цветоразностных (Cr и Cb) составляющих, которые, формируются из пространства R, G и B следующим образом:

$$\begin{cases} Y = 0,2989R + 0,5866G + 0,1145xB \\ Cr = 0,5xR - 0,4184xG - 0,0816xB \\ Cb = -0,1688xR - 0,3312xG + 0,5xB \end{cases} \quad (5)$$

Этап 2: Каждое из составляющих цветового сигнала, полученные на первом этапе, преобразуется в цифровой сигнал. Затем цифровые составные изображения разбиваются на блоки 8x8 так, что на каждые четыре блока составляющего Y будет приходиться по одному блоку составляющих Cr и Cb, в результате чего суммарное число пикселов окажется в два раза меньше по сравнению с форматом RGB Формированные таким образом цифровые сигналы Y, Cr и Cb в дальнейшем обрабатываются в соответствующих кодерах JPEG.

Этап 3: В стандарте JPEG применяют прямое двумерное дискретное косинусное преобразование (ДКП) исходных данных, которые имеют вид матриц 8x8 пикселов сигналов Y, Cr и Cb. Коэффициенты прямого двумерного ДКП для блоков 8x8 вычисляются по следующим формулам для N=8:

$$DCT(i;j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(x;y) \cos(\frac{(2x+1)i\pi}{2N}) \cos(\frac{(2y+1)j\pi}{2N})$$

$$f(x;y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i)C(j)DCT(i;j) \cos(\frac{(2x+1)i\pi}{2N}) \cos(\frac{(2y+1)j\pi}{2N})$$

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}}, x = 0 \\ 1, x > 0 \end{cases}$$

где: x, y - координаты пикселов изображения в блоке $f(x, y)$; u,v- координаты коэффициентов ДКП в $F(u, v)$; $F(u, v)$-значения коэффициентов преобразования. В результате прямого ДКП образуется матрица 8x8, элементамы которой являются коэффициенты преобразования. В качестве примера приведём матрицы преобразуемого $f(x, y)$ и преобразованного с помощью ДКП $F(u, v)$ массивов, соответствующих яркостному составляющему Y изображения.

$$f(x,y) = \begin{pmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 165 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{pmatrix}$$

$$F(u,v) = \begin{pmatrix} 1260 & -1 & -12 & -6 & 2 & -2 & -2 & 2 \\ -22 & -17 & -6 & -3 & -3 & 0 & 1 & -1 \\ -11 & -10 & -1 & 2 & 1 & -1 & -1 & -1 \\ -7 & -2 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & -1 & -1 & 1 & 2 \\ 2 & 0 & 1 & -1 & -1 & 2 & 2 & 0 \\ -1 & -1 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 1 & -3 & -1 & 2 & 1 & -1 & -1 \end{pmatrix}$$

Следует отметить, что коэффициенты в левом верхнем углу матрицы $F(u,v)$ соответствуют низкочастнотной составляющей изображения, а в правом нижнем - высокочастотной, причём плавное изменение яркости или цвета соответствует низкочастотной составляющей изображения, а резкие - высокочастотной (их и будем отбрасывать, при больших коэффициентах сжатия побочно получим эффект Гибса).

Этап 4: Наибольший вклад при формировании большинства реальных изображений, как известно, вносят их низкочастотные составляющие, которые определяют формы и значения элементов основных объектов и фона. Высокочастотные составляющие создают резкие границы и контуры, а также мелкие детали изображения. Возможность уменьшения количества двоичных символов с помощью ДКП основана на указанных свойствах пространственно-частотного спектра реальных изображений и на ограниченной способности человеческого зрения воспринимать изменения и искажения мелкой структуры изображения. Поэтому в стандарте JPEG предусмотрено квантование коэффициентов трансформант ДКП с помощью таблицы квантования Q. Следовательно, на этом этапе производится деление коэффициентов матрицы $F(u,v)$ на соответствующие элементы таблицы квантования Q, которую для яркостного Y и цветоразностных

Cr и Cb составляющих изображения можно представить в виде матриц $[QY][QC]$ соответственно:

$$[QY] = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

$$[QC] = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

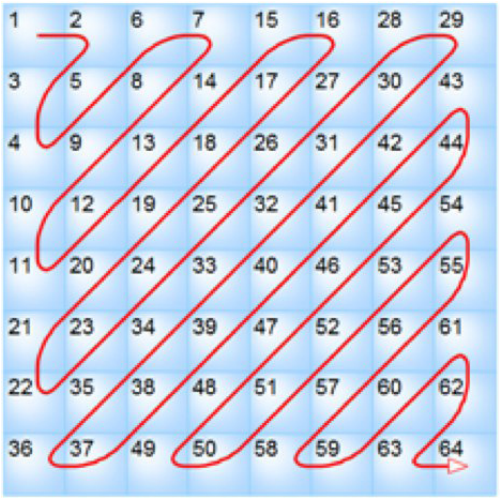
Из матриц квантования следует, что при квантовании низкочастотные составляющие квантуются более точно, чем высокочастотные. Результаты деления округляются до ближайшего целого числа. Так, например, поскольку матрица $F(u, v)$ получена путём преобразования матрицы $f(x, y)$, соответствующей яркостной составляющей изображения, то её элементы надо делить на элементы матрицы $[QY]$, в результате чего получим квантованную матрицу коэффициентов $F'(u, v)$:

$$F'(u, v) = \begin{pmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

С квантованием могут быть связаны специфические искажения на воспроизведённых изображениях, которые при потерях в низкочастотных составляющих проявляются в виде заметного разбиения изображения на блоки 8x8, а при потерях в высокочастотных составляющих - в виде так называемого "эффекта Гиббса при котором вокруг контуров с резкими перепадами образуется своеобразный "нимб".

Этап 5: На этом этапе матрица $F'(u,v)$ преобразуется в 64-элементный вектор при помощи зигзаг-сканирования, которое выполняется на основе рис 2.

Рис. 2:



$79; 0; -2; -1; -1; -1; 0; 0; -1; -1; 0; 0; 0; 0; 0; 0; 0; , ...; , 0.$

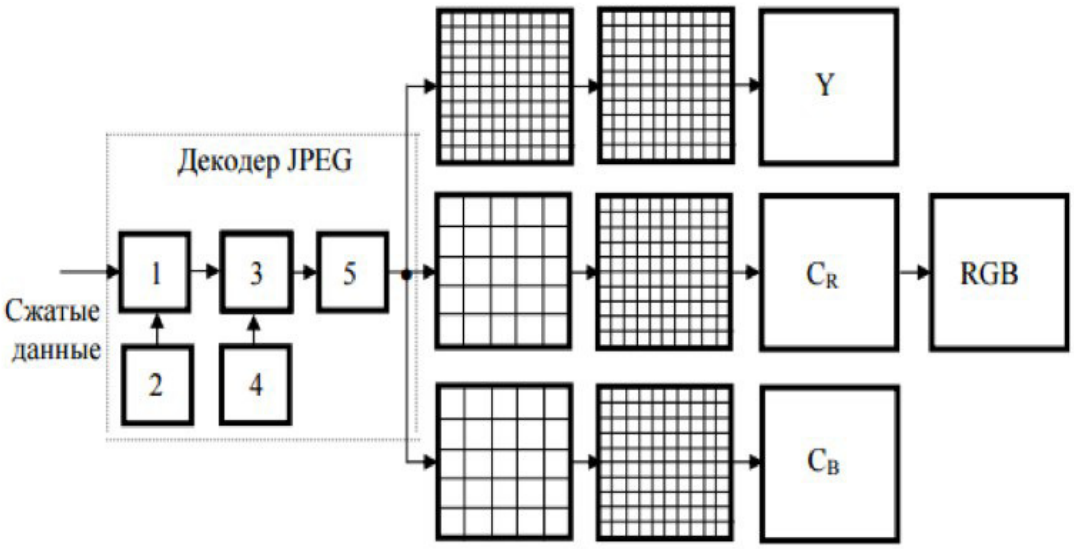
Этап 6: Из матрицы $F'(u,v)$ следует, что в результате квантования многие из коэффициентов ДКП становятся равными нулю, поэтому в векторе, полученном после сканирования, оказывается большое число нулевых коэффициентов. Каждый отличный от нуля коэффициент ДКП представляется в виде пары чисел, первое из которых показывает, сколько нулевых значений подряд прошло в последовательности перед данным ненулевым коэффициентом, а второе - значение самого квантованного коэффициента. В частности, для рассмотренного выше вектора пары чисел будут иметь следующий вид: $(0; 79), (1; -2), (0; -1), (0; -1), (0; -1), (2; -1), (0; -1), (53; 0)$. Поскольку для реальных изображений в результате квантования получается много нулевых и малых по абсолютной величине коэффициентов, поэтому такое кодирование, называемое кодированием с бегущей длиной (runlength coding), даёт значительный выигрыш, так как, во-первых, уменьшается общее количество чисел, представляющий кодируемый блок, а во вторых, уменьшается число двоичных символов для представления большинства

чисел.

Этап 7: Полученная на шестом этапе кодирования последовательность двоичных чисел затем подвергается энтропийному кодированию, при котором чаще всего применяется метод Хаффмена, который заключается в построении такого кода с переменной длиной кодового слова, что чаще встречающимся символам ставятся в соответствие более короткие кодовые слова, а реже встречающимся символам - более длинные кодовые слова. Это позволяет дополнительно уменьшить количество двоичных символов без потери информации, имеющей после квантования коэффициентов ДКП. Заметим, что иногда вместо кодирования по Хаффмену может использоваться другой вид энтропийного кодирования - арифметическое кодирование.

Воспроизведение исходного изображения (декодирование) производится в обратном направлении, которое иллюстрируется на рис 3.

Рис. 3: Схема декодирования цветных изображений по стандарту JPEG. 1-Декодер Хаффмена; 2-Таблица кодов; 3-Деквантователь; 4-Таблица деквантования Q; 5 - Обратное дискретное косинусное преобразование



Этот этап начинается обратным преобразованием слов кода Хаффмена, считываемых из файла сжатых данных, в последовательность чисел, по которым восстанавливаются значения квантованных коэффициентов ДКП. Следует отметить, что кодирование кодом Хаффмена и соответствующее декодирование не приводит к дополнительной потере информации. Затем производится деквантование коэффициентов трансформанты

и обратное ДКП, в результате чего восстанавливаются пикселы Y, Cr и Cb составляющие. Заметим, что восстановление пикселов цветоразностных Cr и Cb составляющих, пропущенных на втором этапе процесса кодирования, производится с помощью интерполяции (см. рис.2), после чего с учётом выражения 1 воспроизводятся цветовые составляющие R,G и B.

$$\begin{cases} R = Y + 1,402xCR \\ G = (0,837xY - 0,598xCR - 0,289xCB)/0,837 \\ B = Y + 1,77xCB \end{cases} \tag{6}$$

Для рассмотренного выше примера восстановленные в процессе декодирования матрицы коэффициентов трансформанты $F(u,v)$, пикселов исходного массива $f(x,y)$, полученного в результате обратного ДКП $F(u,v)$ и погрешностей восстановления пикселов $f(x,y)$ представляются в виде следующих матриц:

$$F'(u,v) = \begin{pmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\widetilde{F}(u,v) = \begin{pmatrix} 1264 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ -24 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\widetilde{f}(x,y) = \begin{pmatrix} 142 & 144 & 147 & 150 & 152 & 153 & 154 & 154 \\ 149 & 150 & 153 & 155 & 156 & 157 & 156 & 156 \\ 157 & 158 & 159 & 161 & 161 & 160 & 159 & 158 \\ 162 & 162 & 163 & 163 & 162 & 160 & 158 & 157 \\ 162 & 162 & 162 & 162 & 161 & 158 & 156 & 155 \\ 160 & 161 & 161 & 161 & 160 & 158 & 156 & 154 \\ 160 & 160 & 161 & 162 & 161 & 160 & 158 & 157 \\ 160 & 161 & 163 & 164 & 164 & 163 & 161 & 160 \end{pmatrix}$$

$$\widetilde{f}(x,y) - f(x,y) = \begin{pmatrix} 3 & 0 & -2 & -3 & -3 & -2 & -1 & -1 \\ 5 & -1 & 0 & -1 & -3 & 1 & 0 & 0 \\ 7 & 3 & -1 & -2 & 3 & 4 & 3 & 2 \\ 3 & 1 & -2 & 3 & 2 & 1 & -1 & -2 \\ 3 & 2 & 1 & 0 & -1 & 3 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & -1 & -3 \\ -2 & -2 & 0 & -1 & -1 & 3 & 1 & 0 \\ -2 & -1 & 2 & 3 & 1 & 5 & 3 & 2 \end{pmatrix}$$

В рассмотренном случае среднеквадратическая ошибка восстановления $\epsilon = 2,24$.

Глава 2

В второй главе предоставлены примеры некоторых численных вычислений в excel и python, результаты которых были использованы в первой главе приложения . На рисунках 4 и 5 представлены алгоритмы в Python, которые совершают прямое и обратное ДКП над блоками матриц. В Главе 1 матрицу, полученную в рис 4 мы обозначили за $F(u, v)$. А в рис 5 проверили алгоритм обратного ДКП, далее благодаря алгоритму в рис 5 из $\widetilde{F}(u, v)$ получаем $\widetilde{f}(x, y)$

Рис. 4: Прямое ДКП в Python

In [21]:
```
from math import sqrt, pi, cos
```

In [24]:
```
def C(x):
    if x == 0.0:
        return 1.0/sqrt(2.0)
    else:
        return 1.0

def DCT(f, N, i, j):
    answer = 0.0
    for x in range(N):
        for y in range(N):
            answer += f[x][y] * cos(((2.0 * x + 1.0) * i * pi)/(2 * N))
            * cos(((2.0 * y + 1.0) * j * pi) / (2.0 * N))
    answer *= 1 / sqrt(2 * float(N)) * C(i) * C(j)
    return answer
```

In [30]:
```
N = 8
f_massive = [[139, 144, 149, 153, 155, 155, 155, 155],
[144, 151, 153, 156, 159, 156, 156, 156],
[150, 155, 160, 163, 158, 156, 156, 156],
[159, 161, 165, 160, 160, 159, 159, 159],
[159, 160, 161, 162, 162, 155, 155, 155],
[161, 161, 161, 161, 160, 157, 157, 157],
[162, 162, 161, 163, 162, 157, 157, 157],
[162, 162, 161, 161, 163, 158, 158, 158]]
DCT_massive = [[] for i in range(N)]
for i in range(N):
    for j in range(N):
        DCT_massive[i].append(round(int(DCT(f_massive, N, i, j))))
DCT_massive
```

Out[30]:
```
[[1259.0, 0.0, -12.0, -5.0, 1.0, -1.0, -2.0, 1.0],
 [-22.0, -17.0, -6.0, -3.0, -2.0, 0.0, 0.0, -1.0],
 [-11.0, -9.0, -1.0, 2.0, 0.0, -1.0, -1.0, 0.0],
 [-7.0, -2.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0],
 [0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0],
 [2.0, 0.0, 1.0, 0.0, -1.0, 1.0, 1.0, 0.0],
 [-1.0, 0.0, 0.0, -1.0, 0.0, 1.0, 0.0, 0.0],
 [-3.0, 1.0, -3.0, -1.0, 2.0, 1.0, -1.0, -1.0]]
```

Рис. 5: Обратное ДКП в Python

In [13]:
```
from math import sqrt, pi, cos
```

In [16]:
```
def C(x):
    if x == 0.0:
        return 1.0/sqrt(2.0)
    else:
        return 1.0

def DCT(f, N, x, y):
    answer = 0.0
    for i in range(N):
        for j in range(N):
            answer += f[i][j] * C(i) * C(j) * cos(((2.0 * x + 1.0) * i * pi)/(2 * N))
            * cos(((2.0 * y + 1.0) * j * pi) / (2.0 * N))
    answer *= 1 / sqrt(2 * float(N))
    return answer
```

In [15]:
```
N = 8
f_massive = [[1259, 0, -12, -5, 1, -1, -2, 1],
 [-22, -17, -6, -3, -2, 0, 0, -1],
 [-11, -9, -1, 2, 0, -1, -1, 0],
 [-7, -2, 0, 1, 1, 0, 0, 0],
 [0, 0, 1, 1, 0, 0, 1, 1],
 [2, 0, 1, 0, -1, 1, 1, 0],
 [-1, 0, 0, -1, 0, 1, 0, 0],
 [-3, 1, -3, -1, 2, 1, -1, -1]]

DCT_massive = [[] for x in range(N)]
for x in range(N):
    for y in range(N):
        DCT_massive[x].append(int(round((DCT(f_massive, N, x, y)))))
DCT_massive
```

Out[15]:
```
[[139, 144, 149, 153, 155, 155, 155, 155],
 [144, 151, 153, 156, 159, 156, 156, 156],
 [150, 155, 160, 163, 158, 156, 156, 156],
 [159, 161, 165, 160, 160, 159, 159, 159],
 [159, 160, 161, 162, 162, 155, 155, 155],
 [161, 161, 161, 161, 160, 157, 157, 157],
 [162, 162, 161, 163, 162, 157, 157, 157],
 [162, 162, 161, 161, 163, 158, 158, 158]]
```

Напомним, что $p_0 = \begin{cases} \max\left(p \in [1; N^2] : \delta^2 \sum\limits_{i=1}^{p}(k_i^{2r} + j_i^{2r}) < 1\right), & \text{если } \delta^2(k_1^{2r} + j_1^{2r}) < 1 \\ 0, & \text{иначе} \end{cases}$

, $\widehat{c_i} = \begin{cases} \delta^2, & i \in [1; p_0], \\ \dfrac{\left(1 - \delta^2 \sum\limits_{i=1}^{p_0}(k_i^{2r} + j_i^{2r})\right)}{k_{q_0}^{2r} + j_{q_0}^{2r}}, & i = q_0, \\ 0, & i \in (p_0; N^2], i \neq q_0. \end{cases}$ и

$E(\delta) = \left[\frac{1}{k_{q_o}^{2r} + j_{q_o}^{2r}}\left(1 + \delta^2 \sum\limits_{i=1}^{p_o}\left((k_{q_0}^{2r} - k_i^{2r}) + (j_{q_0}^{2r} - j_i^{2r})\right)\right)\right]^{\frac{1}{2}}$.

В рис 6 была приведена матрица 20x20 коэффициентов $k_i^{2r} + j_i^{2r}$ в Excel. Красным помечены те коэффициенты, которые удовлетворяют неравенству $\delta^2 \sum_{i=1}^{p} (k_i^{2r} + j_i^{2r}) < 1$ и для них $\widehat{c_i} = \delta^2$. Желтым цветом отмечены пограничные точки , причем отмечено количество пограничных точек, которые нужно брать с $\widehat{c_i} = \delta^2$, а следующую одну брать с коэффициентом $\widehat{c_i} = \dfrac{\left(1 - \delta^2 \sum_{i=1}^{p_0} (k_i^{2r} + j_i^{2r})\right)}{k_{q_0}^{2r} + j_{q_0}^{2r}}$. Синим же цветом отмечены те коэффициенты, которые берем с $\widehat{c_i} = 0$. Также на рис 7 наглядно показано, как меняется граница необходимых нам коэффициентов при изменении r при фиксированном δ, на рис 8 изменение δ при фиксированном r.

В рис 9 написан алгоритм в Python, вычисляющий p_0 и оптимальную погрешность $E(\delta)$ при заданных r, N, δ, F.

Рис. 6: Матрица коэффициентов $k_i^{2r} + j_i^{2r}$ в Excel

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	5	10	17	26	37	50	65	82	101	122	145	170	197	226	257	290	325	362	401
2	5	8	13	20	29	40	53	68	85	104	125	148	173	200	229	260	293	328	365	404
3	10	13	18	25	34	45	58	73	90	109	130	153	178	205	234	265	298	333	370	409
4	17	20	25	32	41	52	65	80	97	116	137	160	185	212	241	272	305	340	377	416
5	26	29	34	41	50	61	74	89	106	125	146	169	194	221	250	281	314	349	386	425
6	37	40	45	52	61	72	85	100	117	136	157	180	205	232	261	292	325	360	397	436
7	50	53	58	65	74	85	98	113	130	149	170	193	218	245	274	305	338	373	410	449
8	65	68	73	80	89	100	113	128	145	164	185	208	233	260	289	320	353	388	425	464
9	82	85	90	97	106	117	130	145	162	181	202	225	250	277	306	337	370	405	442	481
10	101	104	109	116	125	136	149	164	181	200	221	244	269	296	325	356	389	424	461	500
11	122	125	130	137	146	157	170	185	202	221	242	265	290	317	346	377	410	445	482	521
12	145	148	153	160	169	180	193	208	225	244	265	288	313	340	369	400	433	468	505	544
13	170	173	178	185	194	205	218	233	250	269	290	313	338	365	394	425	458	493	530	569
14	197	200	205	212	221	232	245	260	277	296	317	340	365	392	421	452	485	520	557	596
15	226	229	234	241	250	261	274	289	306	325	346	369	394	421	450	481	514	549	586	625
16	257	260	265	272	281	292	305	320	337	356	377	400	425	452	481	512	545	580	617	656
17	290	293	298	305	314	325	338	353	370	389	410	433	458	485	514	545	578	613	650	689
18	325	328	333	340	349	360	373	388	405	424	445	468	493	520	549	580	613	648	685	724
19	362	365	370	377	386	397	410	425	442	461	482	505	530	557	586	617	650	685	722	761
20	401	404	409	416	425	436	449	464	481	500	521	544	569	596	625	656	689	724	761	800

23	r=1	2 * r=	2																				
24	Сумма=		27452		МАХ знач в я	271,4																	
25									След точка=	272													
26	Дельта=	0,006																					
27					Остаток=	325,8			ШТ=	1													
28	(1/Дельта)^2=	27778			Последняя точка=	53,778																	
29																							
30	Сначала выбираем R,потом Дельта, потом ищем МАХ знач в ячейке																						
31																							

Xkj		0,006
Xk0j0=		0,002670868
Xk0+1...j0+1...=		0
		0,006573728
		0,605802976
	27452	

Рис. 7: Изменение границы необходимых коэффициентов при изменении r (0.5;1;1.5;2) при фиксированном δ

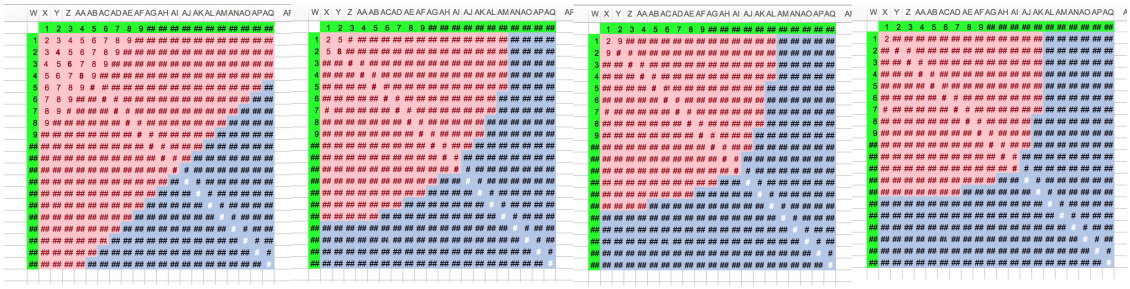


Рис. 8: Изменение границы необходимых коэффициентов при изменении δ (0.05;0,03;0,01;0,005) при фиксированном r

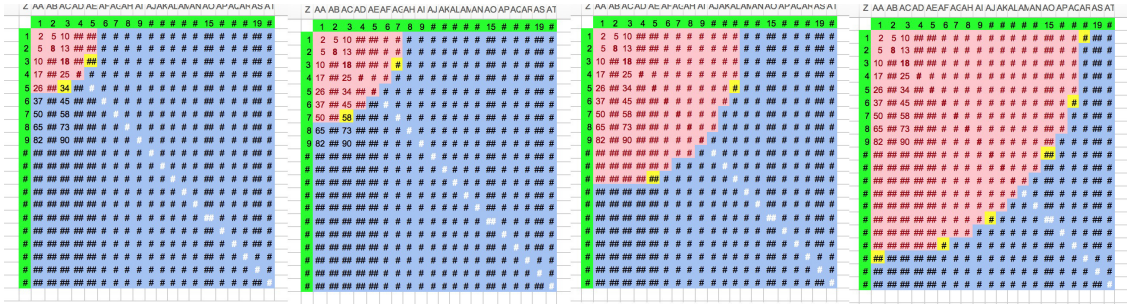


Рис. 9: Поиск p_0 и $E(\delta)$ при заданных r, N, δ, F в Python.

```
In [22]: from math import sqrt

r = 1
F = 1
N = 8
d = 0.1
a = []


for i in range(N):
    for j in range(N):
        a.append(d * d * float(pow(i + 1, r * 2) + pow(j + 1, r * 2)))
a.sort()
print a
x = 0.0
i = 0
while i < len(a) and x <= F:
    x += a[i]
    i += 1
print "p0 = ", i - 1
answer = (i - 1) * a[i - 1]
for k in range(0, i - 1):
    answer -= a[k]
print "sum = ", answer
print "E = ", sqrt((1.0 / (a[i - 1])) * (1.0 + d * d * answer))
```

```
[0.020000000000000004, 0.05000000000000001, 0.05000000000000001, 0.08000000000000002, 0.10000000000000002, 0.10000000
000000002, 0.13000000000000003, 0.13000000000000003, 0.17000000000000004, 0.17000000000000004, 0.18000000000000005,
 0.20000000000000004, 0.20000000000000004, 0.25000000000000006, 0.25000000000000006, 0.26000000000000006, 0.260000000
00000006, 0.29000000000000004, 0.29000000000000004, 0.32000000000000006, 0.34000000000000001, 0.34000000000000001, 0.37
000000000000005, 0.37000000000000005, 0.40000000000000001, 0.40000000000000001, 0.41000000000000001, 0.41000000000000001,
0.45000000000000007, 0.45000000000000007, 0.50000000000000001, 0.50000000000000001, 0.50000000000000001, 0.52000000000000
001, 0.52000000000000001, 0.53000000000000001, 0.53000000000000001, 0.58000000000000001, 0.58000000000000001, 0.61000000000
000001, 0.61000000000000001, 0.65000000000000001, 0.65000000000000001, 0.65000000000000001, 0.65000000000000001, 0.68000000
000000002, 0.68000000000000002, 0.72000000000000002, 0.73000000000000001, 0.73000000000000001, 0.74000000000000001, 0.7400
000000000001, 0.80000000000000002, 0.80000000000000002, 0.85000000000000002, 0.85000000000000002, 0.89000000000000001, 0.8
9000000000000001, 0.98000000000000002, 1.0000000000000002, 1.0000000000000002, 1.1300000000000001, 1.1300000000000001,
 1.2800000000000002]
p0 =  9
sum =  0.7
E =  2.4338301937
```

Рис. 10: Аналогично рис.8, можно посчитать руками и проверить формулу.

In [11]:
```
from math import sqrt

r = 1
F = 20
N = 8
d = 1
a = []


for i in range(N):
    for j in range(N):
        a.append(d * d * float(pow(i + 1, r * 2) + pow(j + 1, r * 2)))
a.sort()
print a
x = 0.0
i = 0
while i < len(a) and x <= F:
    x += a[i]
    i += 1
print "p0 = ", i - 1
answer = (i - 1) * a[i - 1]
for k in range(0, i - 1):
    answer -= a[k]
print "sum = ", answer
print "E = ", sqrt((1.0 / (a[i - 1])) * (1.0 + d * d * answer))
```

```
[2.0, 5.0, 5.0, 8.0, 10.0, 10.0, 13.0, 13.0, 17.0, 17.0, 18.0, 20.0, 20.0, 25.0, 25.0, 26.0, 26.0, 29.0, 29.0, 32.0,
 34.0, 34.0, 37.0, 37.0, 40.0, 40.0, 41.0, 41.0, 45.0, 45.0, 50.0, 50.0, 50.0, 52.0, 52.0, 53.0, 53.0, 58.0, 58.0, 6
1.0, 61.0, 65.0, 65.0, 65.0, 65.0, 68.0, 68.0, 72.0, 73.0, 73.0, 74.0, 74.0, 80.0, 80.0, 85.0, 85.0, 89.0, 89.0, 98.
0, 100.0, 100.0, 113.0, 113.0, 128.0]
p0 =  4
sum =  20.0
E =  1.44913767462
```

.

5 Заключение

В дипломной работе была рассмотрена задача об оптимальном восстановлении матрицы по её неточно заданным элементам на определенном классе.

Был предоставлен оптимальный метод восстановления и соответсвующая погрешность оптимального восстановления. Показан важный результат, а именно - полученный нами метод оптимального восстановления использует не все возможные коэффициенты матрицы, а лишь их часть, тем самым позволяя отбросить несущественные элементы. Что имеет связь с прикладной задачей сжатия изображения, а именно с этапом квантования матрицы, полученной после ДКП.

В приложении к диплому:

1) Рассмотрен пример стандарта JPEG.

2) Поэтапно разобран численный пример кодирования и декодирования матрицы.

3) Представлен скрин работы в Excel, в котором наглядно показано какие элементы матрицы важны, а какие можно отбросить (согласно полученному оптимальному методу восстановления). Также показано как изменяется множество необходимых элементов в зависимости от изменения параметров r и δ.

4) Представлены скрины работы алгоритмов в Python, которые выполняют: а) прямое и обратное ДКП (предоставлен численный пример) б) расчет погрешности оптимального восстановления $E(\delta)$ и точки p_0.

6 Литература

[1] Магарил-Ильяев Г. Г., Осипенко К. Ю. Как наилучшим образом восстановить функцию по неточно заданному спектру? Матем. заметки, 2012, том 92, выпуск 1, страницы 59 - 67

[2[Магарил-Ильяев Г. Г., Осипенко К. Ю. О наилучшем гармоническом синтезе периодических функции?. Фундамент. и прикл. матем., 2013, том 18, выпуск 5, страницы 155?174

[3[Н.Д.Выск. О решении волнового уравнения при неточно заданных коэффициентах Фурье функции, задающей начальную форму струны, Владикавказский мат. журн., 2006, 8, вып. 4, 12-17.

[4[Магарил-Ильяев Г. Г., Осипенко К. Ю. Оптимальное восстановление значений функций и их производных на прямой по неточно заданному преобразованию Фурье, Мат. сб., 2004, 195, №10, 67-82.

[5] Магарил-Ильяев Г. Г., Осипенко К. Ю. Оптимальное восстановление операторов по неточной информации. Итоги науки. Южный федеральный округ. Математический форум. Том 2. "Исследования по выпуклому анализу Владикавказ, 2009, с. 158-192.

[6] TERMINAL EQUIPMENT AND PROTOCOLS FOR TELEMATIC SERVICES https://www.w3.org/Graphics/JPEG/itu-t81.pdf

[7] Ватолин Д.С. Методы сжатия изображений. Интернет Университет информационных