МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«_5_» _10_____ 2023 г

Сжатие изображений по стандарту JPEG

Методические указания к лабораторной работе
для студентов, обучающихся по направлению 09.04.01 «Информатика и вычислительная техника»

Курск 2023

УДК 681.3

Составители: Е.Г.Анпилогов, С.И.Егоров

Рецензент
Доктор технический наук, профессор кафедры ВТ Юго-Западного государственного университета *В.С.Титов*

Сжатие изображений по стандарту JPEG: методические указания к лабораторной работе / Юго-Зап. гос. ун-т; сост. Е.Г.Анпилогов, С.И.Егоров. Курск, 2023. 30 с. Табл. 1, Ил. 7. Библиогр. 6 назв.

Излагаются методические указания по выполнению лабораторной работы на персональной ЭВМ. Изучается процедура сжатия изображений по стандарту JPEG.

Предназначены для студентов, обучающихся по направлению 09.04.01 - "Информатика и вычислительная техника".

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.
Усл. печ. л. Уч.-изд. л. Тираж 20 экз. Заказ Бесплатно.

Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Оглавление

1. Цель работы……4
2. Подготовка к работе……..4
3. Процедуры сжатия и восстановления изображений по стандарту JPEG…….4
4. Алгоритмы дискретно-косинусного преобразования……..12
5. Программа Coding JPEG……18
6. Формат файла DCT……..……22
7. Указания к выполнению предварительного домашнего задания…….…...28
8. Порядок выполнения работы……..28
9. Содержание отчета……...29
10. Вопросы для самопроверки…….30
11. Библиографический список…….30

1. Цель работы

Изучение процедур сжатия и восстановления изображений по стандарту JPEG.

2. Подготовка к работе

В процессе подготовки к лабораторной работе необходимо изучить:

- основные этапы процедур сжатия и восстановления изображений по стандарту JPEG и критерии качества восстановления;
- алгоритмы выполнения дискретно-косинусного преобразования;
- рабочее задание и методические указания к его выполнению.

3. Процедуры сжатия и восстановления изображений по стандарту JPEG

3.1. Критерии оценки алгоритмов сжатия изображений

К алгоритмам сжатия предъявляются следующие противоречивые требования:

1. **Высокая степень компрессии**. Заметим, что далеко не для всех приложений актуальна высокая степень компрессии. Кроме того, некоторые алгоритмы дают лучшее соотношение качества к размеру файла при высоких степенях компрессии, однако проигрываю другим алгоритмам при низких степенях.
2. **Высокое качество изображений.** Выполнение этого требования напрямую противоречит выполнению предыдущего...
3. **Высокая скорость компрессии.** Это требование для некоторых алгоритмов с потерей информации является взаимоисключающим с первыми двумя. Интуитивно понятно, что чем больше времени мы будем анализировать изображение, пытаясь получить наивысшую степень компрессии, тем лучше будет результат. И, соответственно, чем меньше мы времени потратим на компрессию (анализ), тем ниже будет качество изображения и больше его размер.
4. **Высокая скорость декомпрессии.** Достаточно универсальное требование, актуальное для многих приложений. Однако можно привести примеры приложений, где время декомпрессии далеко не критично.
5. **Масштабирование изображений.** Данное требование подразумевает легкость изменение размеров изображения до размеров окна активного приложения. Дело в том, что одни алгоритмы позволяют легко масштабировать изображение прямо во время декомпрессии, в то время как другие не только не позволяют легко масштабировать, но и увеличивают вероятность появления

неприятных артефактов после применения стандартных алгоритмов масштабирования к декомпрессированному изображению. Например, можно привести пример “плохого” изображения для алгоритма JPEG— это изображения с достаточно мелким регулярным рисунком (пиджак в мелкую клетку). Характер вносимых алгоритмом JPEG искажений таков, что уменьшение или увеличение изображения может дать неприятные эффекты.

6. **Возможность показать огрубленное изображение (низкого разрешения),** использовав только начало файла. Данная возможность актуальна для различного рода сетевых приложений, где перекачивание изображений может занять достаточно большое время, и желательно, получив начало файла, корректно показать preview. Заметим, что примитивная реализация указанного требования путем записывания в начало изображения его уменьшенной копии заметно ухудшит степень компрессии.

7. **Устойчивость к ошибкам.** Данное требование означает локальность нарушений в изображении при порче или потере фрагмента передаваемого файла. Данная возможность используется при широковещании (broadcasting— передача по многим адресам) изображений по сети, то есть в тех случаях, когда невозможно использовать протокол передачи, повторно запрашивающий данные у сервера при ошибках. Например, если передается видеоряд, то было бы неправильно использовать алгоритм, у которого сбой приводил бы к прекращению правильного показа всех последующих кадров. Данное требование противоречит высокой степени архивации, поскольку интуитивно понятно, что мы должны вводить в поток избыточную информацию. Однако для разных алгоритмов объем этой избыточной информации может существенно отличаться.

8. **Небольшая стоимость аппаратной реализации. Эффективность программной реализации.** Данные требования к алгоритму реально предъявляют не только производители игровых приставок, но и производители многих информационных систем. Так, декомпрессор фрактального алгоритма очень эффективно и коротко реализуется с использованием технологии MMXи распараллеливания вычислений, а сжатие по стандарту CCITT Group 3 легко реализуется аппаратно.

Очевидно, что для конкретной задачи нам будут очень важны одни требования и менее важны (и даже абсолютно безразличны) другие.

Характеристики алгоритма относительно некоторых сформулированных только что требований зависят от конкретных условий, в которые будет поставлен алгоритм. Так, *степень компрессии* зависит от того, на каком классе изображений алгоритм тестируется. Аналогично, *скорость компрессии* нередко зависит от того, на какой платформе реализован алгоритм. Преимущество одному алгоритму перед другим может дать, например, возможность использования в вычислениях алгоритма технологий нижнего уровня, типа MMX, а это возможно далеко не для всех алгоритмов. Так JPEG существенно выигрывает от применения технологии MMX, а LZW нет. Кроме того, нам

придется учитывать, что некоторые алгоритмы распараллеливаются легко, а некоторые нет.

Таким образом, **невозможно составить универсальное сравнительное описание известных алгоритмов.** Это можно сделать только для типовых классов приложений при условии использования типовых алгоритмов на типовых платформах. Однако такие данные быстро устаревают.

Так, например, еще три года назад, в 1994, интерес к **показу огрубленного изображения**, используя только начало файла (требование 6), был чисто абстрактным. Реально эта возможность практически нигде не требовалась и класс приложений, использующих данную технологию, был крайне невелик. С взрывным распространением Internet, который характеризуется передачей изображений по сравнительно медленным каналам связи, использование Interlaced GIF (алгоритм LZW) и Progressive JPEG (вариант алгоритма JPEG), реализующих эту возможность, резко возросло. То, что новый алгоритм (например, wavelet) поддерживает такую возможность, существеннейший плюс для него сегодня.

В то же время сегодня такому требованию, как **устойчивость к ошибкам,** уделяется сравнительно мало внимания . Можно предположить, что в скором времени (через 5-10 лет) с распространением широковещания в сети Internet для его обеспечения будут использоваться именно алгоритмы устойчивые к ошибкам, даже не рассматриваемые в сегодняшних статьях и обзорах.

Со всеми сделанными выше оговорками, выделим несколько наиболее важных критериев сравнения алгоритмов компрессии.

1. **Худший, средний и лучший коэффициенты сжатия.** То есть доля, на которую возрастет изображение, если исходные данные будут наихудшими; некий среднестатистический коэффициент для того класса изображений, *на который ориентирован алгоритм;* и, наконец, лучший коэффициент. Последний необходим лишь теоретически, поскольку показывает степень сжатия наилучшего (как правило, абсолютно черного) изображения, иногда фиксированного размера.

2. **Класс изображений**, на который ориентирован алгоритм. Иногда указано также, почему на других классах изображений получаются худшие результаты.

3. **Симметричность**. Отношение характеристики алгоритма кодирования к аналогичной характеристике при декодировании. Характеризует ресурсоемкость процессов кодирования и декодирования.

4. **Есть ли потери качества?** И если есть, то за счет чего изменяется коэффициент архивации? Дело в том, что у большинства алгоритмов сжатия с потерей информации существует возможность изменения коэффициента сжатия.

5. **Характерные особенности алгоритма** и изображений, к которым его применяют. Здесь могут указываться наиболее важные для алгоритма свойства, которые могут стать определяющими при выборе алгоритма.

До настоящего времени не найден адекватный критерий оценки потерь качества изображения. А теряется оно постоянно — при оцифровке, при переводе в ограниченную палитру цветов, при переводе в другую систему цветопредставления для печати, и, что для нас особенно важно, при архивации с потерями. Можно привести пример простого критерия: среднеквадратичное отклонение значений пикселов (L_2 мера, или root mean square — RMS):

$$d(x,y) = \sqrt{\frac{\sum_{i=1,j=1}^{n,n}\left(x_{ij} - y_{ij}\right)^2}{n^2}} \qquad (1)$$

По нему изображение будет сильно испорчено при понижении яркости всего на 5% (глаз этого не заметит — у разных мониторов настройка яркости варьируется гораздо сильнее). В то же время изображения со “снегом” — резким изменением цвета отдельных точек, слабыми полосами или “муаром” будут признаны “почти не изменившимися”. Свои неприятные стороны есть и у других критериев.

Рассмотрим, например, максимальное отклонение:

$$d(x,y) = \max_{i,j}\left|x_{ij} - y_{ij}\right| \qquad (2)$$

Эта мера, как можно догадаться, крайне чувствительна к биению отдельных пикселов. Т.е. во всем изображении может существенно измениться только значение одного пиксела (что практически незаметно для глаза), однако согласно этой мере изображение будет сильно испорчено.

Мера, которую сейчас используют на практике, называется мерой отношения сигнала к шуму (peak-to-peak signal-to-noise ratio — PSNR).

$$d(x,y) = 10 \cdot \log_{10} \frac{255^2 \cdot n^2}{\sum_{i=1,j=1}^{n,n}\left(x_{ij} - y_{ij}\right)^2} \qquad (3)$$

Данная мера, по сути, аналогична среднеквадратичному отклонению, однако пользоваться ей несколько удобнее за счет логарифмического масштаба шкалы. Ей присущи те же недостатки, что и среднеквадратичному отклонению.

Лучше всего потери качества изображений оценивают наши глаза. Отличной считается архивация, при которой невозможно на глаз различить первоначальное и разархивированное изображения. Хорошей — когда сказать, какое из изображений подвергалось архивации, можно только сравнивая две находящихся рядом картинки. При дальнейшем увеличении степени сжатия, как правило, становятся заметны побочные эффекты, характерные для данного алгоритма. На практике, даже при отличном сохранении качества, в изображение могут быть внесены регулярные специфические изменения.

Поэтому алгоритмы архивации с потерями не рекомендуется использовать при сжатии изображений, которые в дальнейшем собираются либо печатать с высоким качеством, либо обрабатывать программами распознавания образов. Неприятные эффекты с такими изображениями могут возникнуть даже при простом масштабировании изображения.

3.2. Алгоритм JPEG

JPEG— один из самых популярных и достаточно мощных алгоритмов сжатия изображений. Практически он является стандартом де-факто для полноцветных изображений. Алгоритм разработан группой экспертов в области фотографии JPEG специально для сжатия 24-битных изображений. JPEG— Joint Photographic Expert Group — подразделение в рамках ISO — Международной организации по стандартизации. Название алгоритма читается ['jei'peg].

Алгоритм предусматривает разбиение изображения на блоки 8x8 пикселов. К каждому блоку применяют дискретное косинусное преобразование (ДКП), получая в результате матрицу коэффициентов ДКП. Для восстановления исходного изображения к матрице коэффициентов необходимо применить обратное преобразование.

В каждом блоке 8x8 пикселов яркость и цвет меняются сравнительно плавно. Вследствие этого, при разложении блока в двойной ряд по косинусам (см. формулы ниже) значимыми оказываются только первые коэффициенты. Таким образом, сжатие в JPEG осуществляется за счет плавности изменения цветов в изображении.

ДКП раскладывает изображение по амплитудам некоторых частот, таким образом, при преобразовании получается матрица, в которой многие коэффициенты либо близки, либо равны нулю. Кроме того, человеческая система цветового восприятия слабо распознает определенные частоты. Поэтому можно аппроксимировать некоторые коэффициенты более грубо без заметной потери качества изображения.

Для этого используется квантование коэффициентов (quantization). В самом простом случае — это арифметический побитовый сдвиг вправо. При этом преобразовании теряется часть информации, но могут достигаться большие коэффициенты сжатия.

Характеристики алгоритма JPEG:

Коэффициенты компрессии: 2-200 (Задается пользователем).

Класс изображений: Полноцветные 24 битные изображения, или изображения в градациях серого без резких переходов цветов (фотографии).

Симметричность: 1:1

Характерные особенности: В некоторых случаях, алгоритм создает “ореол” вокруг резких горизонтальных и вертикальных границ в изображении (эффект Гиббса). Кроме того, при высокой степени сжатия изображение распадается на блоки 8x8 пикселов.

3.3. Этапы работы алгоритма

Шаг 1.

Разбиваем исходное изображение на матрицы 8x8 для каждого из 3-х цветов структуры RGB (красный (Red), зеленый (Green) и синий (Blue)).

Пример такого разбиения для изображения с вертикальной разрешающей способностью 576 строк и горизонтальной разрешающей способностью 720 показан на рисунке 1.

Выборкам внутри каждого блока 8x8 назначены горизонтальные индексы x со значениями от 0 до 7, и вертикальные индексы y со значениями от 0 до 7. Полученный 8x8 массив принято называть минимальным блоком данных (MDU).

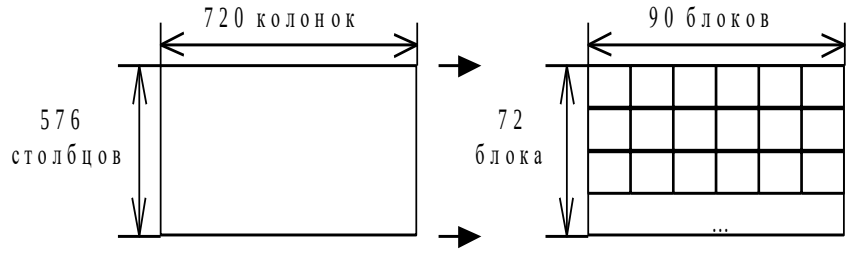

Рис. 1. Разбиение изображения на блоки

Шаг 2.

Сдвиг уровня для получения чисел различных по знаку. Входные данных точности P, перед кодированием прямым ДКП сдвигаются путем вычитания $2^{(P-1)}$. В нашем случае для 8 разрядных чисел (0..255) это достигается вычитанием $2^{(8-1)}=128$.

Шаг 3.

Применяем ДКП к каждому MDU в соответствии со следующими формулами.

Уравнение, определяющее ПДКП 8x8 блока, имеет вид:

$$F(u,v) = \frac{1}{4} \cdot c(u) \cdot c(v) \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} \left[f(x,y) \cdot \cos\left(\frac{(2 \cdot x + 1) \cdot \pi}{16} \cdot u \right) \cdot \cos\left(\frac{(2 \cdot y + 1) \cdot \pi}{16} \cdot v \right) \right] \quad (4)$$

где $f(x,y)$ - значения смещенных отсчетов на входе в процедуру ПДКП;
$F(u,v)$ - значения коэффициентов ДКП.

Уравнение, определяющее ОДКП 8x8 блока, имеет вид

$$f(x,y) = \frac{1}{4} \cdot \sum_{u=0}^{7} \sum_{v=0}^{7} \left[c(u) \cdot c(v) \cdot F(u,v) \cdot \cos\left(\frac{(2 \cdot x + 1) \cdot \pi}{16} \cdot u \right) \cdot \cos\left(\frac{(2 \cdot y + 1) \cdot \pi}{16} \cdot v \right) \right] \quad (5)$$

где $c(u)=\frac{1}{\sqrt{2}}$, для u=0;

$c(u)=1$, для u≠0;

$c(v)=\frac{1}{\sqrt{2}}$, для v=0;

$c(v)=1$, для v≠0.

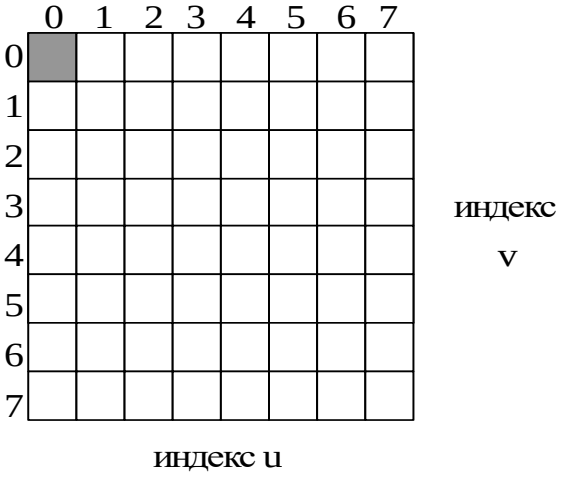

Рис. 2. Расположение DCT коэффициентов в матрице

При этом получается матрица, в которой коэффициенты в левом верхнем углу соответствуют низкочастотной составляющей изображения, а в правом нижнем - высокочастотной. Позиция коэффициента, соответствующего постоянной составляющей (DC), на рисунке 2 заштрихована, позиции AC коэффициентов не заштрихованы. Горизонтальные "частоты" в матрице

увеличиваются слева направо, вертикальные "частоты" увеличиваются сверху вниз.

Шаг 4.

Производим квантование. В принципе это просто деление MDU на матрицу квантования поэлементно. Для каждой компоненты (R, G и B), в общем случае, задается своя матрица квантования q[u,v] (далее МК).

$$MDUq[u, v] = IntegerRound\left(\frac{MDU[u, v]}{q[u, v]}\right)$$

На этом шаге осуществляется управление степенью сжатия, и происходят самые большие потери. Понятно, что, задавая МК с большими коэффициентами, мы получим больше нулей и, следовательно, большую степень сжатия.

В стандарт JPEG включена рекомендованная МК, построенная опытным путем. Матрицы для большего или меньшего коэффициентов сжатия получают путем умножения исходной матрицы на некоторое число gamma.

С квантованием связаны и специфические эффекты алгоритма. При больших значениях коэффициента gamma потери в низких частотах могут быть настолько велики, что изображение распадется на квадраты 8x8. Потери в высоких частотах могут проявиться в так называемом «эффекте Гиббса», когда вокруг контуров с резким переходом цвета образуется своеобразный «нимб».

Шаг 5.

Переводим матрицу 8x8 в 64-элементный вектор при помощи «зигзаг»-сканирования, т.е. берем элементы с индексами (0,0), (0,1), (1,0), (2,0)...

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$a_{0,6}$	$a_{0,7}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$a_{1,6}$	$a_{1,7}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,0}$			
$a_{3,0}$	$a_{3,0}$	$a_{3,0}$	$a_{3,0}$				
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$					
$a_{5,0}$	$a_{5,1}$						
$a_{6,0}$	$a_{6,1}$						
$a_{7,0}$	$a_{7,1}$						

Рис. 3. Структура «зигзаг»-сканирования

Таким образом, в начале вектора мы получаем коэффициенты матрицы, соответствующие низким частотам, а в конце — высоким.

Шаг 6.

Свертываем вектор с помощью алгоритма группового кодирования. При этом получаем пары типа (пропустить, число), где «пропустить» является счетчиком пропускаемых нулей, а «число» — значение, которое необходимо поставить в следующую ячейку. Так, вектор 42 3 0 0 0 -2 0 0 0 0 1 ... будет свернут в пары (0,42) (0,3) (3,-2) (4,1) ... .

Шаг 7.

Свертываем получившиеся пары кодированием по Хаффману с фиксированной таблицей.

Процесс восстановления изображения в этом алгоритме полностью симметричен. Метод позволяет сжимать некоторые изображения в 10-15 раз без серьезных потерь.

4. Алгоритмы дискретно-косинусного преобразования

Непосредственное вычисление ПДКП и ОДКП по формулам (4) и (5), даже если предварительно вычислить все необходимые произведения косинусов, при программной реализации требует больших вычислительных затрат. Так, только для одного блока 8x8 отсчетов следует выполнить для ПДКП 64·64=4096 умножений вещественных чисел, а также 64·63=4032 сложений.

Поэтому необходимо использовать методы ускоренного выполнения двумерного ДКП. Эти методы можно объединить в три основные группы:

1. сведение двумерного ДКП к одномерным ДКП небольшой длины;
2. сведение двумерного ДКП к двумерным ДКП меньшей длины;
3. сведение двумерного ДКП к дискретному преобразованию Фурье с использованием многочисленных быстрых алгоритмов выполнения последнего.

4.1. Разложение двумерного ДКП на одномерные.

Рассмотрим более подробно первую группу методов. Эти методы сводятся к разложению двумерного 64-точечного преобразования (для ДКП базовой системы) на одномерные 8-точечные преобразования, для выполнения которых в свою очередь можно использовать быстрые методы.

Преобразуем исходное уравнение ПДКП (4) следующим образом:

$$F(u,v)=\sum_{x=0}^{7}\left[\frac{1}{2}\cdot c(u)\cdot\cos\left(\frac{(2\cdot x+1)\cdot\pi}{16}\cdot u\right)\cdot W_x(v)\right] \qquad (6)$$

где $$W_x(v)=\sum_{y=0}^{7}\left[\frac{1}{2}\cdot c(v)\cdot\cos\left(\frac{(2\cdot y+1)\cdot\pi}{16}\cdot v\right)\cdot f(x,y)\right] \qquad (7)$$

Уравнения (6) и (7) представляют собой одномерные ДКП на 8 точек. Таким образом для получения коэффициентов F(u,v) по формулам (6) и (7) необходимо сначала восемь раз выполнить одномерное ДКП на восемь точек по строкам блока отсчетов для получения массива векторов $W_x(v)$ (по формуле (7)). Массив промежуточных результатов $W_x(v)$ можно записать в виде матрицы W, строки которой представляют собой результаты одномерного ДКП над строками блока отсчетов. Тогда коэффициенты F(u,v) получаются после выполнения восьми одномерных восьмиточечных ДКП над столбцами матрицы W.

Вычисление двумерного ДКП через последовательность одномерных иллюстрируется рисунком 4.

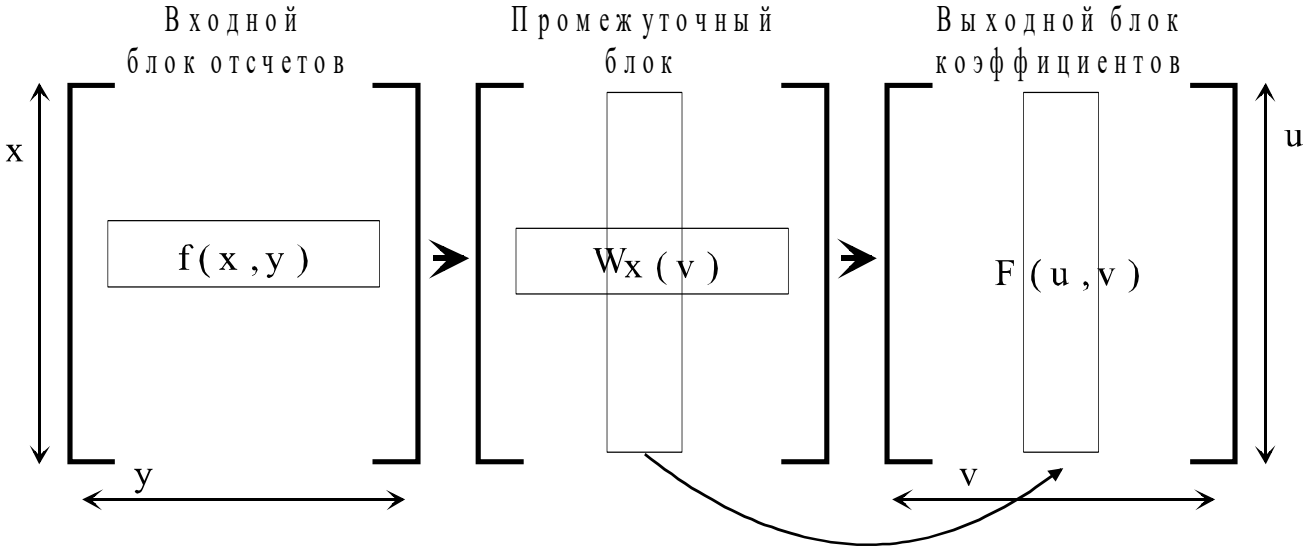

Рис. 4. Разложение двумерного ДКП на последовательность одномерных

Общее число умножений и сложений для выполнения ДКП уменьшается в результате до 1024 и 896 соответственно. Скорость вычисления ПДКП блока отсчетов по описанному только что методу (сокращенное название метода - метод строк и столбцов) в четыре раза выше, чем по формуле (4).

ОДКП блока коэффициентов также можно выполнить по вышеописанному методу.

Дальнейшее ускорение ДКП возможно с использованием быстрых одномерных дискретно-косинусных преобразований. При этом восьмиточечное ДКП представляется в виде последовательности преобразований меньшей длины. На этом принципе построены алгоритмы Ли [3] и Хоу [4], которые содержат значительно меньшее число умножений, чем алгоритм реализации ДКП непосредственно по формуле (7).

Оба этих алгоритма содержат одинаковое число умножений, но алгоритм Хоу осуществляет ДКП с меньшей погрешностью. Рассмотрим алгоритм Хоу подробнее.

4.2. Алгоритм Хоу

Запишем одномерное восьмиточечное ДКП в следующем виде:

$$X(k) = \frac{1}{2} \cdot c(k) \cdot \sum_{n=0}^{7} \left[x(n) \cdot \cos\left(\frac{(2 \cdot n + 1) \cdot \pi}{16} \cdot k \right) \right] \qquad (8)$$

где c (k) определяются аналогично c(u) или c(v) в формуле (4) и k = 0,...,7. Для упрощения введем переменные X'(k)=2*X·(k)/c(k).

Используя отображение в соответствии с [5,6]:

x'(n) = x(2n) и x'(7-n) = x(2n+1), n=0,...,3,

получим

$$X'(k) = \sum_{n=0}^{7} \left[x'(n) \cdot \cos\left(\frac{(4 \cdot n + 1) \cdot \pi}{16} \cdot k \right) \right], \ \text{k = 0,..,7.} \qquad (9)$$

Тогда коэффициенты X' с четными индексами могут быть вычислены по формулам:

$$X'(2 \cdot k) = \sum_{n=0}^{3} \left[x'(n) \cdot \cos\left(\frac{(4 \cdot n + 1) \cdot \pi}{8} \cdot k \right) \right] + \sum_{n=0}^{3} \left[x'(n+4) \cdot \cos\left(\frac{(4 \cdot n + 1 + 16) \cdot \pi}{8} \cdot k \right) \right] =$$

$$= \sum_{n=0}^{3} \left[\left(x'(n) + x'(n+4) \right) \cdot \cos\left(\frac{(4 \cdot n + 1) \cdot \pi}{8} \cdot k \right) \right] \qquad (10)$$

Последнее выражение представляет собой четырехточечное ДКП над суммами x(n) + x(n+4) (сравните с (9)).

Для нечетных X' легко получить следующее выражение:

$$X'(2\cdot k+1)=\sum_{n=0}^{3}\left[\left(x'(n)-x'(n+4)\right)\cdot\cos\left(\frac{(4\cdot n+1)\cdot\pi}{16}\cdot(2\cdot k+1)\right)\right] \qquad (11)$$

Используя тригонометрическое тождество:

$$\cos[(2\cdot k+1)\cdot q] = 2\cdot\cos(q)\cdot\cos(2\cdot k\cdot q) - \cos[(2\cdot k-1)\cdot q],$$

где $q = \pi\cdot(4\cdot n+1)/16$; можно выражение (11) представить в следующем виде:

$$X'(2\cdot k+1)=\sum_{n=0}^{3}\left[x''(n)\cdot\cos\left(\frac{(4\cdot n+1)\cdot\pi}{8}\cdot k\right)\right]-$$
$$-\sum_{n=0}^{3}\left[\left[x'(n)-x'(n+4)\right]\cdot\cos\left(\frac{(4\cdot n+1)\cdot\pi}{16}\cdot(2\cdot k-1)\right)\right] \qquad (12)$$

где $x''(n)=2\cdot\left(x'(n)-x'(n+4)\right)\cdot\cos\left(\frac{(4\cdot n+1)\cdot\pi}{16}\right)$

Первый член выражения (12) представляет собой четырехточечное ДКП, последний член представляет собой не что иное как X'(2k-1).

Таким образом коэффициент X'(2k+1) может быть рекурсивно вычислен через ДКП с вдвое меньшим числом точек и вычисленный на предыдущем шаге коэффициент X'(2k-1).

Диаграмма вычисления одномерного ДКП на восемь точек в соответствии с алгоритмом Хоу приведена на рис. 5.

Из рисунка видно, что для реализации ДКП по алгоритму Хоу требуется 12 умножений и 29 сложений. Тогда для реализации двумерного ПДКП в соответствии со стандартом JPEG по методу строк и столбцов с использованием алгоритма Хоу потребуется всего 192 умножений и 464 сложений.

К настоящему моменту разработано достаточно много алгоритмов быстрого ДКП, предназначенных главным образом для уменьшения количества наиболее трудоемких операций умножения при вычислении ПДКП и ОДКП. Эти алгоритмы дают существенное ускорение программной реализации ДКП. Хуже проработаны алгоритмы, ориентированные на распараллеливание вычислительного процесса. В этом направлении требуются дальнейшие исследования, связанные с решением сложных математических задач.

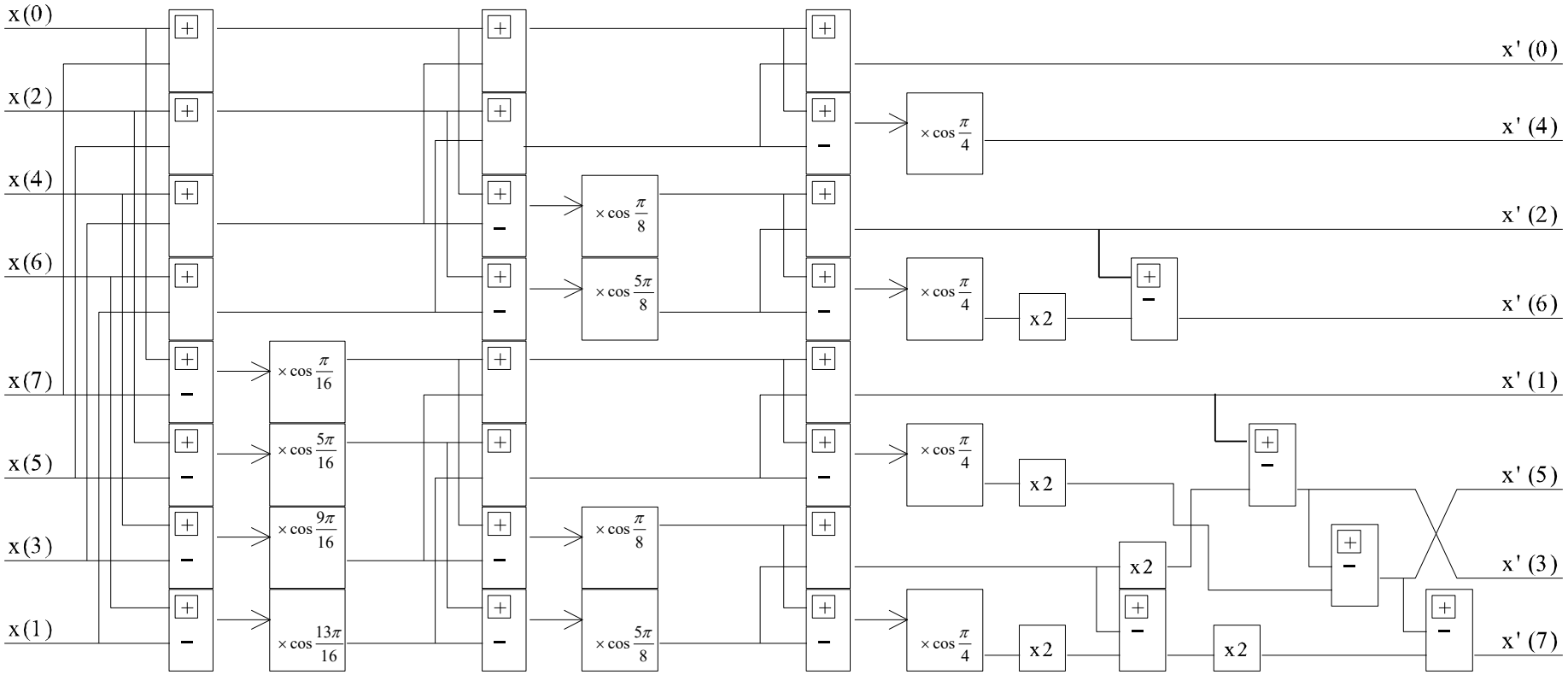

Рис. 5. Диаграмма вычисления одномерного ДКП по алгоритму Хоу

4.3. Распределенная арифметика для реализации ДКП

Под распределенной арифметикой здесь понимается обработка входных переменных ДКП по битам (как правило, начиная с младших разрядов). Распределенная арифметика позволяет распараллелить вычислительный процесс и при этом использовать несложные побитовые умножители на постоянные коэффициенты и сумматоры. Распределенная арифметика позволяет реализовать ДКП на массивах специализированных относительно простых процессорных элементов (ПЭ).

Рассмотрим применение распределенной арифметики для построения 1 - D ДК - преобразователя, реализующего вычисления по формуле (9).

Переменные x'(n) представляются в восьмиразрядном двоичном дополнительном коде.

$$x'(n) = -x_0'(n) + \sum_{j=1}^{7} \left[x_j'(n) \cdot 2 \right] \qquad (13)$$

Тогда X'(k) в соответствии с формулой (9) можно представить в следующем виде:

$$X'(k) = -f\left[x_0'(0),...,x_0'(7)\right] + \sum_{j=1}^{7} \left[f\left[x_j'(0),...,x_j'(7)\right] \cdot 2 \right] \qquad (14)$$

где $$f\left[x_j'(0),...,x_j'(7)\right] = \sum_{n=0}^{7} \left[a(k,n) \cdot x_j'(n) \right] \qquad (15)$$

a(k,n) - постоянный коэффициент, представленный в формуле (9) соответствующим косинусом. Функции (15) представляют собой частные суммы постоянных коэффициентов, зависящие от соответствующих бит входных данных и индекса k. Частные суммы (15), зависящие от восьми битовых переменных, можно предварительно вычислить и записать в восьмивходовое ПЗУ.

Таким образом в соответствии с уравнением (14) X'(k) можно вычислить выполнив последовательность сдвигов и сложений частных сумм f [x'(0),...,x'(7)]. Вычисление X'(k) по формуле (14) выполняет процессорный элемент, изображенный вместе с регистрами преобразования входных данных из параллельного вида в последовательный, на рисунке 6.

ПЭ содержит ПЗУ на 256 слов и суммирующий со сдвигом накопитель. ПЭ работает следующим образом.

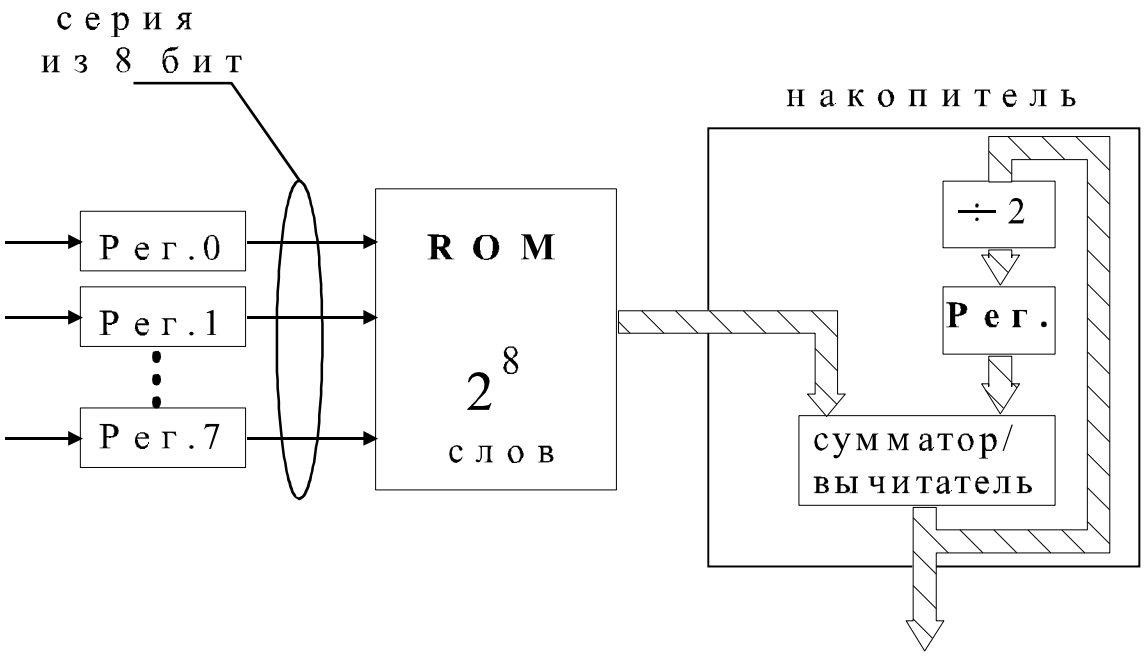

Рис. 6. Реализация распределения арифметики для вычисления ДКП

Входные данные (восемь очередных отсчетов) параллельно загружаются в сдвиговые регистры R0-R7. Младшие разряды регистров R0-R7 поступают на адреса выборки слова из ПЗУ. В каждом из восьми (для первого этапа) тактов вычисления из ПЗУ выбирается сумма произведений коэффициентов a(k,n) на соответствующие младшие разряды сдвиговых регистров R0-R7. С выхода ПЗУ информация поступает на последовательный накопитель результата, где складывается на сумматоре с накопленной ранее суммой частичных произведений, предварительно сдвинутой на один разряд вправо. В каждом такте младший разряд сдвигаемой влево суммы выгружается из накопителя. После прогона всех бит входных данных через ПЭ регистр последнего содержит старшие разряды X'(k).

На основе рассмотренного ПЭ можно построить 1-D ДК - преобразователь. Преобразователь должен содержать содержит блок регистров для преобразования входных данных из параллельного вида в последовательный, восемь ПЭ, блок регистров для преобразования последовательных выходных данных в параллельные.

На основе использования формул (10) и (11) можно построить преобразователь с ПЗУ емкостью на 16 слов. Однако при этом в преобразователь необходимо ввести четыре последовательных сумматора и четыре последовательных вычитателя.

5. Программа Coding JPEG

Программа Coding JPEG создана специально для изучения стандарта JPEG.

В программе реализованы все 7 шагов обработки изображения, рассмотренные в разделе 3. Влиять на результат работы программы можно только на первых 4 шагах, 5 – 7 шаги выполняются автоматически при сохранении конечного файла.

Входные данные:

- Файл структуры BMP с ограничениями, которые будут указанны далее;
- Файл после ДКП с расширением dct.

Выходные данные:

- Файл, подлежащий ДКП с расширением dct;
- Файл после преобразования по стандарту JPEG с расширением npg.

Программа предусматривает выполнение следующих 6 операций, необходимых для выполнения лабораторной работы:

1) Открытие изображения
2) Определение рабочей области
3) Выбор минимального блока данных
4) Кодирование изображения
5) Запись изображения
6) Анализ изображения

1) Открытие изображения

Чтобы открыть изображение используйте команду меню: **Файл\ Открыть**.

Вы можете выбрать файл DIB, или BMP без RLE компрессии. Причем размер изображения не должен превышать 0.8 части экрана (т.е. при разрешении экрана 800x600 вы можете открыть файл размером не более чем 640x480).

2) Определение рабочей области

Рабочая область – выбранная область изображения которая подлежит дальнейшему кодированию.

Для выделения рабочей области поместите курсор на окно с открытым файлом BMP и нажмите левую кнопку мыши, не отпуская кнопки, двигайте

мышь в любую сторону в пределах изображения. Выделив нужную область, отпустите кнопку мыши.

Для выделения всего изображения, нажмите правую кнопку мыши в любом месте окна файла BMP.

3) Выбор минимального блока данных

Минимальный блок данных выбирается в рабочей области изображения с помощь команды меню: **Кодирование\ Сетка MDU**. После чего, поместив курсор в рабочую область необходимо выбрать нужный минимальный блок и отметить его, нажав левую кнопку мыши. Окно выбора блока сообщит вам о номере выбранного блока.

4) Кодирование изображения

Чтобы начать процесс кодирование используйте команду меню: **Кодирование\ Кодирование….**

Окно кодирования содержит две рабочих области:

1) Настройка качества.
2) Пошаговое кодирование.

Настройка качества.

Область настройки качества предназначена для влияния на качество изображения с помощью матрицы квантования.

Вы можете изменить матрицу квантования с помощью коэффициента квантования перемещая движок или изменяя значение в диалоговом окне коэффициента квантования.

Коэффициент квантования – число, на которое делятся все элементы матрицы квантования.

Для более точной настройки качества изображения пользуйтесь кнопкой Настройка матрицы квантования.

В окне параметров матрицы квантования находятся 64 элемента. Изменение любого элемента ведет к изменению качества изображения в той или иной степени в зависимости от важности данного элемента.

Пошаговое кодирование.

Для пошагового кодирования используйте последовательно следующие кнопки:

8x8

Операция разбиение вашего изображения на минимальные блоки данных размером 8x8 пикселей.

-128

Осуществление сдвига уровня путем вычитания 128 из каждого элемента изображения.

W

Сохранение полученных данных в файл с расширением dct для дальнейшего самостоятельного дискретно косинусного кодирования.

R

Открытие DCT файла для дальнейшего преобразования.

Квантование

Операция квантования данных.

Просмотр результата кодирования

Используйте эту кнопку для просмотра результатов после каждой операции кодирования.

С помощью кнопки **Вывести** производится декодирование данных и вывод изображения на экран в стандарте JPEG.

5) Запись изображения

Запишите полученное после кодирования изображение на диск с помощью команды меню: **Файл\ Сохранить**.

6) Анализ изображения

Иногда недостаточно визуальной оценки качества изображения, программа предлагает следующие способы оценки качества сжатия:

Интегральный анализ

В окне интегрального анализа расположена информации о файлах BMP и JPEG.

Оценка качества производится с помощью меры биения (2) и меры отношения сигнала к шуму (3).

Последняя информация окна интегрального анализа отражает отношение между размером BMP и JPEG файлов в процентном выражении.

Разностный анализ

Для определения потерь данных изображения во время кодирования используется разностный анализ, предусматривающий получение окна разности между BMP и JPEG файлами.

6. Формат файла DCT

Для реализации DCT кодирования студентам необходимо написать собственную программу, реализующую любой из предложенных алгоритмов DCT кодирования, которая будет взаимодействовать с программой Coding JPEG следующим образом:

1) подготовка данных:

После загрузки файла структуры BMP выполните следующие шаги кодирования описанные в разделе 5 пункте 4:

- Операция разбиения 8x8,
- Сдвиг уровня,
- Сохранение полученных данных в файл с расширением dct.

2) DCT кодирование:

Загрузите написанную вами программу реализации DCT кодирования, которая должна выполнить следующие действия:

- Открыть сохраненный вами файл с расширением dct (структура данных файла приведена ниже),
- Выполнить DCT кодирование,
- Изменить статус файла, поменяв значение состояния файла с FALSE на TRUE,
- Сохранить результат под тем же именем с тем же расширением (dct).

3) загрузка файла после DCT кодирования:

В программе Coding JPEG выполните команду «Открытие DCT файла для дальнейшего преобразования», описанную в разделе 5 пункте 4.

Структура данных расположенных в файле DCT представляется следующим образом (рис. 7).

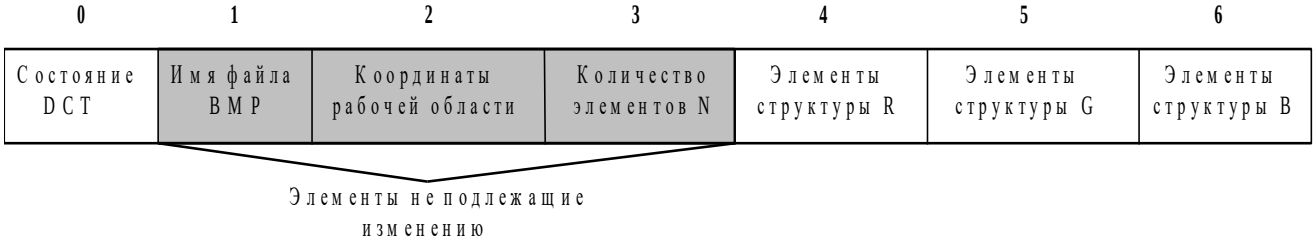

Рис. 7. Структура файла DCT

Состояние DCT – статус DCT файла [FALSE – DCT кодирование не применялось, TRUE – файл после DCT кодирования].

Количество элементов N – количество данных в каждой структуре. (Например, N=10, т.е. количество элементов структуры R = 10, структуры G = 10 и структуры B = 10).

Элементы структуры RGB – собственно данные подлежащие DCT кодированию.

Тип, размер и значение данных представлены в таблице:

Номер	**Тип и размер**	**Входное значение**	**Выходное значение**
0	Bool	FALSE	TRUE
1	Char [_MAX_FNAME + + _MAX_EXT]	Имя файла	Не изменять
2	Int [4]	Координаты	Не изменять
3	Int	Количество элементов N	Не изменять
4	Float·N	Элементы структуры R	DCT R
5	Float·N	Элементы структуры G	DCT G
6	Float·N	Элементы структуры B	DCT B

После выполнения DCT кодирования необходимо сохранить все данные в той последовательности, в которой они были первоначально.

Для примера приведем несколько функций программы выполняющей DCT кодирование:

//Программа выполнена на MS Visual C++ 6.0 for Windows.

//Файл DCT_File.cpp выполняющий чтение и запись данных.

```
char    szSaveName [_MAX_FNAME + _MAX_EXT]; //Имя файла BMP
bool    bDCT; //Состояние DCT
int     iCoordinates[4]; //Координаты рабочей области


extern CHAR     szFileName[_MAX_PATH]; //путь к файлу DCT
extern CHAR     szFileTitle[_MAX_FNAME + _MAX_EXT]; //имя файла DCT
extern float    *bufRed, *bufGreen, *bufBlue; // элементы структуры RGB
extern int      enddata; //Количество элементов

//********** LoadFile *************
//Функция для загрузки содержимого файла или очистки поля
BOOL LoadFile ()
{
 DWORD    dwFileSize;
 HANDLE   hFile;

 bufRed = (float*)  calloc (4800*64, sizeof (float));
 bufGreen = (float*)  calloc (4800*64, sizeof (float));
 bufBlue = (float*)  calloc (4800*64, sizeof (float));

 // Открываем файл по имени
 hFile = CreateFile (szFileName, GENERIC_READ, 0, NULL,
  OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

 // Определяем длинну файла
 dwFileSize = GetFileSize (hFile, NULL);

 // Читаем содержимое файла
 ReadFile (hFile, &bDCT, sizeof(bool), &dwFileSize, NULL);
 ReadFile (hFile, &szSaveName, sizeof(szSaveName), &dwFileSize, NULL);
 ReadFile (hFile, &iCoordinates, sizeof(iCoordinates), &dwFileSize, NULL);
 ReadFile (hFile, &enddata, sizeof(int), &dwFileSize, NULL);
 ReadFile (hFile, bufRed, sizeof(bufRed)*enddata, &dwFileSize, NULL);
 ReadFile (hFile, bufGreen, sizeof(bufGreen)*enddata, &dwFileSize, NULL);
 ReadFile (hFile, bufBlue, sizeof(bufBlue)*enddata, &dwFileSize, NULL);
```

```
 // Закрываем файл
 CloseHandle(hFile);
return FALSE;
}

//********** SaveFile **********
//Функция для сохранения данных в файл
BOOL SaveFile ()
{
 DWORD    dwFileSize;
 HANDLE   hFile;

 // По имени открываем или создаем файл, если он не существует
 hFile = CreateFile(szFileName, GENERIC_WRITE,
         0, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

 // Удаляем содержимое файла
 SetEndOfFile(hFile);

 bDCT = TRUE; //DCT кодирование завершено

 // Запись в файл
 WriteFile (hFile, &bDCT, sizeof(bool), &dwFileSize, NULL);
 WriteFile (hFile, &szSaveName, sizeof(szSaveName), &dwFileSize, NULL);
 WriteFile (hFile, &iCoordinates, sizeof(iCoordinates), &dwFileSize, NULL);
 WriteFile (hFile, &enddata, sizeof(int), &dwFileSize, NULL);
 WriteFile (hFile, bufRed, sizeof(bufRed)*enddata, &dwFileSize, NULL);
 WriteFile (hFile, bufGreen, sizeof(bufGreen)*enddata, &dwFileSize, NULL);
 WriteFile (hFile, bufBlue, sizeof(bufBlue)*enddata, &dwFileSize, NULL);

 // Закрываем файл
 CloseHandle(hFile);

 free (bufRed);
 free (bufGreen);
 free (bufBlue);

 return TRUE;
}

//Файл DCT_Coder.cpp реализующий DCT кодирование.

 struct rgb {float matrix[8][8];} Red, Green, Blue;
```

```
 CHAR        szFileName[_MAX_PATH]; //задать самостоятельно
 CHAR        szFileTitle[_MAX_PATH]; // задать самостоятельно
 float    *bufRed, *bufGreen, *bufBlue;
 int      enddata;
 int      v, u;

struct rgb FDCT (struct rgb);
BOOL LoadFile ();
BOOL SaveFile ();

//*********** CDCTCoder ***********
//Функция для подготовки данных к DCT кодированию
void CDCTCoder()
{
 int     m, i;

 LoadFile();
 m=0;
 i=0;
 do {
          for (v=0; v<8; v++)
                for (u=0; u<8; u++)
                {
                 Red.matrix[v][u] = *(bufRed+i) ;
                 Green.matrix[v][u] = *(bufGreen+i);
                 Blue.matrix[v][u] = *(bufBlue+i);
                 ++i;
                }

         Red = FDCT (Red);
         Green = FDCT (Green);
         Blue = FDCT (Blue);

         for (v=0; v<8; v++)
                for (u=0; u<8; u++)
                {
                 *(bufRed+m) = Red.matrix[v][u];
                 *(bufGreen+m) = Green.matrix[v][u];
                 *(bufBlue+m) = Blue.matrix[v][u];
                 ++m;
                }

 }while (enddata != m);
```

```
 SaveFile();
}

//************ FDCT ************
//Функция DCT кодирования
struct rgb FDCT (struct rgb code )
{
int     i, x, y, j;
float  sum;
float  fdctmas[8][8], cucv[8][8], kcos[8][8];
double doPI=0.1963495408494;

cucv[0][0]=0.125;

for (j=1; j<8; j++)
 for (i=1; i<8; i++)
   cucv[i][j]=0.25;

 for (i=1; i<8; i++)
  {
   cucv[i][0]=0.1785714285714;
   cucv[0][i]=cucv[i][0];
  }

for (i=0;  i<8; i++)
  for (u=0; u<8; u++)
     kcos[i][u]=cos((2*i+1)*u*doPI);

          for (u=0; u<8; u++)
             for (v=0; v<8; v++)
                  {
                   sum=0;
                   for (i=0; i<8; i++)
                      for (j=0; j<8; j++)
                         sum=sum+code.matrix[i][j]*(kcos[i][u])*(kcos[j][v]);
              fdctmas[u][v]=cucv[u][v]*sum;
       }

         for (x=0; x<8; x++)
       for (y=0; y<8; y++)
         code.matrix[x][y] = fdctmas[x][y];
return code;
}
```

7. Указания к выполнению предварительного домашнего задания

Составить программу выполнения прямого двумерного дискретно-косинусного преобразования. Входные данные программы хранятся в файле DCT. Вычисленные коэффициенты DCT также необходимо сохранить в файл DCT. Использовать формат файла DCT, описанный в предыдущем разделе. При составлении программы использовать описание программы выполнения DCT, приведенное в разделе 6. Программа должна реализовать один из алгоритмов, приведенных ниже.

Варианты:

1. Разложение двумерного преобразования на последовательность одномерных 8-точечных преобразований;
2. Разложение + алгоритм Ли;
3. Разложение + алгоритм Хоу;
4. Разложение + распределенная арифметика.

Описание алгоритмов приведено в разделе 4.

8. Порядок выполнения работы

Реализация ДКП по заданному алгоритму.

8.1. Необходимо отладить написанную программу реализации ДКП.
8.2. Запустить программу Coding JPEG. В пошаговом режиме ввести значения отсчетов эталонного бока MDU, указанные преподавателем.
8.3. Сохранить блок MDU на жесткий диск, в виде файла с расширением DCT.
8.4. Запустить написанную программу на выполнение ДКП для записанного файла.
8.5. Загрузить в программу Coding JPEG получившийся файл с расширением DCT.
8.6. Записать получившиеся для заданного MDU коэффициенты ДКП.
8.7. Выполнить пункты 8.2 – 8.6 с использованием эталонной программы.
8.8. При совпадении результатов предъявить их преподавателю.

Исследование основных этапов алгоритма сжатия и восстановления минимального блока данных.

8.9. Запустить программу Coding JPEG. Загрузить указанный преподавателем файл изображения.
8.10. В пошаговом режиме записать значения отсчетов для указанного преподавателем MDU.
8.11. Записать смещенные значения блока отсчетов.

8.12. Сохранить блоки MDU на жесткий диск, в виде файла с расширением DCT.
8.13. Запустить написанную программу на выполнение ДКП для записанного файла.
8.14. Загрузить в программу Coding JPEG получившийся файл с расширением DCT.
8.15. Записать получившиеся для заданного MDU коэффициенты ДКП.
8.16. Записать получившиеся квантованные коэффициенты ДКП.
8.17. Сохранить файл со сжатым изображением.
8.18. Записать значения отсчетов соответствующего MDU восстановленного файла.
8.19. Результаты предъявить преподавателю.

Исследование влияния матрицы квантования на качество сжатия изображения.

8.20. Запустить программу Coding JPEG. Загрузить указанный преподавателем файл изображения.
8.21. В пошаговом режиме записать значения отсчетов для указанного преподавателем MDU.
8.22. Подбирая матрицу квантования добиться сжатия изображения с качеством, заданным преподавателем.
8.23. Записать характеристики сжатия изображения.
8.24. Результаты предъявить преподавателю.
8.25. Записать значения отсчетов соответствующего MDU восстановленного файла.
8.26. Вычислить значения критериев качества сжатия для заданного MDU.

9. Содержание отчета

Отчет должен содержать :
- блок-схему программы, подготовленной в рамках домашнего задания;
- текст программы в форме таблицы;
- результаты выполнения пунктов 8.6, 8.10, 8.11, 8.15, 8.16, 8.18, 8.21, 8.23, 8.25, 8.16.

10. Вопросы для самопроверки

1. Назовите основные требования приложений к алгоритмам компрессии.
2. Почему высокая скорость компрессии, высокое качество изображений и высокая степень компрессии взаимно противоречивы? Покажите противоречивость каждой пары условий.
3. Назовите основные характеристики алгоритмов сжатия изображений.
4. Расскажите о критериях оценки качества сжатия изображений.
5. Дайте общую характеристику алгоритму JPEG.
6. Расскажите об этапах сжатия по алгоритму JPEG.
7. Расскажите о дискретно-косинусном преобразовании.
8. Каким образом осуществляется переход от двумерного ДКП к одномерному?
9. Как работает алгоритм Хоу?
10. Расскажите об использовании распределенной арифметики для реализации ДКП.

Библиографический список

1. JPEG Chair (Gregory Wallace, Digital Equipment Corporation), William B.Pennebaker (IBM), Joan L.Mitchell (IBM). JPEG Technical Specification, Revision 5. // JPEG-8-R5 January 2, 1990.

2. Ватолин Д.С. Алгоритмы сжатия изображений. Методическое пособие. Издательский отдел факультета Вычислительной математики и Кибернетики МГУ им М.В.Ломоносова, 1999 г. – 76 с.

3. Lee B.G. A new algorithm to compute the discrete cosine transform//IEEE Trans. Acoust.,Speech, and Signal Processing. 1984. -32, Dec. -P.1243-1245.

4. Hou H.S. A fast recursive algorithm for computing the discrete cosine transform//IEEE Trans. Acoust., Speech, and Signal Process. 1987. -35, Oct.- P.1445-1461.

5. Narasimha M.J.,Peterson A.M. On the computation of the discrete cosine transform// IEEE Trans. On Communications. 1978. -26, June - P.934-936.

6. Markoul J. A cosine transform in one and two dimensions//IEEE Trans. Acoust.,Speech, and Signal Processing. 1980. -28, Feb. -P.27-34.