Министерство образования и науки Российской федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Нижегородский научно-исследовательский государственный университет

им. Н.И. Лобачевского»

Физический факультет

Кафедра информационных технологий в физических исследованиях

Сжатие – восстановление изображения (архивация) методом, аналогичным JPEG.

Отчет по учебной практике

Выполнил

студент группы 0522М1ИС

Боровков Сергей

Проверил

профессор каф. ИТФИ, д.ф.-м.н.

Морозов О.А.

Нижний Новгород

2022

# Цели работы

1. Провести поиск и анализ литературных источников по тебе сжатия – восстановления методом JPEG.
2. Разработать компьютерную программу, реализующая сжатие – восстановления изображения методом, аналогичным JPEG.
3. Провести анализ полученных результатов.

# Введение

В последнее время изображения и иллюстрации стали использоваться повсеместно. Проблема, связанная с их большим размером, появилась при работе и на рабочих станциях, и на персональных компьютерах. Так, одно полноэкранное (640х480) полноцветное (24 бита на точку) изображение занимает почти мегабайт. Учитывая то, что обычно используется несколько иллюстраций, и то, что они часто бывают гораздо большего размера, держать их в неупакованном виде становится накладно. В последние годы решению этой проблемы уделяется достаточно серьезное внимание. Разработано большое количество различных алгоритмов архивации графики: использовались 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 работе подробно рассмотрен алгоритм сжатия JPEG, который на сегодняшний момент является одним из самых популярных и достаточно мощных алгоритмов и не имеет особенных конкурентов в области сжатия изображений фотографического качества.

Началось все в 1986 году, когда под эгидой международной организации по стандартизации (ISO) и международной электротехнической комиссии (IEC) была создана рабочая группа, в которую вошли ведущие специалисты в области фотографии со всего мира. Именно тогда мир услышал слово JPEG, которое на самом деле является аббревиатурой от Joint Photographic Experts Group (Объединенная группа экспертов в области фотографии).

Основной целью Joint Photographic Experts Group было создание метода сжатия изображений фотографического качества, до размеров, приемлемых для передачи по компьютерным сетям. И уже в августе 1990 года ISO стандартизовал метод сжатия графических изображений с потерями информации, созданный Joint Photographic Experts Group. Он основан на удалении из изображения той части информации, которая слабо воспринимается человеческим глазом. Лишенное избыточной информации изображение занимает гораздо меньше места, чем исходное. Степень сжатия, а, следовательно, и количество удаляемой информации, плавно регулируется. Тем самым достигается компромисс между размером и качеством изображений. Таким образом, этот метод лучше всего подходит для сжатия фотографий и многоцветных рисунков с множеством деталей и плавных цветовых переходов. Но нужно помнить, что JPEG пригоден для конечного представления графики, а не для хранения промежуточных данных, так как после каждого повторного сохранения некоторая часть графической информации безвозвратно теряется.

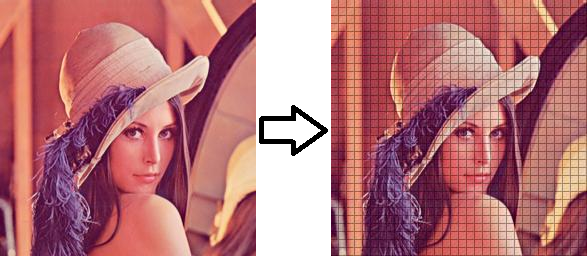
По сравнению с любыми другими общепринятыми форматами изображений JPEG обеспечивает максимальное сжатие фотографических изображений. Фотографию, для хранения которой в виде файла Windows BMP требуется 1 Мбайт, в формате JPEG обычно можно сжать до 50 Кбайт, при этом достигается сжатие более чем в 20 раз за счёт незначительных потерь качества.

Целью данной работы является исследование существующего алгоритма сжатия JPEG, а создание собственной компьютерной программы, осуществляющее сжатие-восстановление изображения методом, аналогичным JPEG.

# Принципы сжатия JPEG.

1. Сегментация изображения

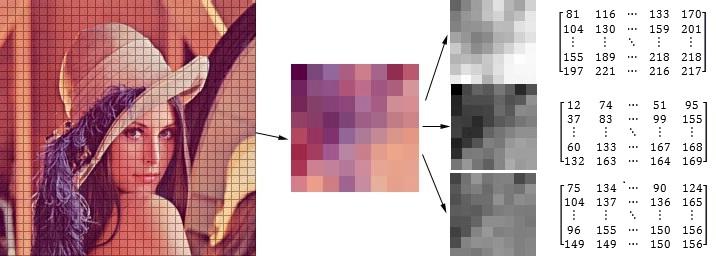
На первом этапе алгоритма JPEG изображение разделяется на маленькие блоки 8х8 (рис. 1), которые обрабатываются индивидуально в течение всего процесса сжатия.



*Рис. 1. Сегментация изображения*

2. Преобразование цветового пространства RGB → YCbCr

Так называемое «цветовое пространство» относится к математической модели, которая выражает цвета. Например, общая модель RGB разлагает цвета на три компонента: красный, зеленый и синий, так что изображение может быть разложено на три изображения в градациях серого. В математическом выражении каждый сегмент 8х8 может быть выражен в виде трех матриц 8х8, диапазон значений которых обычно находится между [0, 255] (рис. 2).



*Рис. 2. Цветовое представления RGB.*

В принципе алгоритм JPEG способен кодировать изображения, основанные на любом типе цветового пространства. JPEG кодирует каждый компонент цветовой модели отдельно, что обеспечивает его полную независимость от любой модели цветового пространства.

В случае применения цветового пространства яркость/цветность, например такого, как YUV или YCbCr, достигается лучшая степень сжатия. Компонента Y представляет собой интенсивность, а U и V - цветность. Эта модель может быть переведена в RGB посредством преобразования без какой-либо коррекции насыщенности. Для полутоновых изображений (в градациях серого) используется только одна составляющая Y.

Преобразование цветовой модели RGB в модель YCbCr осуществляется с помощью следующих соотношений:

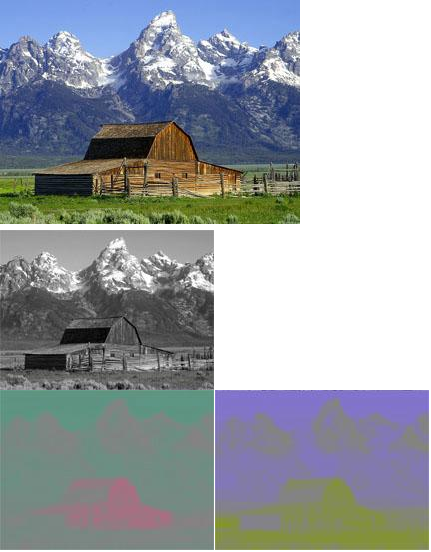
(1)

(2)

(3)

На экране компьютера мы практически никогда не видим реально полноцветных изображений реального мира. Это объясняется ограниченными возможностями по цифровому представлению в памяти ПЭВМ, искажениями при воспроизведении цвета монитором и видеокартой. В результате на мониторе ПЭВМ воспроизводятся, в зависимости от выбранного видеорежима, цвета наиболее близкие к реальным.

Большая часть визуальной информации, к которой наиболее чувствительны глаза человека, состоит из высокочастотных, полутоновых компонентов яркости (Y) цветового пространства YCbCr. Две других составляющих цветности (Сb и Сr) содержат высокочастотную цветовую информацию, к которой глаз человека менее чувствителен. Следовательно, определенная её часть может быть отброшена и, тем самым, можно уменьшить количество учитываемых пикселей для каналов цветности.



*Рис. 3. Цветовое представление YCbCr*

На рис.3 хорошо видно, что на карте яркости больше деталей. После того, как JPEG преобразует изображение в YCbCr, может выполняться различная обработка в зависимости от важности данных. Вот почему JPEG использует это цветовое пространство.

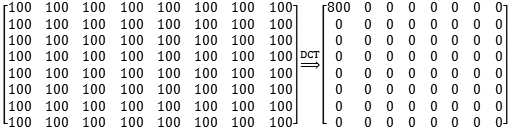
3. Дискретно-косинусное преобразование

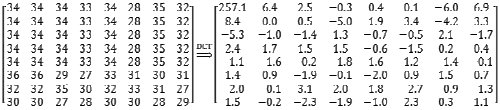
Ключевым компонентом работы алгоритма является дискретное косинусное преобразование. Дискретное косинусное преобразование (DCT) представляет собой разновидность преобразования Фурье и, также как и оно, имеет обратное преобразование. Графическое изображение можно рассматривать как совокупность пространственных волн, причем оси X и Y совпадают с шириной и высотой картинки, а по оси Z откладывается значение цвета соответствующего пикселя изображения. Дискретное косинусное преобразование позволяет переходить от пространственного представления картинки к её спектральному представлению и обратно. Воздействуя на спектральное представление картинки, состоящее из «гармоник», т.е., отбрасывая наименее значимые из них, можно балансировать между качеством воспроизведения и степенью сжатия.

Формула дискретного косинусного преобразования (4):

Формула обратного дискретного косинусного преобразования (5):

В получившейся матрице коэффициентов низкочастотные компоненты расположены ближе к левому верхнему углу, а высокочастотные - справа и внизу. Это важно потому, что большинство графических образов на экране компьютера состоит из низкочастотной информации. Высокочастотные компоненты не так важны для передачи изображения. Таким образом, дискретное косинусное преобразование позволяет определить, какую часть информации можно безболезненно выбросить, не внося серьезных искажений в картинку.





*Рис. 4. Примеры DCT-преобразования*

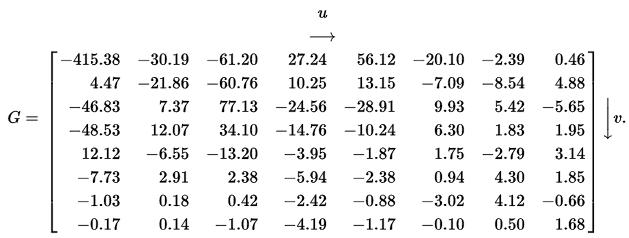
4. Квантование

После расчёта DCT следующий шаг включает поиск и отбрасывание коэффициентов, вклад которых в формирование изображения минимален.

Для решения этой задачи стандарт JPEG определяет простой механизм, именуемый квантованием. Чтобы выполнить квантование коэффициентов DCT, необходимо разделить их на конкретное значение (коэффициент квантования) и округлить результат до ближайшего целого числа.

Чем больше коэффициент квантования, тем больше данных теряется, поскольку реальное DCT-значение представляется все менее и менее точно. Каждая из 64 позиций выходного блока DCT имеет собственный коэффициент квантования. Причем термы большего порядка квантуются с большим коэффициентом, чем термы меньшего порядка. Кроме того, для данных яркости и цветности применяются отдельные таблицы квантования, позволяющие квантовать данные цветности с большими коэффициентами, чем данные яркости. Таким образом, JPEG использует различную чувствительность глаза к яркости и цветности изображения.

Пример матрицы данных яркости, её данные следующие (рис. 5):

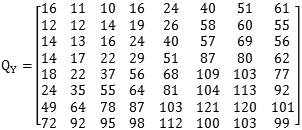
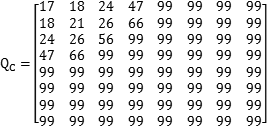


*Рис. 5. Матрица данных яркости*

Алгоритм квантования, предоставляемый JPEG, выглядит следующим образом:

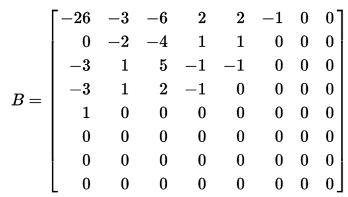
(6)

Среди них G – это матрица изображения, которую нам нужно обработать, а Q – это матрицы квантования. Алгоритм JPEG предоставляет две стандартные матрицы коэффициентов квантования для обработки данных яркости Y и данных цветового различия Cr и Cb (рис. 6).

*Рис. 6 Таблицы квантования стандартной яркости (слева) и цветового различия (справа)*

Окончательный квантованный результат (рис. 7):



*Рис. 7. Результат квантования матрицы данных*

Видно, что большая часть данных стала 0, что очень способствует последующему сжатому хранению.

4. Кодирование Хаффмана

Последним шагом сжатия JPEG является кодирование данных Хаффманом, Хаффман является основой почти всех алгоритмов сжатия. Его основной принцип заключается в корректировке длины кодирования элементов в соответствии с частотой использования элементов в данных для получения более высокой степени сжатия.

Возьмем пример, такой как следующий фрагмент данных

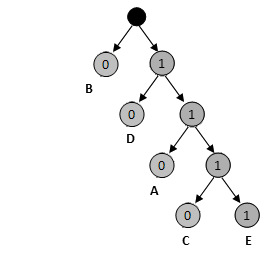
Этот фрагмент данных содержит 33 символа. Каждый символ встречается следующее количество раз.

Если мы используем нашу обычную кодировку фиксированной длины, каждый символ составляет 3 бита.

Таким образом, для сохранения этого текста требуется всего бит, но если мы используем следующую кодировку, основанную на вероятности появления символа

Тогда этот текст требует всего бита для сохранения, а степень сжатия составляет 63% .

Код Хаффмана обычно генерируется с использованием двоичного дерева. Код, полученный таким образом, соответствует правилу префикса, т.е. более короткий код не может быть префиксом более длинного кода. Например, приведенный выше код генерируется двоичным деревом ниже (рис. 8).



*Рис. 8. Пример дерева Хаффмана*

Таким образом, матрицы данных изображения кодируются методом Хаффмана, куда вместе с деревом, вписываются в файл, получая по итогу сжатые данные об изображении.

5. Восстановление изображения

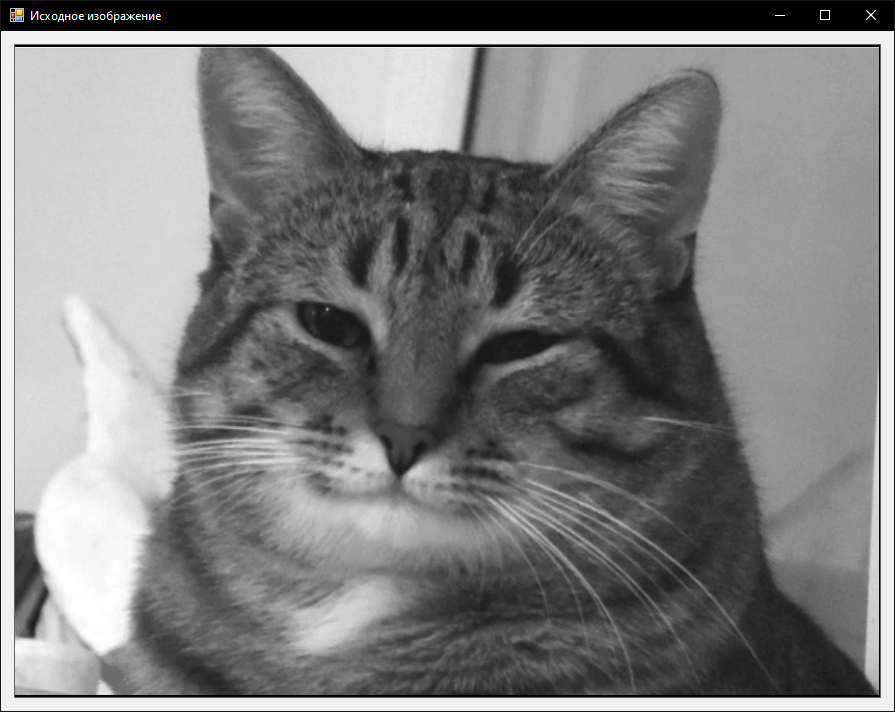
Процесс восстановления изображения в этом алгоритме полностью симметричен.

Сперва, декодируется файл с данными, используя сохраненное дерево Хаффмана, т.е. должны получиться данные, что на рис. 7. Далее применяется обратно матрицы квантования и выполняется обратное DCT. В результате получается восстановленное изображение с потерями.

# Практическая часть

В данной работе была разработана компьютерная программа на языке C#. Был реализован весь порядок действия обработки изображения, описанный выше. Для упрощения задачи, все исходные изображения переводились в полутоновые, т.е. работали только с яркостной матрицей данных, так как она содержит всю основную информацию.

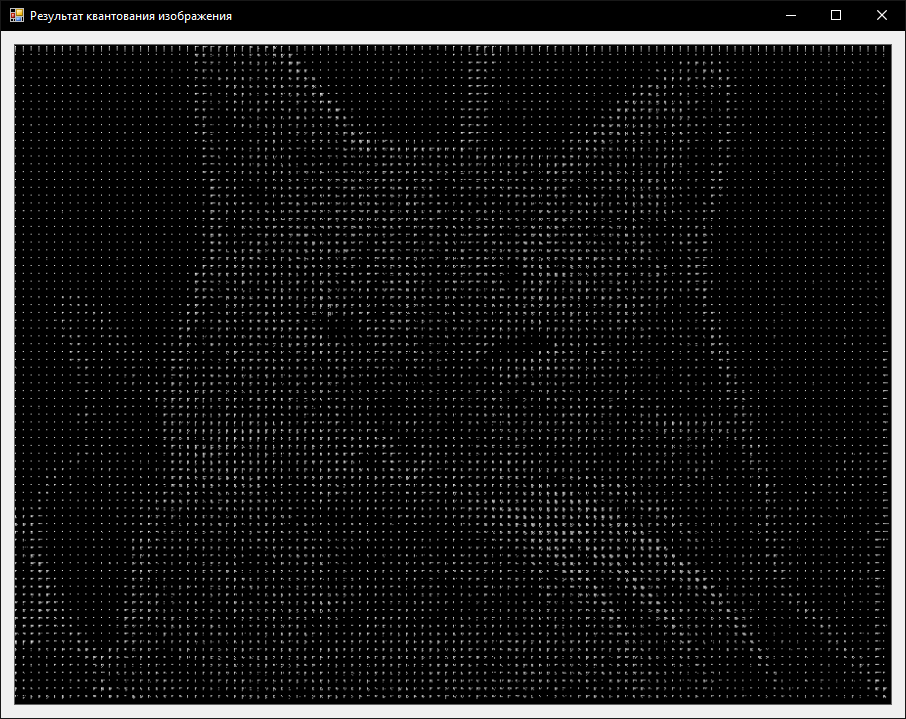
Таким образом, результаты получились следующими. В качестве исходного изображения бралась следующая картинка размером 899х673 (рис. 9):



*Рис. 9. Исходная картинка.*

Для правильности работы алгоритма, картинка дополнительно доводилась до размера, кратного 8.

Далее картинка разбивалась на подматрицы размером 8x8, от каждой подматрицы брались DCT-преобразования и наконец каждая подматрица проходила процедуру «квантования». Результат этих операций показан на рис. 10.



*Рис. 10. Промежуточный этап сжатия изображения.*

Можно заметить, что картинка разбита на сегменты и энергия каждого сегмента сконцентрирована в левом в верхнем углу – это особенность дискретно-косинусного преобразования. Видны также много черных пикселей – это всё нулевые энергии.

Далее, значения каждого пикселя данного изображения записываются в файл.



Размер файла исходной картинки составил 1198 кБ.

Затем файл сжимался алгоритмом Хаффмана.



Вес файла сжался до 249 кБ, т.е. добились почти 5-кратного сжатия.

И наконец, провели процедуру декодирования файла.



В итоге, получили файл того же размера.

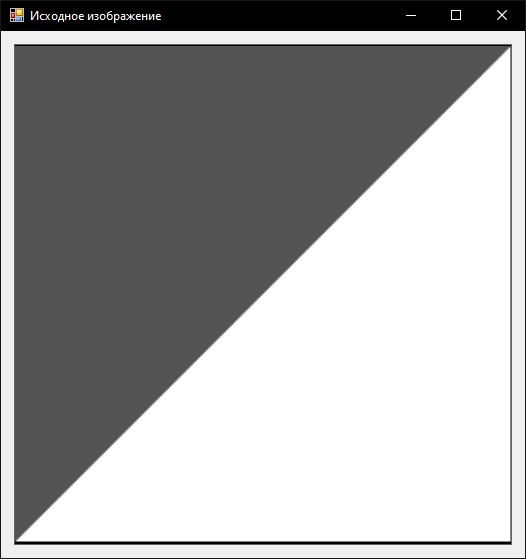
Далее провелась обратная процедура, т.е. обратное «квантование», обратное DCT, и получили восстановленное изображение с потерями (рис. 11).



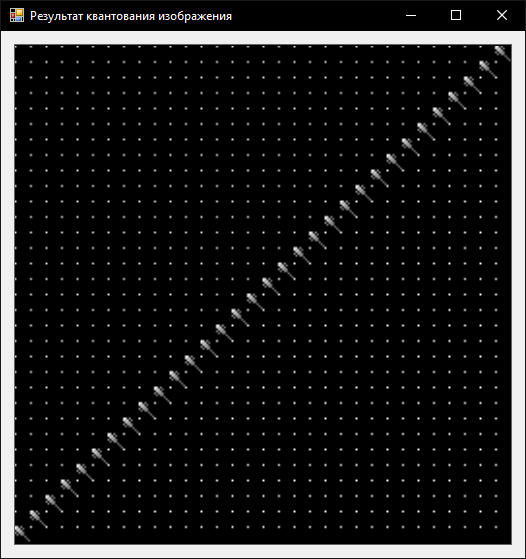
*Рис. 11. Восстановленное изображения.*

Среднеквадратичное отклонения между исходным и восстановленным изображением составило 0.00052, что мало. На картинке видны едва заметные артефакты, возникшие из-за процедуры «квантования», но в целом качество изображения не упало – основные детали сохранились.

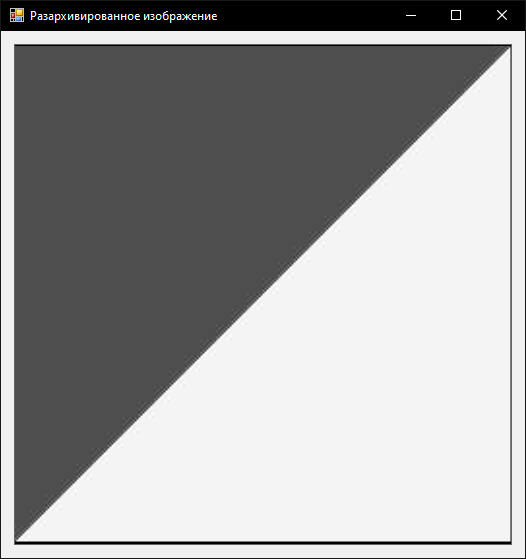
Похожие результаты получились и с более простой картинкой (рис. 12-14)



*Рис. 12. Исходное изображение*

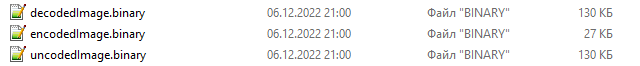


*Рис. 13. Промежуточное состояние изображения*



*Рис. 14. Восстановленное изображение*

Размеры файлов, следующие:



Также удалось добиться почти 5-кратного сжатия. Среднеквадратичное отклонение – 0.00028.

# Вывод

В ходе проделанной работы было сделано:

1. Создана компьютерная программа, реализующая сжатие – восстановления изображения.
2. Удалось добиться 5-кратного сжатия и низкое потерю качества изображения при восстановлении.

# Литература

1. Про JPEG: <https://habr.com/ru/post/482728>
2. Про JPEG: <https://habr.com/ru/post/206264>
3. Про JPEG: <https://russianblogs.com/article/8889932437>
4. Про DCT: <https://studfile.net/preview/3065299/page:25>
5. Алгоритм сжатия Хаффмана: [https://habr.com/ru/company/otus/blog/497566](https://habr.com/ru/company/otus/blog/497566/)
6. Huffman. D. (1952). «A Method for the Construction of Minimum-Redundancy Codes». Proceedings of the IRE. 40 (9): 1098—1101.