МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт промышленной автоматизации и цифровизации
имени А. Буркитбаева

Кафедра «Электроника, телекоммуникации и космические технологии»

Ғалымжан Жансая Нұрмухаммедқызы

Анализ методов сжатия информации при кодировании

ДИПЛОМНАЯ РАБОТА

специальность 5В071900 – Радиотехника, электроника и телекоммуникация

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт промышленной автоматизации и цифровизации
имени А. Буркитбаева

Кафедра «Электроника, телекоммуникации и космические технологии»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ЭТ и КТ
_____И.Сыргабаев
“____ “ _____2020г

ДИПЛОМНАЯ РАБОТА

На тему: Анализ методов сжатия информации при кодировании

по специальности 5В071900 – Радиотехника, электроника и телекоммуникации

Выполнил Ғалымжан Ж.Н.

Рецензент
канд.техн.наук, профессор АУЭС
_____Байкенов А.С.
“__27__ “ ___05____2020г.

Научный руководитель
маг-р техн. наук
_____Байкенова Г.М
“_27__” ____05_____2020г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт промышленной автоматизации и цифровизации
имени А. Буркитбаева

Кафедра Электроника, телекоммуникации и космические технологии

5В071900 – Радиотехника, электроника и телекоммуникации

УТВЕРЖДАЮ
Заведующий кафедрой ЭТ и КТ
_____И.Сыргабаев
“____“ _____2020г

ЗАДАНИЕ
на выполнение дипломной работы

Обучающемуся *Жансая Ғалымжан*
Тема: *Анализ методов сжатия информации при кодировании*
Утвержден *приказом Ректора Университета № 762-б от “27”января 2020г.*
Срок сдачи законченной работы *“5” июня 2020г*.
Исходные данные к дипломной работе: *Теория кодирования. Принципы сжатия информации. Код Шеннона-Фано. Алгоритм кодирования с помощью деревьев Хаффмана. Коэффициент сжатия.*
Краткое содержание дипломной работы:
а) Цель и принцип сжатия информации в сетях связи.
б) Анализ и сравнение методов сжатия информации при кодировании.
в) Расчет степени сжатия для изображений видео, аудиоданных.
Перечень графического материала (с точным указанием обязательных чертежей): *Функциональная схема кодера MPEG. Блок-схема кодера JPEG. Обобщенная структура кодера звукового сигнала с компрессией цифровых аудиоданных.*
Рекомендуемая основная литература:
1. Фрактальное сжатие изображений спутниковой системы X-SAR европейского космического агентства. А.Ф. Богданов и Н.А. Потемкин, Томский государственный университет систем управления и радиоэлектроники, Томск, 2016.
2. Аналитический обзор алгоритмов сжатия цифровой информации. В. В. Кириченко. Объединенный институт проблем информатики НАН Беларуси, Минск, 2016.
3. Эффективное кодирование видеоинформации в новом стандарте H.264/AVC. - Дворкович А. В. Труды НИИР, 2005.

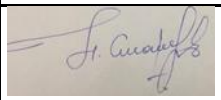
ГРАФИК

подготовки дипломной работы (проекта)

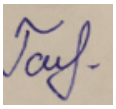
Наименования разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Цель и принцип сжатия информации в сетях связи.	31.01.2020 г.	выполнила
Анализ и сравнение методов сжатия информации при кодировании.	30.03.2020 г.	выполнила
Расчет степени сжатия для изображений видео, аудиоданных.	15.04.2020 г.	выполнила

Подписи

консультантов и нормоконтролера на законченную дипломную работу (проект) с указанием относящихся к ним разделов работы (проекта)

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Нормоконтролер	Доктор PhD Смайлов Н.К.	25.05.2020	

Научный руководитель  Байкенова Г.М.

Задание принял к исполнению обучающийся  Ғалымжан Ж.

Дата "_27_"__01__2020г.

АННОТАЦИЯ

Дипломная работа посвящена анализу методов сжатия данных при передаче по каналам связи с целью уменьшения занимаемого ими объёма. Сжатие, как нам известно, основано на устранении избыточности, содержащейся в исходных данных. Целью данной работы является исследование методов сжатия данных и сравнение их с другими эффективными методами по времени работы и коэффициенту сжатия данных.

Тема дипломной работы очень актуальна, поскольку алгоритмы сжатия используются во многих областях, например, при передаче данных по сетям связи, в программах резервного копирования, в программах-архиваторах и многом другом.

В дипломной работе даются понятия сжатия информации, виды сжатия, область применения, общие правила при кодировании. Рассмотрены методы сжатия данных, а также звука, видео и изображения, так как эти форматы очень актуальны на сегодняшний день. А также приведены расчетные данные, сделан общий анализ всех методов сжатия и определен наиболее оптимальный способ сжатия непосредственно для звука, видео и изображения.

АҢДАТПА

Дипломдық жұмыс байланыс арналары арқылы тарату кезінде алынған көлемді азайту мақсатында деректерді тығыздау әдістерін талдауға арналған. Тығыздау, бізге белгілі болғандай, бастапқы деректерде қамтылған артық шығындарды жоюға негізделген. Бұл жұмыстың мақсаты деректерді тығыздау әдістерін зерттеу және оларды жұмыс уақыты мен деректерді тығыздау коэффициенті бойынша басқа да тиімді әдістермен салыстыру болып табылады.

Дипломдық жұмыста ақпаратты тығыздау ұғымдары, тығыздау түрлері, қолдану саласы, қодалау кезіндегі жалпы ережелер беріледі. Деректерді тығыздау әдістері, сондай-ақ дыбыс, бейне және суретті тығыздау әдістері қарастырылған, өйткені бұл форматтар бүгінгі күні өте өзекті. Сондай-ақ, есептік деректер келтірілген, барлық тығыздау әдістеріне жалпы талдау жасалған және тікелей дыбыс, бейне және сурет үшін тығыздаудың ең оңтайлы тәсілі анықталған.

ANNOTATION

The thesis is devoted to the analysis of data compression methods during transmission over communication channels in order to reduce the volume occupied by them. Compression, as we know, is based on eliminating the redundancy contained in the source data. The aim of this work is to study data compression methods and compare them with other effective methods in terms of operating time and data compression ratio.

The topic of the thesis is very relevant, since compression algorithms are used in many areas, for example, when transmitting data over communication networks, backup programs, archiver programs and much more.

The thesis gives the concepts of information compression, types of compression, scope, general rules for coding. The methods of data compression, as well as sound, video and image are considered, since these formats are very relevant today. Also, the calculated data are presented, a general analysis of all compression methods is made, and the most optimal compression method is determined directly for sound, video and image.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В течение последних десятилетий наблюдается стремительное развитие вычислительной техники. В современном мире человек всегда сталкивается с проблемой как уместить большой поток информации и при этом сэкономить место. Вычислительная техника стремительно совершенствуется, поэтому алгоритмы и методы сжатия данных должны также постоянно развиваться и адаптироваться. Таким образом, задача хранения и передачи текстовой, мультимедийной и другой информации в наиболее компактном виде очень актуальна. Последние десятилетия в этой области ведутся активные разработки.

Ещё в предыдущем столетии было проблематично сократить объём данных. Самым первым методом сжатия информации считается система условных знаков, используемая в военном деле для передачи важных сигналов на дальние расстояния. Чуть позже, когда стали появляться печатные станки, человек стал выделять главное для читателя большим шрифтом, а менее важное – мелким шрифтом, и конечно же, использовать аббревиатуры и сокращения для наиболее встречающихся слов. Спустя некоторое время размер шрифта утратил своё значение, так как стали появляться электронные системы передач. Однако появился новый способ сжатия. Первым алгоритмом сжатия информации считается алгоритм Шеннона-Фано, который основан на принципе замены наиболее часто встречаемых символов на более короткие последовательности бит, а наименее часто встречаемых символов—на наиболее длинные последовательности бит.

Однако алгоритм Шеннона-Фано не являлся оптимальным. При некоторых условиях в нём появлялась избыточность, но уже в 1952 году Дэвидом Хаффманом был разработан алгоритм оптимального кодирования. Этот алгоритм и его принципы широко используется во многих алгоритмах сжатия, которые будут описаны в дипломной работе.

Информация стала основой нашей жизни и её становится всё больше, что требует всё больших ресурсов для её хранения. Объёмы информации, получаемые и отправляемые с помощью сети Интернет, увеличиваются, а в совокупности это означает увеличение затрат, необходимых для хранения и транспортировки информации. С появлением множества различных форматов информации, таких как музыка, изображения и видео появилось множество специализированных алгоритмов сжатия с высокой степенью сжатия для определённых типов файлов.

В рамках дипломной работы был проведён обзор алгоритмов сжатия, проведен сравнительный анализ и определен оптимальный метод сжатия для всех форматов данных.

1 Теория сжатия информации и классические алгоритмы сжатия данных

1.1 Понятия избыточности в теории сжатия

Информация является величиной, не поддающейся количественной оценке. Тем не менее, существует область математики под названием теория информации, где информация изучается с количественной точки зрения. Именно теория информации дала определение понятию избыточности.

Избыточность — превышение количества информации, используемой для передачи или хранения сообщения. Целью сжатия является удаление избыточности, другими словами, удаляются те компоненты, без которых можно обойтись для передачи исходной информации; также сжатие применяется для экономии времени и места при передаче по сетям связи без значительной потери качества.

1.2 Сжатие без потерь

Все методы сжатия можно разделить на две большие группы: сжатие с потерями и сжатие без потерь. Сжатие без потерь применяется в тех случаях, когда информацию нужно восстановить с точностью до бита, пикселя, вокселяи т.д. Такой подход является единственно возможным при сжатии, например, текстовых данных. В некоторых случаях, однако, не требуется точного восстановления информации и допускается использовать алгоритмы, реализующие сжатие с потерями, которое, в отличие от сжатия без потерь, обычно проще реализуется и обеспечивает более высокую степень архивации.

Сжатие без потерь основывается на трех теоретических алгоритмах:

– RLE (Run-Length Encoding) – для графических данных;
– KWE (Keyword Encoding) – для текстовых данных;
– алгоритм Хаффмана – для любых данных.

Теоретическая основа этих алгоритмов была заложена американским ученым Клодом Элвудом Шенноном с его теоремой кодирования для дискретного канала без помех.

1.3 Алгоритм Шеннона - Фано

Метод Шеннона - Фано пользуется кодами переменной длины: часто встречающийся знак кодируется кодом наименьшей длины, изредка

встречающийся — кодом большей длины. Этот алгоритм является одним из первых алгоритмов сжатия, впервые сформулированных американскими учеными Клодом Шенноном и Робертом Фано.

Основные этапы:

1 Символы алфавита записывают по убыванию вероятностей.

2 Далее эти символы делятся на две части, суммарные вероятности символов которых максимально приближены друг к другу.

3 В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».

4 Полученные части делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде. В таблице 1.1 представлен пример кодирование по алгоритму Шеннона-Фано.

Таблица 1.1 - Кодирование по алгоритму Шеннона-Фано

a_i	p(a)	1	2	3	4	Итого
a_1	0.36	0	00			00
a_2	0.18		01			01
a_3	0.18	1	10			10
a_4	0.12		11	110		110
a_5	0.09			111	1110	1110
a_6	0.07				1111	1111

Но в практике сжатия данный алгоритм не используется, поскольку алгоритм не гарантирует сжатия данных наилучшим образом.

1.4 Код Хаффмана

Алгоритм Хаффмана представляет собой алгоритм кодирования буквенных префиксов с минимальной избыточностью. В отличие от алгоритма Шеннона-Фано, алгоритм Хаффмана считается оптимальным, поскольку он также подходит для алфавитов с более чем двумя символами. Классический алгоритм Хаффмана на входе получает таблицу частоты появления символов в сообщении. Потом на основе этой таблицы строится дерево кодирования Хаффмана (рисунок 1.1).

Этапы построения дерева Хаффмана:

1) в строчку записываются все символы в порядке убывания их вероятностей;

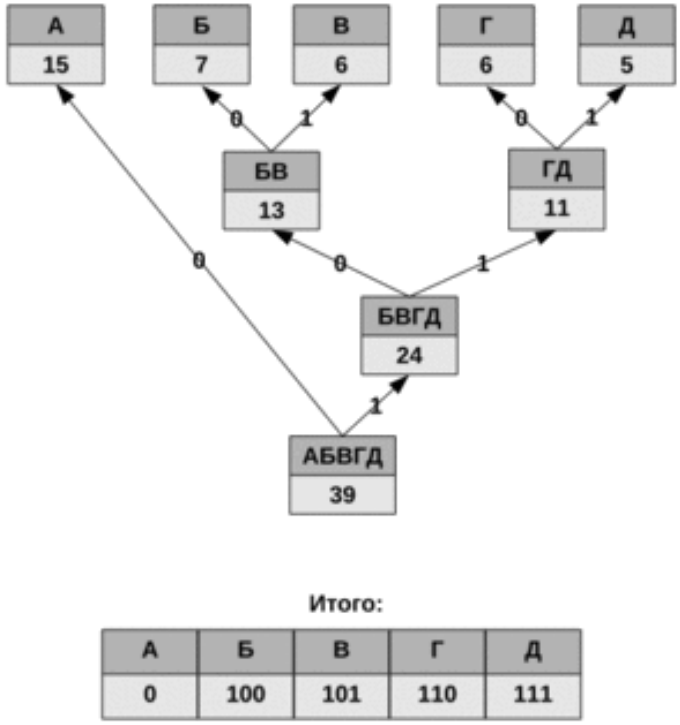
2) объединяются два последних символа с наименьшими вероятностями;

3) создается их родитель, равный сумме вероятностей двух дочерних узлов

4) в очередь добавляется родитель, а два дочерних узла удаляются из списка свободных;

5) процесс повторяется до тех пор, пока не останется только один вакантный узел. Он и будет считаться корнем дерева;

6) далее каждому из узлов полагается код: левому символу приписывается «0», правому «1».



Итого:

А	Б	В	Г	Д
0	100	101	110	111

Рисунок 1.1 — Пример построения дерева Хаффмана

1.5 Простейшие алгоритмы сжатия: RLE и KWE

Кодирование длин серий (англ. run-length encoding, RLE) или кодирование повторов — алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов. Методом кодирования длин серий могут быть сжаты произвольные файлы с двоичными данными, поскольку спецификации на форматы файлов часто включают в себя повторяющиеся байты в области выравнивания данных.

В основу кодирования KWE положен принцип кодирования лексических единиц исходного документа группами байтов фиксированной длины. Лексическая единица – последовательность символов, справа и слева ограниченная пробелами или символами конца абзаца. Лексической единицей может быть слово. Результат кодирования сводится в таблицу, которая прикладывается к исходному коду и представляет собой словарь. Для англоязычных текстов принято использовать двухбайтовую кодировку слов. В

русском языке много длинных слов, большое количество приставок, суффиксов и окончаний, поэтому не всегда удаётся ограничиться 2-х байтовыми словами.

Эффективность метода зависит от длины документа. Из-за необходимости прикладывать к архиву словарь слов, длина кратких документов не только не уменьшается, но даже возрастает. Алгоритм наиболее эффективен для англоязычных текстовых документов и файлов баз данных.

1.6 Сжатие с потерями

Сжатие данных с потерями — метод сжатия данных, при использовании которого распакованный файл может очень сильно отличаться от оригинала на уровне сравнения «бит в бит», но практически неотличим для человеческого уха или глаза в большинстве практических применений. Преимущество методов сжатия с потерями над методами сжатия без потерь состоит в большой степени сжатия. При использовании сжатия с потерями необходимо учитывать, что повторное сжатие обычно приводит к деградации качества. Однако если повторное сжатие выполняется без каких-либо изменений сжимаемых данных, качество не меняется.

Идея, лежащая в основе всех алгоритмов сжатия с потерями, довольно проста: на первом этапе удалить незначительную информацию, а на втором этапе к оставшимся данным применить наиболее подходящий алгоритм сжатия без потерь. Подходы здесь могут быть разными в зависимости от типа сжимаемых данных. Если мы работаем со звуком, чаще всего удаляют частоты, которые человек просто не способен воспринять.

1.7 Постановка задачи

Моя дипломная работа посвящена анализу методов сжатия информации. В дипломной работе я определяю наиболее эффективный алгоритм для каждого формата данных путем сравнения степеней сжатия и качества файла после компрессии. Для этого будут рассмотрены следующие вопросы: теория сжатия информации, классификация видов сжатия при кодировании, достоинства и недостатки каждого метода, схемы кодирующих устройств. Основными критериями эффективности сжатия является коэффициент и качество сжатия.

2 Анализ и сравнение методов сжатия информации при кодировании

2.1 Сжатие изображений

В настоящее время изображения и иллюстрации используются повсеместно. Проблема, связанная с их большим размером, коснулась каждого пользователя гаджетов и компьютеров. Так, одно полноэкранное полноцветное изображение занимает почти мегабайт. Учитывая то, что обычно при работе мы используем несколько иллюстраций, и то, что они часто бывают гораздо большего размера, держать их в неупакованном виде становится неудобно и невыгодно. В последние годы разработано большое количество различных алгоритмов архивации изображений: использовались как видоизмененные универсальные, так и абсолютно новые алгоритмы, ориентированные только на изображения.

2.1.1 Алгоритм сжатия JPEG

Началось все в 1986 году, когда была создана рабочая группа JPEG, в которую вошли ведущие специалисты в области фотографии со всего мира. Объединенная группа экспертов в области фотографии (Joint Photographic Experts Group) создала метод сжатия изображений фотографического качества, до размеров, приемлемых для передачи по компьютерным сетям. Он основан на удалении из изображения той части информации, которая слабо воспринимается человеческим глазом. Изображение без избыточной информации 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.

Функционирование JPEG Алгоритм JPEG можно разделить на несколько этапов.

1 Дискретизация. Перевод из пространства GRB в цветовое пространство YCbCr, в котором разделены яркостная и цветовая составляющие. На этом шаге потерь не происходит, ведь каждый пиксель по-прежнему состоит из трех компонентов. Только теперь это яркостная компонента и две цветовых.

2 Ресемплинг. Человеческий глаз более чувствителен к изменениям яркости, чем к изменениям цвета. Это помогает убрать часть деталей в цветовых каналах. Каждые 4 цветовых пикселя объединяются в один. Происходят незаметные потери.

3 Разделение на блоки 8x8.

4 Дискретное косинусное преобразование. Алгоритм должен каким-то образом понять, насколько много деталей в каждом блоке. Он определяет насколько много деталей в файле.

Делается такой анализ с помощью дискретного косинусного преобразования. Любое монохромное изображение 8 на 8 пикселей можно представить как смесь из 64 картинок (рисунок 2.1):

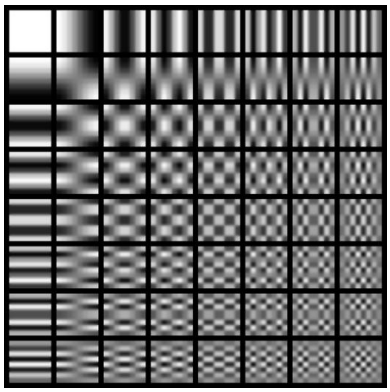


Рисунок 2.1 — Монохромное изображение 8 на 8 пикселей

Дискретное косинусное преобразование (DCT) как раз вычисляет коэффициенты для наложения каждой такой базовой картинки. Всего 64 числа (рисунок 2.2):

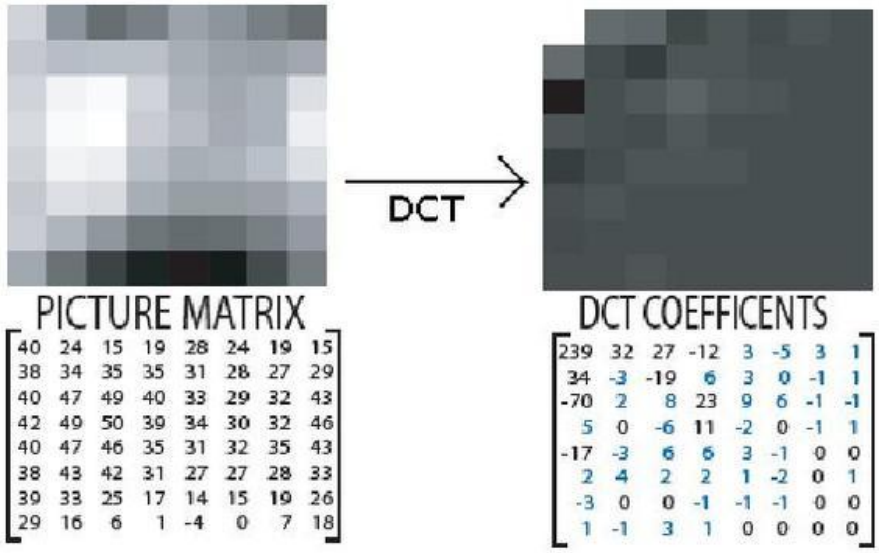


Рисунок 2.2 — Дискретное косинусное преобразование

Здесь коэффициентом является вес, другими словами, важность этой составляющей. Есть те, которые отвечают за плавные переходы. И те, которые отвечают за частые переходы, то есть в изображении много деталей. Как раз, если деталей мало, то коэффициенты у последних будут практически нулевыми, так как блок почти монотонный. На этом этапе и начинается по-настоящему эффективное сжатие.

5 Квантование. Матрица коэффициентов делится на матрицу квантования, которая зависит от настроек сжатия. Например, если вы сохраняете JPEG со 100% сжатием, каждый коэффициент делится на 1 (то есть остается таким же). Если вы сохраняете с меньшим качеством, то каждый коэффициент делится на определенное число (опять же в зависимости от выбранного качества).

Далее, как на рисунке 2.3 производится округление поделенных коэффициентов до целого значения. При сильном сжатии многие из них округляются до нуля:

239	32	27	-12	3	-5	3	0
34	-3	-19	6	3	0	0	0
-70	2	8	23	9	6	0	0
5	0	-6	11	0	0	0	0
-17	-3	6	6	3	0	0	0
0	4	0	0	0	0	0	0
-3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Рисунок 2.3 - Округление коэффициентов до нуля

На этом этапе теряется много информации из-за этих округлений. Но основная суть остается.

6 Сжатие. На рисунке 2.4 сначала матрица коэффициентов сканируется зигзагом:

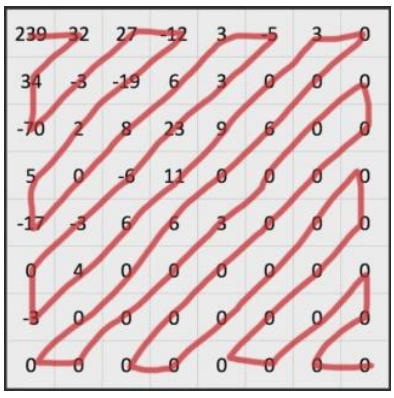


Рисунок 2.4 - Сканирование матрицы зигзагом

Получается последовательность чисел, в конце которой зачастую одни нули: 239, 32, 34, -70, -3, 27, -12, -19, 2, 5, -17, 0, 8, 6, 3, -5, 3, 23, -6,-3, 0, -3, 4, 6, 11, 9, 0, 3, 0, 0, 6, 0, 6, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0 ,0, 0, 0, 0, 0, 0, 0.

Затем последовательность пакуется следующим способом. Если в ней есть длинная череда нулей, не кодируется каждый из них. Вместо этого просто указывается одно число, обозначающее сколько их: 239, 32, 34, -70, -3, 27, -12, -19, 2, 5, -17, (0,1), 8, 6, 3, -5, 3, 23, -6, -3, (0,1), -3, 4, 6, 11, 9, (0,1), 3, (0,2), 6, (0,1), 6, (0,6), 3, (0,24).

Таким образом, в конце сжатая последовательность кодируется кодом Хаффмана. Каждому символу присваивается определенный код. Например, так (таблица 2.1):

Таблица 2.1 - Кодирование Хаффмана

Символ	Код	Встречаемость в фрейме
00000000	00	735 раз
00011001	10	129 раз
00000110	010	97 раз
01001100	110	56 раз
11000010	0110	48 раз
00010001	0111	45 раз

Код присваивается по степени встречаемости в файле. Получается, что нули и другие символы, которых очень много кодируются малым количеством бит, а остальные символы, которые реже встречаются, большим количеством бит. Затем все блоки 8 на 8 складываются в один файл и получается JPEG. На рисунке 2.5 изображены все вышеперечисленные этапы кодирования по алгоритму JPEG:

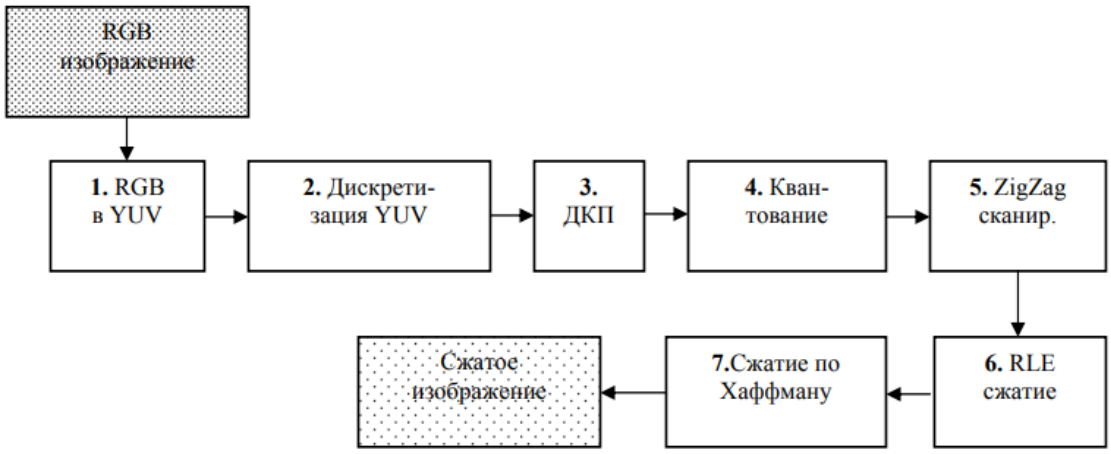


Рисунок 2.5 — Этапы кодирования JPEG

Ниже на рисунке 2.6 представлен пример сжатия фотографии по алгоритму JPEG. Стоит отметить, что даже после трехкратной компрессии качество

картинки не поддается заметному искажению для человеческого глаза. Хотя размер фотографии уменьшился в 3.7 раз. Благодаря этому JPEG и пользуется большой популярностью среди современных алгоритмов сжатия.

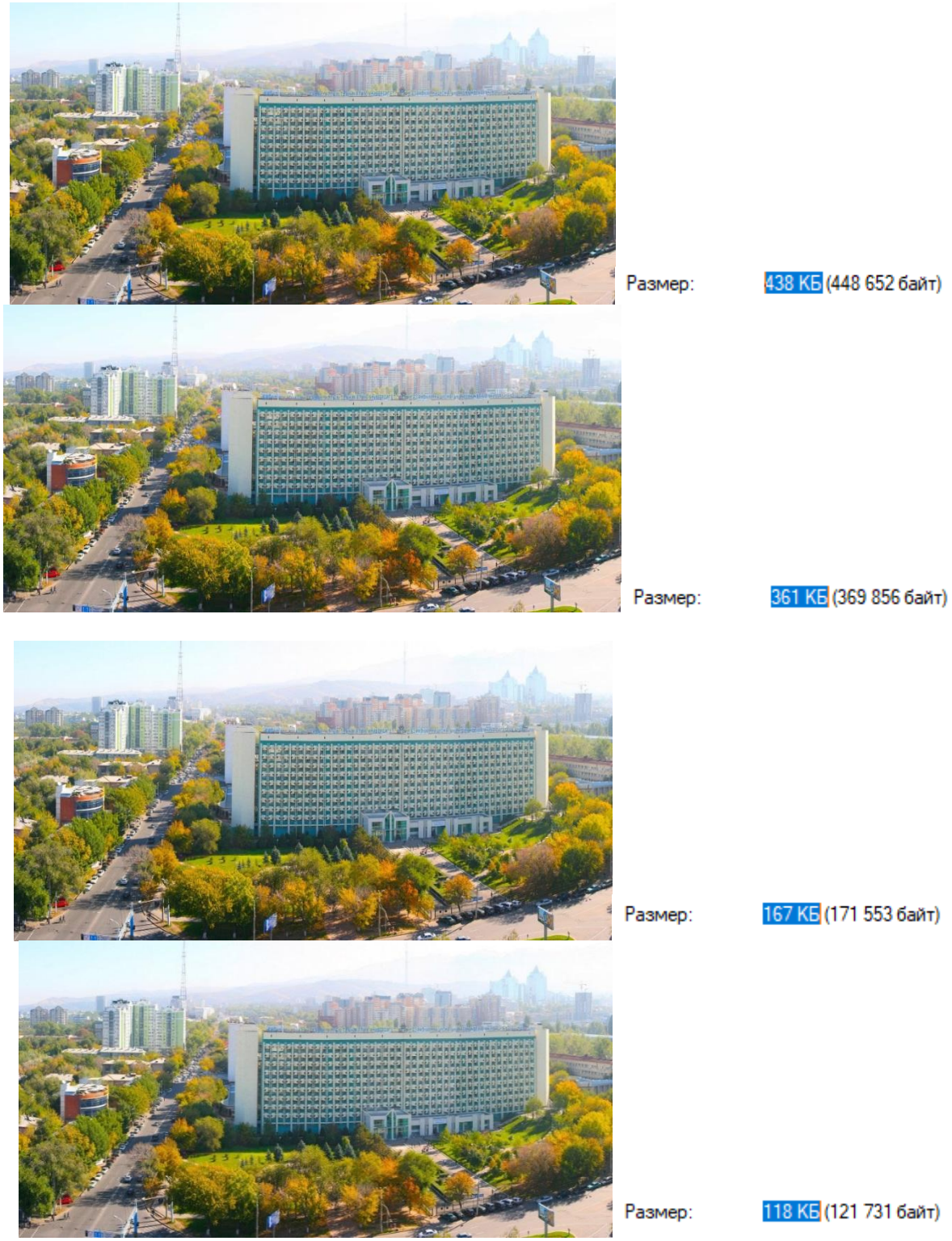


Рисунок 2.6 — Трехкратное сжатие изображения и его размер

2.1.2 Алгоритм фрактального сжатия

Во многом недостаток формата сжатия JPEG состоит в том, что при сжатии никак не учитываются специфика изображения, структура и характерные участки. Именно учет специфики изображения лежит в основе фрактального метода [3].

Фрактал - это структура, выделенная при анализе изображения, и обладающая схожей формой независимо от ее размеров. Многие вещи в природе являются фракталами, например, облака, листья деревьев, снежинки, кровеносная система. В данном формате сжатия используется система итерируемых функций (Iterated Function System - IFS). Наиболее распространённым примером фрактального изображения является изображение папоротника (рисунок 2.7) , которое, состоит из 4-х аффинных преобразований. Фрактальная компрессия – это поиск самоподобных областей и определение для них параметров аффинных преобразований. Данный алгоритм известен тем, что в некоторых случаях позволяет получить очень высокие коэффициенты сжатия при приемлемом визуальном качестве для реальных фотографий природных объектов.

Рисунок 2.7 — Изображение папоротника, сгенерированного с помощью IFS

Сохранение качества при сжатии является не единственным достоинством данного формата. Так, фрактальный архиватор позволяет, например, при распаковке произвольно менять разрешение изображения без появления эффекта зернистости.

Схема кодирования выглядит так:

1 изображение делится на маленькие неперекрывающиеся квадраты, называемые ранговыми блоками;

2 строится пул всевозможных перекрывающихся блоков в четыре раза больших ранговых — доменных блоков;

3 для каждого рангового блока по очереди «примеряем» доменные блоки и ищем такое преобразование, которое делает доменный блок наиболее похожим на текущий ранговый;

4 пара «преобразование-доменный блок» ставится в соответствие ранговому блоку. В закодированное изображение сохраняются коэффициенты преобразования и координаты доменного блока. Содержимое доменного блока нам ни к чему — нам все равно с какой точки начинать сжатие изображения. На рисунке 2.8 ранговый блок обозначен желтым, а соответствующий ему доменный — красным[1].

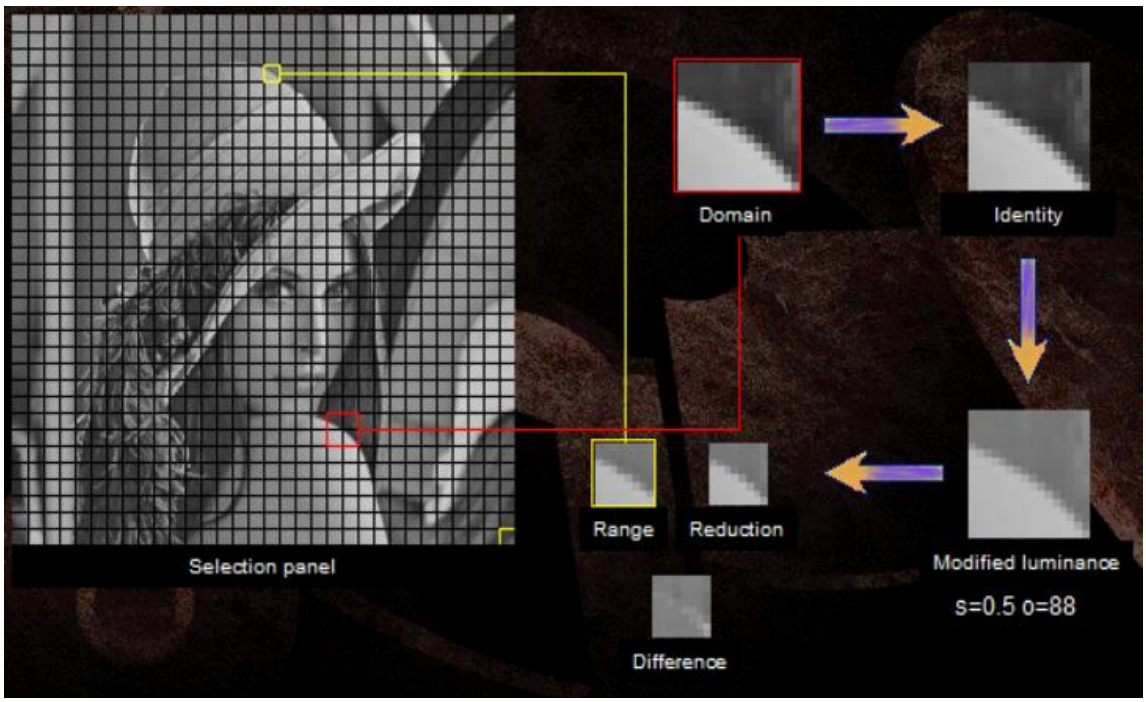


Рисунок 2.8 — Процесс выбора доменных и ранговых блоков

Если алгоритм не может подобрать для какого-либо фрагмента изображения подобный ему, достаточно разбить этот фрагмент на более мелкие и попытаться поискать для них. Однако понятно, что процедуру эту нельзя повторять до бесконечности, иначе количество необходимых преобразований станет так много, что алгоритм перестанет быть алгоритмом компрессии. Следовательно, допускаются потери в какой-то части изображения.

Для фрактального алгоритма компрессии, как и для других алгоритмов сжатия с потерями, очень важны механизмы, с помощью которых можно будет регулировать степень сжатия и степень потерь. К настоящему времени разработан достаточно большой набор таких методов. Во-первых, можно ограничить количество преобразований, заведомо обеспечив степень сжатия не ниже фиксированной величины. Во-вторых, можно потребовать, чтобы в ситуации, когда разница между обрабатываемым фрагментом и наилучшим его приближением будет выше определенного порогового значения, этот фрагмент дробился обязательно (для него обязательно заводится несколько линз). В-

третьих, можно запретить дробить фрагменты размером меньше, допустим, четырех точек. Изменяя пороговые значения и приоритет этих условий, можно очень гибко управлять коэффициентом компрессии изображения: от побитного соответствия, до любой степени сжатия. На рисунке 2.9 показан кодер фрактального сжатия.

Рисунок 2.9 — Функциональная схема кодера фрактального сжатия

2.1.3 Сравнение с JPEG

При сравнении фрактального метода сжатия с форматом сжатия JPEG, во-первых, нужно отметить, что оба алгоритма оперируют 8-битными (в градациях серого) и 24-битными полноцветными изображениями. Оба являются алгоритмами сжатия с потерями и обеспечивают близкие коэффициенты архивации. И у фрактального алгоритма, и у JPEG существует возможность увеличить степень сжатия за счет увеличения потерь.

Различаются алгоритмы во времени необходимом для архивации/разархивации. Так, фрактальный алгоритм сжимает в сотни и даже в тысячи раз дольше, чем JPEG. Распаковка изображения у фрактального сжатия, наоборот, произойдет в 5-10 раз быстрее. Поэтому, если изображение будет сжато только один раз, а передано по сети и распаковано множество раз, то выгодней использовать фрактальный алгоритм.

JPEG использует разложение изображения по косинусоидальным функциям, поэтому потери в нем проявляются в волнах и ореолах на границе резких переходов цветов. Именно за этот эффект его не любят использовать при сжатии изображений, которые готовят для качественной печати: там этот эффект может стать очень заметен. Фрактальный алгоритм избавлен от этого недостатка.

Более того, при печати изображения сжатым форматом JPEG каждый раз приходится выполнять операцию масштабирования, поскольку растр печатающего устройства не совпадает с растром изображения. При преобразовании также может возникнуть несколько неприятных эффектов, с которыми можно бороться либо масштабируя изображение программно, либо снабжая устройство печати своим процессором, и набором программ обработки изображений. При использовании фрактального алгоритма таких проблем практически не возникает.

Вытеснение JPEG фрактальным алгоритмом в повсеместном использовании произойдет ещё не скоро (хотя бы в силу низкой скорости архивации последнего), однако в области приложений мультимедиа, в компьютерных играх его использование вполне оправдано.

2.1.4 Сжатие по алгоритму GIF

До недавнего времени формат сжатия GIF являлся самым популярным среди форматов, использующихся для сжатия изображений. Расшифровывается GIF как Graphics Interchange Format (Формат графического обмена). Особенностями данного формата является поддержка до 256 цветов, хранение нескольких изображений в одном файле, с помощью которого создаются анимации картинок (рисунок 2.10). Во многих случаях это свойство используется для создания мультипликаций в web-браузерах. Благодаря лучшему сжатию и большей глубине цвета формат GIF был заменен на формат JPEG. Но формат GIF в настоящее время пользуется популярностью среди других приложений, однако распространению сильно мешают юридические препятствия.

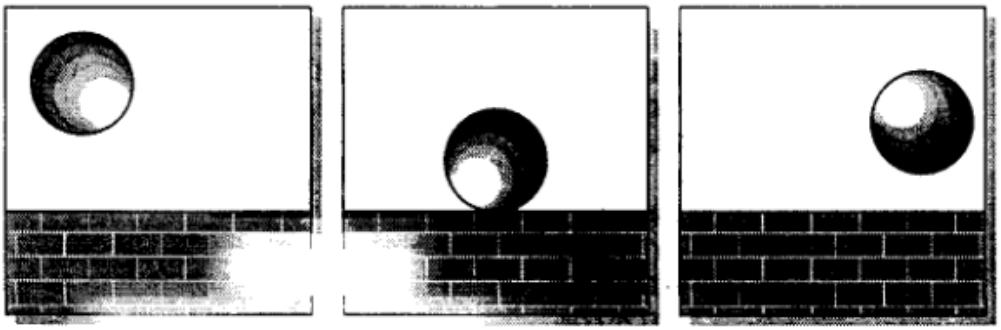


Рисунок 2.10 — Примеры gif-анимации

Формат GIF хранит многобайтовые целые числа с младшим байтом на первом месте (прямой порядок байтов). Строки битов считываются в направлении от самого младшего бита к самому старшему биту. В битовых строках, которые пересекают границы байта, биты второго байта старше битов первого байта [2].

Структура файлов формата JPEG имеет блочный характер. Другими словами, все файлы формата состоят из отдельных блоков и никак не связаны друг с другом. Некоторые программы, не имеющие возможность распознать тип блоков, пропускают их. Для этого каждый блок содержит у себя в заголовкеего размер. Анимацию составляют все изображения, которые идут одно за другим и сменяют друг друга, создавая при этом иллюзию движения. В файле могут также находиться и другие блоки, и неважно будут они находиться до или после (или даже между ними):

1 комментарии. Скрытый текст виден только через специальные программы, такие как аниматоры GIF;

2 простой. Строки символов с ограничениями форматирования. В настоящее время не используется;

3 графические блоки управления, которые устанавливают выходные параметры для отдельных изображений;

4 глобальные и локальные цветовые палитры, определяющие, какие цвета будут на изображениях;

5 специальные блоки, которые могут использоваться только программами , которые знают об их существовании.

Минимально необходимый набор блоков - простейший не анимированный GIF (рисунок 2.11):

Рисунок 2.11 — Структура файла GIF

Дескриптор глобального экрана указывает размеры области, а также цвет фона. Дескрипторы отдельных изображений определяют, в каком месте логического экрана должно располагаться изображение.

Глобальная палитра цветов. Единичные иллюстрации файла имеют все шансы использовать глобальную палитру цветов, либо без помощи других установить свою. Плюсом таблицы расцветок считается сокращение файла, также легкость во применении во концепциях.

Типы блоков. Затем располагается переменная часть GIF-файла. Изображение формируется из блоков, определяющихся 1-байтовым кодом в истоке блока.

Завершитель. Блок завершителя отмечает конец файла. Этот однобайтовый блок состоит исключительно из кода блока.

На рисунке 2.12 представлен один и тот же портрет, сжатый двумя разными алгоритмами: JPEG и GIF.


Рисунок 2.12 — Сравнение изображений JPEG и GIF

2.1.5 Формат сжатия PNG

PNG (Portable Network Graphics) – переносимая сетевая графика. Данный алгоритм является самым новым алгоритмом из всех перечисленных. Сейчас данный алгоритм широко применим в сети интернет при передаче изображений

PNG-изображения поддерживают использование Альфа-канала для управления прозрачностью изображения. Он, в свою очередь, позволяет объединить изображение с его фоном. Каждое значение пикселя содержит дополнительное Альфа-значение (alpha value), размер которого в битах равен глубине цвета изображения. Цветовая модель RGB с Альфа-каналом может использоваться только при глубине цвета равной 8 и 16 битам. Нулевоезначение Альфа-канала означает, что пиксел виден через изображение. Фон полностью закрыт изображением.

Файлы PNG расположены в последовательности блоков, называемых стандартными порциями PNG. Существует три источника, которые определяют типы порций. Некоторые типы порций определены в стандарте PNG; это самые важные части, которые декодер должен обработать. Команда разработчиков PNG также ведет список зарегистрированных открытых типов порций. Когда кто-то создает новый тип, который может быть полезен для широкой публики, разработчик может представить свой новый тип для рассмотрения и последующего возможного добавления в список типов предоставления PNG.

Плюсы использования формата PNG заключаются в следующем:

1) сжатие происходит без потерь в качестве детализации;

2) глубина цвета не ограничена;

3) регулировка размера файла в зависимости от палитры;

4) наличие канала прозрачности.

В отличии от GIF, у PNG есть поддержка градаций прозрачности за счет дополнительного альфа-канала. Обычно на прозрачность указываетшахматный фон, как видно из расположенного ниже изображения.

Внешне файлы в формате PNG практически не отличаются от JPG-изображений. Если важна прозрачность, лучше выбирать именно этот формат. В случае если необходимо сжать изображение без потерь, многократноредактируя

его, данный формат также превзойдет JPEG. На рисунке 2.13 можно сравнить два изображения, сжатых разными методами.

Рисунок 2.13 — Изображения формата png

Но также у данного формата есть свои недостатки. Во-первых, большой размер самого файла, из-за которого возникнут проблемы при работе со сложными реалистичными изображениями. Во-вторых, невозможно создать анимацию. В обоих случаях предпочтительным является формат JPG.

2.1.6 Сравнительный анализ алгоритмов сжатия изображений

Учитывая все критерии при выборе лучшего метода, включая степень сжатия, качество изображения и область использования, самым оптимальным алгоритмом выбран JPEG. Используют и оптимизируют формат JPG чаще других форматов по причине того, что большинство современных изображений содержат большое количество цветовых оттенков и градиентов. При сжатии даже больше чем на 50-70%, наши глаза могут и не заметить отличий от оригинала большого размера. К тому же, очень низкая скорость кодирования фрактального сжатия не делает его эффективным. Но если мы говорим о хранении изображений, стоит признать, что JPEG с данной задачей справится плохо, так как при многоразовом сжатии качество заметно ухудшится. С этой задачей отлично справится PNG. Он не теряет качество при сжатии, поэтому используется для хранения изображений, содержащих текст: скриншотов, чертежей, сканированного текста, комиксов. По качеству цветового отображения превосходит .jpeg и .gif, но размер файла будет на 30-40% больше. В таблице 2.2 приведено сравнение всех вышеперечисленных алгоритмов сжатия изображений.

Таблица 2.2 - Сравнения алгоритмов сжатия изображений

Формат изображения	Доступные цвета	Сжатие	Размер файла (средние значения)	Лучше всего для:
Фрактальное сжатие	16 млн	С потерями	Большой	Хранения
JPEG	16,7 млн	С потерями	Небольшой (<1МБ)	Интернета
GIF	256	Без потерь	Небольшой (<1 МБ)	Анимации
PNG	16млн +прозрачность	Без потерь	Большой (<2МБ)	Хранения и редактирования

2.2 Сжатие видеоданных

Для передачи цифрового видео от источника к получателю, необходима целая цепь незаменимых компонентов и процессов. Главными функциями этой цепи являются кодирование, т.е. сжатие данных и декодирование. При этом видео сжимается в размере необходимом для дальнейшей передачи и хранения, и потом восстанавливается для отображения на видеоэкране.

Чтобы достигнуть высокой степени сжатия необходимо применить компрессию с потерями. В процессе сжатия видео с потерями используются следующие принципы: игнорируется незначительное изменение цвета картинки в соседних пикселях, удаляется избыточность в цветовых палитрах, максимально используется сходство между кадрами - незначительное изменение соседних кадров.

2.2.1 Формат сжатия Motion JPEG

Motion JPEG можно представить как последовательность кадров JPEG. Motion JPEG пользуется большой популярностью среди стандартов сжатия видео. Скорость составляет 16 кадров в секунду, тем самым человеческий глаз воспринимает поток изображений как непрерывное видео.

Характеристики Motion-JPEG: Сжимает видео в 5-10 раз, быстро сжимает и разжимает видео, может редактировать (накладывать эффекты), легко реализуем (за счет низкой стоимости аппаратуры и простоты реализации). Отлично делать стоп-кадры (поэтому используется в системах видеонаблюдения). Но по сравнению с другими алгоритмами имеет не очень высокую степень сжатия.

Область применения формата: цифровые камеры, камеры IP , а также веб – камера; PlayStation консоли и веб - браузеры, такие как Safari , Google Chrome, Mozilla Firefox.

2.2.2 Алгоритм MPEG-1

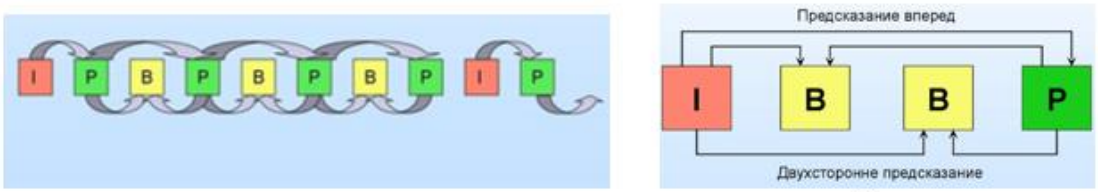
Алгоритм MPEG-1 в целом очень похож с вышеописанным стандартом Motion JPEG.

Характеристики MPEG-1: Скорость потока составляет 1.5 Мбит/с; также очень прост в аппаратной реализации; практически вся аппаратура поддерживает этот формат. Также характеризуется невысокой степенью сжатия. Во время сжатия видео, видеокадры сжимаются с использованием различных алгоритмов, называемых типами изображений или типами кадров.

Рисунок 2.14 — Типы кадров

Основной принцип MPEG сжатия заключается в сравнении двух последовательных образов и передача по сети только небольшого количества кадров, содержащих полную информацию об изображении. Остальные кадры (промежуточные кадры) содержат только отличия этого кадра от предыдущего. На рисунке 2.14 приведена классификация кадров. Иногда применяют двунаправленные кадры, информация в которых кодируется на основании предыдущего и последующего кадров, что позволяет дополнительно повысить степень сжатия видео. Во всех форматах MPEG используется метод компенсации движения.

Также в области сжатия видео есть понятие GOP-Group of Pictures (рисунок 2.15) , который представляет собой совокупностью кадров, состоящий из трех типов кадров.



а б

а - Group of Pictures, б - Последовательность кадров GOP
Рисунок 2.15

Рисунок 2.16 описывает общий принцип сжатия и удаление избыточности в алгоритме JPEG.

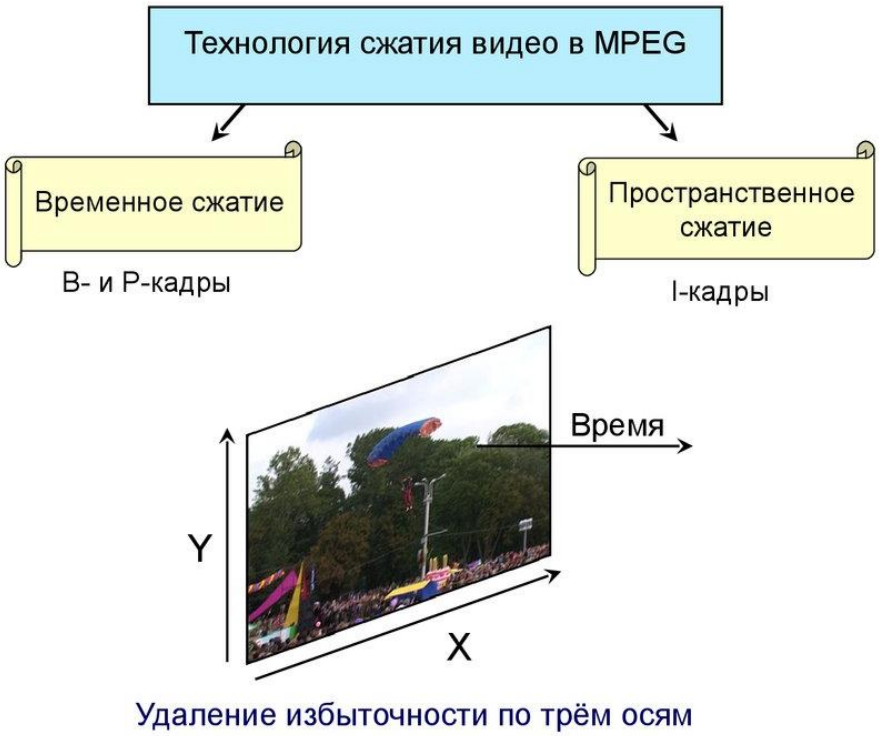


Рисунок 2.16 — Удаление избыточности в алгоритме MPEG

MPEG сжатие позволяет значительно снизить объемы передаваемой по сети информации по сравнению с MotionJPEG. Основа кодирования у группы алгоритмов MPEG общая. На сегодняшний день нам известно три стандарта MPEG: MPEG-1, MPEG-2, MPEG-4. На рисунке 2.17 изображена схема кодера MPEG.

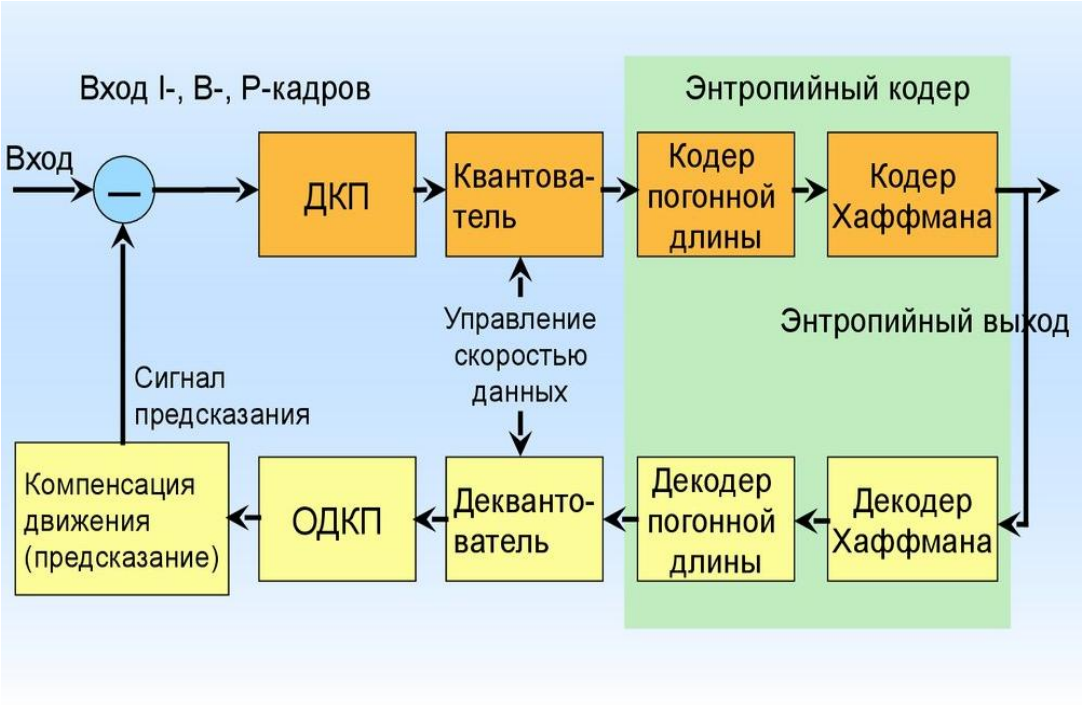


Рисунок 2.17 — Элементы кодера MPEG

Преимущества MPEG-2 по отношению к MPEG-1:

1) масштабируемость уровней качества изображения в одном видеопотоке;
2) точность векторов движения увеличена до 1/2 пикселя;
3) возможность выбора точности дискретного косинусного преобразования;
4) дополнительные режимы прогнозирования.

2.2.3 Стандарты видеосжатия MPEG-4 и H.264

Необходимость в лучшем сжатии породило разработку дальнейших стандартов видеосжатия MPEG-4 и H.264. Оба стандарта имеют общее происхождение и характеристики; были разработаны на основе предыдущих стандартов сжатия. Впрочем, они разрабатывают стандарты в совершенно разных направлениях. MPEG-4 отличается тем, что кодек работает с объектами в произвольной форме, в то время как предыдущие форматы разделяют изображение на прямоугольники. Например, птица, летящая в небе, будет отслеживаться как отдельный предмет, перемещающаяся относительно неподвижного объекта - неба. Также среди других стандартов MPEG-4 выделяется широким диапазоном скоростей передачи (от 5 кбит/с до 10 Мбит/с). Более того, усовершенствованный алгоритм сжатия, высокое качество и эффективность делают алгоритм самым распространенным, несмотря на сложность реализации.

Стандарт H.264 также выполняет те же действия, что и предыдущие стандарты, но с более высокой эффективностью и стабильностью, обеспечивая совместимость со всеми широко распространенными типами приложений,

такими как широкое телевещание, хранение визуальной информации и передача потокового видео.

2.2.4 Стандарт H.261

Стандарт H.261 специфицирует кодирование и декодирование видеопотока для передачи по каналу p*64 Кбит, где p=1..30. В качестве канала может выступать, например, несколько телефонных линий.

Характеристики H.26: Поток 0,01-2 Мбит/с(p*64кбит/с), где p=1..30; Прост в аппаратной реализации; как и два вышеуказанных алгоритма имеет невысокую степень сжатия. Не заменим в процессе сжатия видео для видеоконференций. Минус этого алгоритма состоит в том, что он специализируется на видео с незначительными движениями в кадре.

2.2.5 Стандарт H.263

H.263 представляет собой продолжение формата H.261 и является его расширенной и дополненной версией. Специализируется на передаче видео с постоянной, фиксированной скоростью. Фиксированная скорость сопровождается падение качества видео. H.263 был разработан для видео конференц-связи, а не для наблюдения, где отображение деталей являются более важным, чем скорость передачи данных.

Ниже указаны все важные изменения от предыдущего стандарта:

1) за место кодирования Хаффмана применяется арифметическое кодирование, тем самым повышает степень сжатия на 5-10%;
2) появилась опция задания вектора смещения для каждого блока 8x8 в макроблоке, что в ряде случаев существенно увеличивает сжатие;
3) внедрились b-кадры, позволяющие увеличить степень сжатия, за счет усложнения и увеличения времени работы декодера;
4) применяется особый режим квантования и специальная таблица хаффмана для улучшения сжатия i-кадров в ряде случаев;
5) поменялось разрешение и видоизменился базовый кадр.

2.2.6 Сравнение алгоритмов видео

В таблице 2.3 приведено сравнение всех вышеперечисленных алгоритмов сжатия видеоданных.

При сравнении различных методов кодирования видео учитывают выходной битрейт и оценивают качество восстановленного последекодирования видео потока. Эффективнее всего оценку качества дает человеческий глаз. Сжатие можно рассматривать отличным, в случае если глаза не отличил видео до и после сжатия. Но при сжатии с потерями чаще всего добавляются различные помехи и искажения, которые заметны нашему глазу.

Таблица 2.3 — Сравнение вышеупомянутых алгоритмов сжатия

Название стандарта	Область применения	Первичный алгоритм	Вторичный алгоритм	Поток	Достоинства и недостатки
H.261	ISDN-видеоконференции, аппаратные кодеки	DCT и квантование межкадровой разности	Метод Хаффмана	0.04-2 Мбит/с (p*64 Кбит/с, где p от 1 до 30)	Простота реализации Недостатки: низкая степень сжатия, плохая компенсация движения
H.263	Видеоконференции по широкополосным каналам	DCT	Арифметическое кодирование	0.04-2 Мбит/с (p*64 Кбит/с, где p от 1 до 30)	Улучшенный алгоритм компенсации движения, более эффективный вторичный алгоритм
MPEG-1	Хранение видео на CD-ROM , Video CD	DCT	Код Хаффмана	1,5 Мбит/с	Простота реализации Недостатки: низкая степень сжатия, недостаточная гибкость
MPEG-2	Хранение на DVD, Спутниковое телевидение, кабельное телевидение, архивирование	DCT	Код Хаффмана	3-15 Мбит/с	Простота реализации Недостатки: недостаточная степень сжатия, малая гибкость
MPEG-4	Video CD второго поколения	DCT либо DWT и квантование межкадровой разности	CAVLC или CABAC	Универсальный	Ориентированная работа с потоком данных Недостатки: сложность реализации

MPEG-4, будучи самым новым стандартом, использует наиболее эффективные алгоритмы сжатия данных. В этом отношении, форматы MPEG имеют преимущество по сравнению с M-JPEG в 2.5 – 7 раз. Но это достигается за счет более сложных алгоритмов, требующих большего времени на обработку видео, что приводит к значительным задержкам (до 3-4 секунд) между реальным событием и отображением его на экране.

В случаях, когда нужно только записывать или просматривать видео, MPEG-4 идеальный выбор, но если необходимо анализировать изображение

предпочтение стоит отдать Motion-JPEG, позволяющий найти компромисс между качеством и скоростью передачи видео изображения.

2.3 Алгоритмы сжатия аудиоданных

Качество звука всех цифровых музыкальных файлов измеряется его битовой скоростью(bit rate), отображаемой как kbps. Чем выше скорость передачи данных, тем больше данных у файла и тем лучше звучит MP3. Наиболее распространенные скорости передачи данных — 128 кбит/с, 192 кбит/с и 256 кбит/с.

2.3.1 MP3 сжатие

MP3 (формально MPEG-1 Audio Layer III) является одним из самых популярных форматов сжатия для аудио. Основной принцип сжатия по алгоритму MP3 заключается в неточных приближениях и частичного отбрасывания данных. Это уменьшает размер аудиофайла примерно в 10 раз. Данный алгоритм удобен в передаче, хранении и скачивании на плеер. К тому же, чуть ли не всё современное оборудование поддерживает данный формат аудио. Минимально возможное значение битрейта — 32 килобита в секунду, максимальное — 320 килобита в секунду. Чем меньше величина битрейта, тем меньше будет размер файла. Например, одна минута аудио с максимальным битрейтом 320 килобита в секунду, занимает чуть больше 2 Мбайт, а при значении 128 килобита в секунду — примерно 0,9 Мбайт, в то время как минута аудиоданных WAV весит больше 10 Мбайт.

2.3.2 Алгоритм сжатия аудио FLAC

Free Lossless Audio Codec — свободный аудио-кодек без потерь, первый выпуск которого был 20 июля 2001 года. Алгоритм FLAC описывает сигнал, таким образом, чтобы разность, полученная после вычитания функции из оригинала, могла быть закодированной минимальным количеством бит. Когда эту функцию подобрали, алгоритм отнимает аппроксимацию от оригинала с целью получить ошибочный (остаточный) сигнал, который далее закодируется без потерь. Сжатие FLAC особенно хорошо при работе с записями голоса.

Алгоритм сжимает аудио до 60%. Формат очень удобен для ежедневого прослушивания и архивирования, поскольку алгоритм поддерживается тегами и встроенными обложками. Более того, имеет функцию быстрой смены места воспроизведения. Популярным данный формат делает легкодоступность и поддержка различных программ.

Данный алгоритм незаменим для аудиослушателей, желающих сохранить свои аудио в случае потери или повреждении файла. FLAC может сделать

неотличимую копию оригинала, в то время как MP3 не имеет такой возможности.

Информация в формате FLAC представляется в виде блоков (фреймов), независимых друг от друга. Скорость кодирования может быть разной, все зависит от степени сжатия данных. Но декомпрессия производится крайне быстро

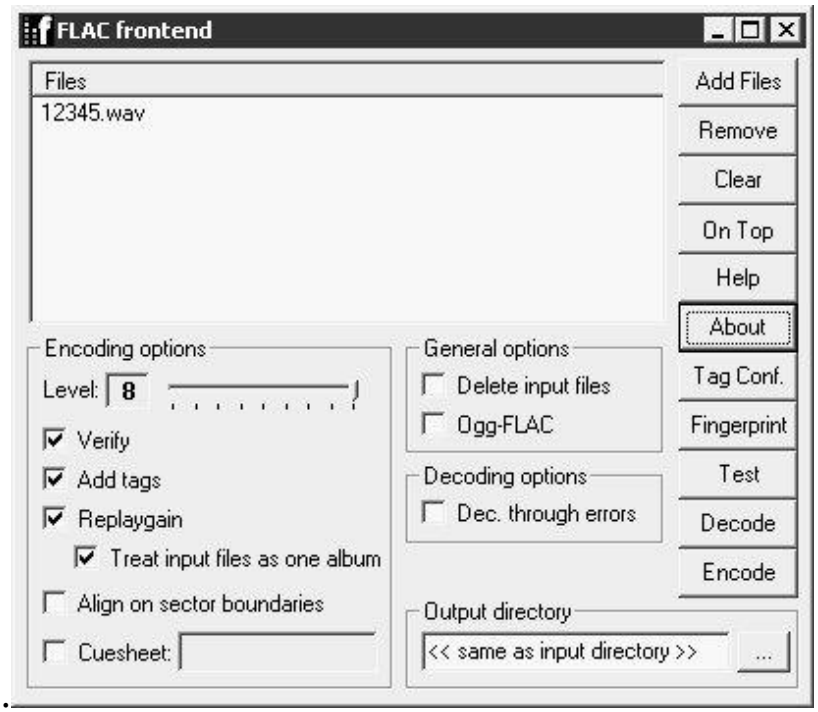


Рисунок 2.18 — Frontend

Утилита FLAC включает в себя кодек и программную оболочку для кодека – Frontend (рисунок 2.18). Работать с кодеком просто: необходимо добавить файлы в окно программы с помощью кнопки «Добавить файлы», настроить параметры кодирования и нажать кнопку «Кодировать»: программа создаст файл FLAC.

Ниже приведены основные параметры кодирования:

1) параметр Level (Уровень) отвечает за уровень сжатия данных. Он может варьироваться от 0 до 8. Следовательно, чем выше уровень сжатия, тем меньше вес выходного файла;

2) параметр verify(проверить) проверяет выходные файлы;

3) параметр add tags (добавить теги) добавляет теги в готовый файл (например, они могут содержать название песни, автора и т. д.); вы можете настроить их, нажав на кнопку confel label. (настройка этикетки);

4) параметр replaygain (уровень проигрывания) добавляет в файлы параметр, который показывает уровень громкости файла;

5) группа параметров general options (общие параметры);

6) параметр output directory (выходная директория) содержит путь к выходным файлам;

7) в группе параметров decoding options (функции декодирования) есть параметр «декодировать несмотря на ошибки».

2.3.3 Алгоритмом сжатия без потерь ALAC

Apple Lossless Audio Codec, ALAC является алгоритмом сжатия без потерь, был разработан компанией Apple 28 апреля 2004 года. Поддержка в iPod,iTunes, Mac OS и iOS устройствах является главным преимуществом перед другими форматами сжатия аудиоданных. Сжимаются ALAC файлы от 40 до 60% оригинала. Его сжатие менее эффективно по сравнению с FLAC. Все-таки качество не отличается от исходного звука. Он бесплатный и легкодоступный. Также при желании мы можем конвертировать файл в любой другой формат, так как музыка хранится оригинальном виде. Более того, есть возможность восстановить поврежденные аудиозаписи компакт-дисков.

2.3.4 Алгоритмом сжатия AAC

Advanced Audio Coding, также известный как AAC, похож на MP3, хотя он немного эффективнее. Это означает, что вы можете иметь файлы, занимающие меньше места, но с тем же качеством звука, что и MP3. Имеет слегка улучшенные звуковые характеристики и большую степень сжатия. Применяется на Android, iOS, iTunes, YouTube, Nintendo и последних версиях PlayStation.

Причины, которые позволяют файлам в формате AAC звучать лучше, чем MP3:

1) при кодировании в AAC используются семпы с более широким охватом частот от 8 до 96 кГц, при кодировании в mp3 используются семплы с охватом от 16 до 48 кГц;

2) могут быть закодированы до 48 каналов, в то время как в MP3 кодируются 2 канала в режиме MPEG-1 и до 5.1 каналов в режиме MPEG-2

3) используются произвольные битовые скорости и переменная длина кадра;

4) большая точность при кодировании звука AAC обеспечивается за счёт использования меньшего размера семплов, которые равны 128 или 120 единиц, в то время как для кодирования в mp3 используются семплы размером 192 единиц. В данном случае меньший размер семлпа позволяет кодировать с большей точностью, т.к. исходная звуковая волна является аналоговым сигналом, и для более точного её описания необходимо производить её измерение как можно чаще. Поэтому меньший размер семлпа лучше, т.к. позволяет более точно описать аналоговый характер звуковой волны в цифровом виде;

5) эффективнее обрабатываются звуковые частоты выше 16 кГц. Напомню, что в формате mp3 эти частоты попросту удаляются

2.3.5 Анализ сравнения методов сжатия аудиоданных

Теперь, исходя из всех достоинств и недостатков, мы можем провести анализ, и определить, какой метод будет эффективнее всего. Важными критериями при сравнении алгоритмов сжатия является степень сжатия, качество и область применения форматов музыки. Если наша цель заключается в обычном прослушивание аудио, то MP3 и AAC — это самый оптимальный выбор. Они оба совместимы с большинством проигрывателей, и к тому же очень сложно отличить сжатые аудиофайлы от исходных. Но все же MP3 обеспечивает самое лучшее сжатие. В последнее время этот формат завоевал огромную популярность. В другом случае, если мы нуждаемся в хранении музыки, и в дальнейшем её преобразовании, то лучше будет хранить в формате без потерь FLAC. В противном случае, если мы воспользуемся другими форматами для преобразования, например, AAC или MP3, то заметим значительное ухудшение качества. Поэтому для архивации рекомендуется FLAC.

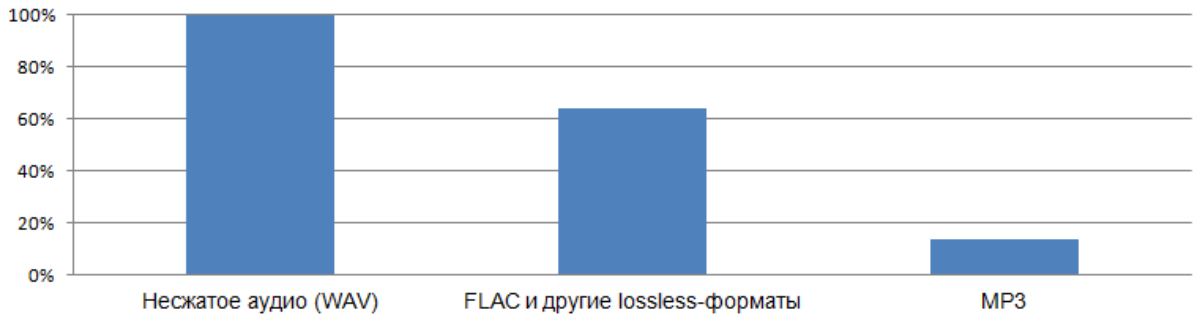


Рисунок 2.19 — Приблизительное соотношение размера файлов WAV, FLAC и MP3

3 Расчет степени сжатия для изображений видео, аудиоданных

3.1 Определение степени сжатия по алгоритмам Шенона-Фано и Хаффмана

А) Рассчитаем степень сжатия по алгоритму Шеннона-Фано. Нам дана последовательность
'cccccccccccccccccccccccccccccceeeeeeeeeeeeeeeeeeeeeeeeebbbbbbbbbbbbbbbbbbbffffffffffaaaaaaaaaaddddd'.
Таблица состоит из двух граф: символы последовательности c(i) и их вероятности p(c(i)). Записываем символы в порядке убывания их вероятностей.

c(i)	p(c(i))
c	30/100
e	25/100
b	20/100
f	10/100
a	10/100
d	5/100

Далее, таблицу символов делим на две группы так, чтобы каждая из групп имела приблизительно одинаковую частоту по сумме символов. Первой группе назначается начало кода в '0', второй в '1'. Для вычисления следующих бит кодов символов, данная процедура повторяется рекурсивно для каждой группы, в которой больше одного символа. Таким образом, для нашего случая получаем следующие коды символов:

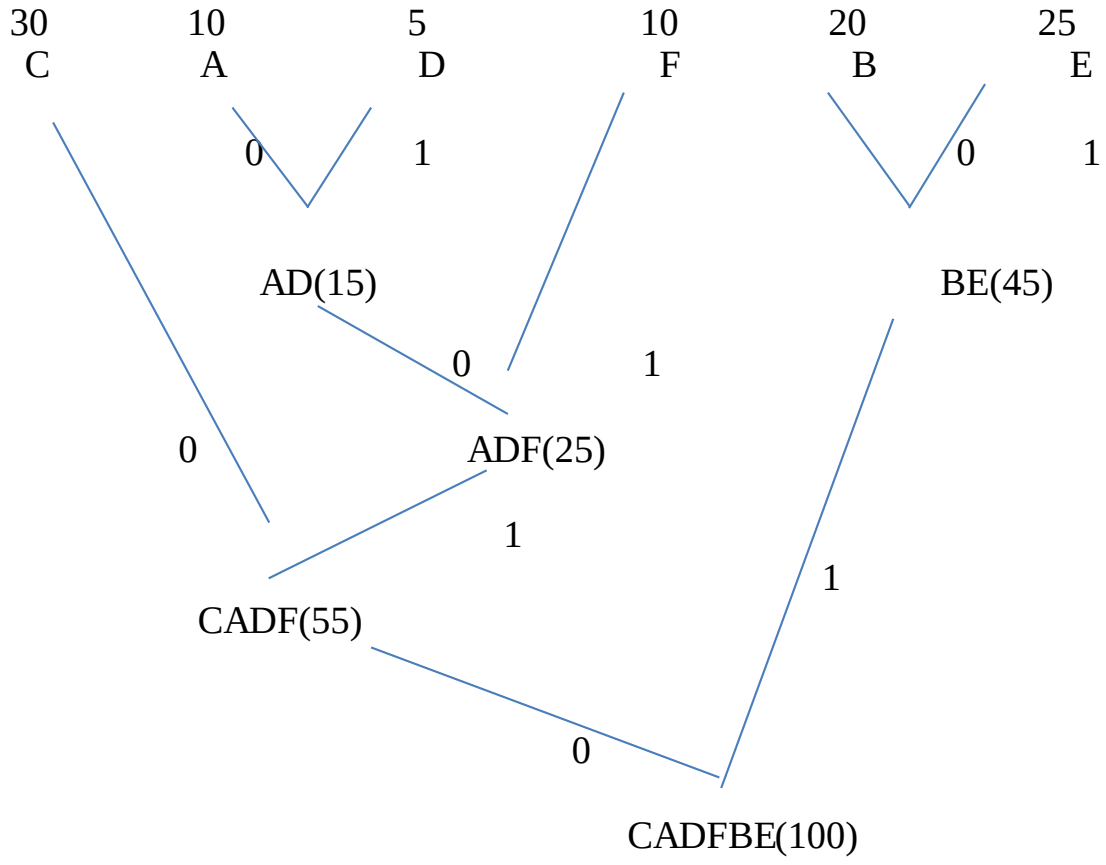
Символ	Код
c	00
e	01
b	10
f	110
a	1110
d	11110

Используя полученную таблицу кодов, кодируем входной поток - заменяем каждый символ соответствующим кодом. В сжатом виде, наша последовательность принимает вид:
0000000000000000000000000000000000000000000000000000000001010101
010101010101010101010101010101010101010101101010101010101010101010
10101010101010110110110110110110110110110111011101110111011101110
11101110111011101111011110111101111011110 длиной в 225 бит.

Учитывая, что оригинал имел длину равную 800 бит, получаем коэффициент сжатия ~28% - не так уж и плохо.

Б) Мы имеем файл длинной в 100 байт и имеющий 6 различных символов в себе. Мы подсчитали вхождение каждого из символов в файл и получили следующее:

Символ	A	B	C	D	E	F
Число вхождений	10	20	30	5	25	10



Символ	Код	Вес
C	00	2 бита
A	0100	4 бита
D	0101	4 бита
F	011	3 бита
B	10	2 бита
E	11	2 бита

Каждый символ изначально представлялся 8-ю битами (один байт), и так как мы уменьшили число битов необходимых для представления каждого

символа, мы, следовательно, уменьшили размер выходного файла. Сжатие получается следующим образом:

Частота	Первоначально	Уплотненные биты	Уменьшено на
C	30 x 8 = 240	30 x 2 = 60	180
A	10 x 8 = 80	10 x 3 = 30	50
D	5 x 8 = 40	5 x 4 = 20	20
F	10 x 8 = 80	10 x 4 = 40	40
B	20 x 8 = 160	20 x 2 = 40	120
E	25 x 8 = 200	25 x 2 = 50	150

Первоначальный размер файла : 100 байт - 800 бит;
Размер сжатого файла : 30 байт - 240 бит;
240 - 30% из 800 , так что мы сжали этот файл на 70%.

3.2 Расчет степени сжатия изображения по алгоритму RLE

Рассмотрим рисунок, содержащий текст чёрного цвета на сплошном белом фоне. При построчном чтении пикселей такого изображения будут встречаться серии белых (фон) и чёрных (буквы) пикселей. Буквой B обозначим чёрный пиксель, а буквой W — белый. Рассмотрим некую произвольную строку изображения длиной 47 символов:

WWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW

Посчитаем количество символов:

4 символа «B»;

43 символов «W».

Итого найдено 5 серий. Заменим серии на число повторов и сам повторяющийся символ:

9W3B24W1B14W.

Получилась последовательность из 18 символов. Исходная последовательность состояла из 47 символов. Данные были сжаты в 47/18≈2.61 раза.

Ниже изображены кодирующие устройства, построенные в программе MatLAB, и выполняющие функцию кодирования.

3.3 Модель реализации кодирующих устройств сжатия

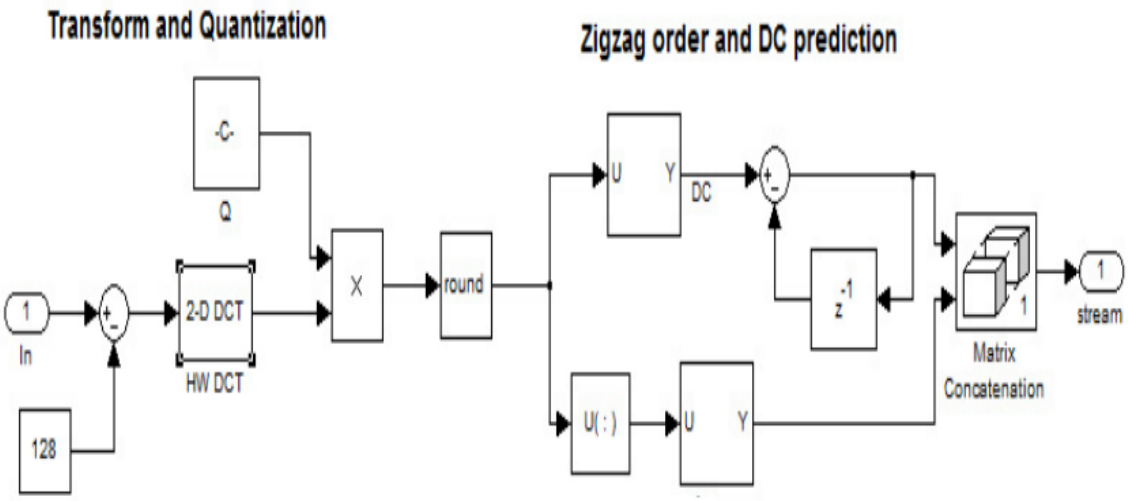


Рисунок 3.1 — Кодер для формата изображений JPEG

В кодере, изображенном на рисунке 3.1, каждое изображение разбивается на неперекрывающиеся блоки размером 8x8 элементов и к каждому из них применяется ДКП. После этого каждый из 64 коэффициентов ДКП квантуется со значением квантования, подобранным отдельно для каждого коэффициента в соответствии с критерием зрительного восприятия. После квантования самый младший ДКП-коэффициент (DC), связанный со средней интенсивностью блока, кодируется с использованием дифференциальной импульсно-кодовой модуляции. Оставшиеся ненулевыми после квантования остальные коэффициенты ДКП (AC) сканируются "зигзагом" и кодируются с помощью энтропийного кодирования (кодом переменной длины и с применением метода Хаффмана или арифметического кодирования, опять же на основе заданной табличной спецификации).

На рисунке 3.2 изображен кодер MPEG-4. На вход кодера поступают исходные видеоданные, например, цифровой телевизионный сигнал. На выходе кодера формируется элементарный поток видеоданных (блоки 8*8). Далее эти блоки поступают в ДКП - блок прямого дискретного косинусного преобразования. Следующий этап квантование, где происходит основная часть сжатия. Алгоритм должен каким-то образом понять, насколько много деталей в каждом блоке. Он определяет насколько много деталей в файле. Inverse quantization - блок обратного дискретного косинусного преобразования. Intra predicted 8x8 block- блоки, выполняющие формирование предсказанных изображений в разных режимах кодирования.

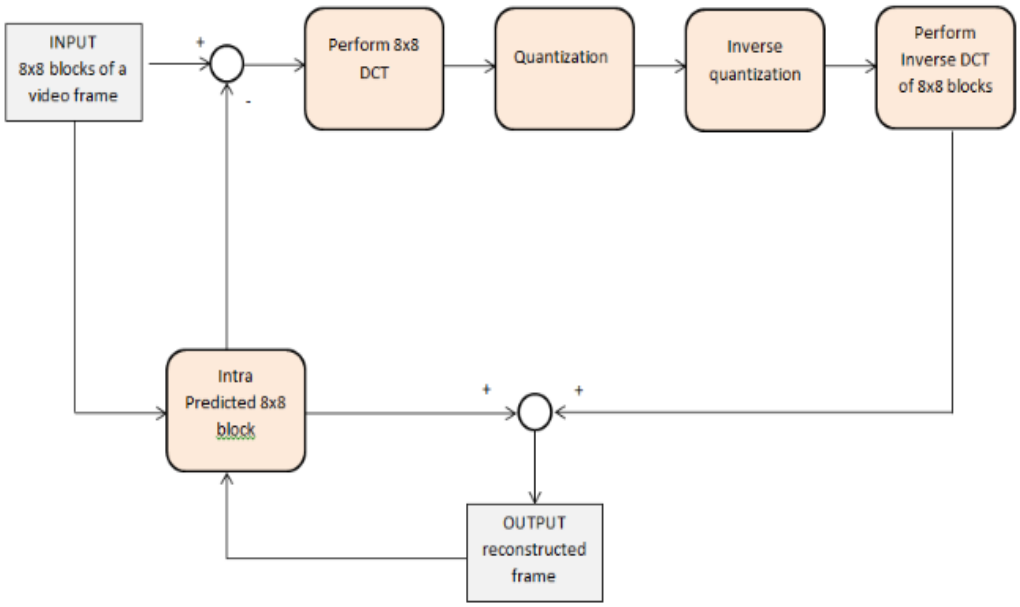


Рисунок 3.2 — Кодирующее устройство алгоритма MPEG-4

Реконструкция рамы. Для одного кадра прямое преобразование в основном состоит из разложения кадра на 8 × 8 непересекающихся блоков, выполнения преобразования DCT 8 × 8 каждого блока, квантования блоков, а затем мы выполняем некоторые более сложные вещи, такие как зигзаг. упорядочение, кодирование длин серий и т. д. По сути, кадр представлен в виде сжатой последовательности битов. Реконструкция кадра происходит в обратном порядке. А точнее, восстановление последовательности и отмена зигзагообразного упорядочения, затем деквантование блока и применение IDCT. Причина, по которой они называют это «реконструкцией», заключается в том, что вы представляли кадр в другом формате. Вы конвертируете кадр обратно в то, что он должен был быть перед сжатием кадра.

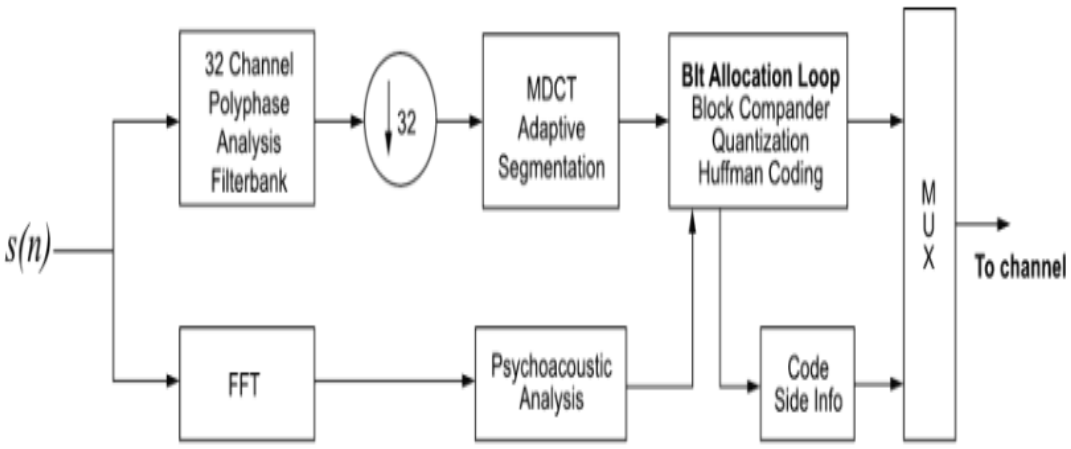


Рисунок 3.3 — Кодер алгоритма сжатия аудио MP3

Банк гибридных фильтров используется для аппроксимации модели диапазона для человеческого слуха. Этот банк фильтров включает в себя адаптивную сегментацию и состоит из фильтров поддиапазонов, за которыми следует Модифицированное дискретное косинусное преобразование (MDCT). Банк фильтров и MDCT выполняют частотно-временной анализ с адаптивным разрешением. Банк фильтров состоит из тридцати двух каналов с фиксированной полосой пропускания, за которыми следует MDCT. MDCT работает с адаптивной длиной кадра, что позволяет проводить спектральный анализ низкого и высокого разрешения. Сложные стратегии распределения битов, которые основаны на неоднородном квантовании и энтропийном кодировании, используются, чтобы уменьшить общую скорость передачи битов и эффективно «упаковать» запись аудиоданных в небольшой файл.

ЗАКЛЮЧЕНИЕ

При написании дипломной работы я изучила популярные алгоритмы сжатия информации для всех форматов данных. И определила для каждого типы данных самый оптимальный алгоритм.

Главными критериями от которых я отталкивалась при анализе, являлись степень сжатия, качество сжатого файла и скорость компрессии. Для формата изображений самым эффективным по степени сжатия был выбран JPEG. Он, в свою очередь, сжимает изображение на 50-70%, и содержит большое количество цветовых оттенков и градиентов. Также высокая скорость кодирования играет большую роль при сравнении с другими методами. Для видео самым оптимальным алгоритмом оказался MPEG-4, будучи самым новым стандартом. Он использует наиболее эффективные алгоритмы сжатия данных. В случаях, когда нужно записывать или просматривать видео, MPEG-4 идеальный выбор. Что касается сжатия аудиоданных, были выбраны алгоритмы MP3 и AAC. Они оба совместимы с большинством проигрывателей, и к тому же очень сложно отличить сжатые аудиофайлы от исходных. Но все же MP3 обеспечивает самое лучшее сжатие. В последнее время этот формат завоевал огромную популярность.

Таким образом, на основании полученных результатов, для каждого формата аудиоданных был выбран самый эффективный алгоритм сжатия.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Фрактальное сжатие изображений спутниковой системы X-SAR европейского космического агентства. А.Ф. Богданов и Н.А. Потемкин, Томский государственный университет систем управления и радиоэлектроники, Томск, 2016.

2 Форматы и алгоритмы сжатия изображений в действии. Дж. Миано. Издательство «Триумф», Москва, 2003.

3 Методы сжатия изображений, аудиосигналов и видео. А.Ю. Тропченко, А.А. Тропченко. Санкт-Петербургский Государственный Университет Информационных Технологий, Механики и Оптики, Санкт-Петербург, 2009.

4 Цифровая связь. Бернард Скляр. Теоретические основы и практическое применение. – М.: Издательский дом «Вильямс», 2003. – 1104 с.

5 Сжатие данных, изображений и звука. Д.Сэломон. «Техносфера», Москва, 2004. - 368с.

6 Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. "Диалог - МИФИ", 2003 — 384 с.

7 Эффективное кодирование видеоинформации в новом стандарте H.264/AVC. - Дворкович А. В. Труды НИИР, 2005.

8 Аналитический обзор алгоритмов сжатия цифровой информации. В. В. Кириченко. Объединенный институт проблем информатики НАН Беларуси, Минск, 2016.