Глава VI Алгоритм фрагментарного сжатия видеопотока

Актуальность задачи сжатия информации трудно переоценить. Алгоритмы сжатия позволяют не только более эффективно использовать дорогие устройства хранения информации, но и существенно снизить нагрузку на каналы передачи [Кириллин Ю.П., сорин В.Я. Повышение эффективности цифровой системы передачи видеосигнала // Телевидение (Межвузовский сборник). 1978. – с. 80-82.]. В Глава V нами уже были рассмотрены классические методы сжатия, а также способы их применения к отдельным изображениям. В данной главе нами будут в общих чертах рассмотрены активно используемые в настоящее время алгоритмы сжатия видео с потерями, а также мы подробно рассмотрим предложенный авторами метод фрагментарного сжатия видео без потерь.

1 Методы сжатия видеопотока с потерями

Популярность сжатия видео с потерями объясняется крайне высокими коэффициентами сжатия (до двухсот и более раз), которых пока невозможно достичь, используя сжатие без потерь [Глассман К.Ф., Логунов А.Н. Оценка заметности артефактов видеокомпрессии // Техника кино и телевидения. – 2003. - №4. – с. 29-32.].

Большинство используемых на практике методов крайне сложны в реализации и работают в несколько этапов:

- Предварительная сглаживающая фильтрация позволяет избавиться от шумов и мелких деталей, тем самым сделав изображение более однородным и пригодным для сжатия;
- Преобразование цветовых пространств позволяет преобразовывать кадры в вид, более удобный для сжатия;
- Огрубление некоторых цветовых каналов как способ сжатия уже упоминалось ранее (см. Глава V3.2);

- Компенсация и предсказание движения – один из важнейших алгоритмов, применяемых для сжатия видеопотоков, который мы рассмотрим ниже;
- Другие методы обработки.

Большинство упомянутых этапов были нами рассмотрены в предыдущих главах, поэтому сейчас необходимо сказать несколько слов о том, как работает алгоритм компенсации и предсказания движения [Беляев Е.А., Тюрликов А.М. Алгоритмы оценки движения в задачах сжатия видеоинформации на низких битовых скоростях // Компьютерная оптика. – т. 32(2008). - №4. – с. 69-76.]. Основная идея метода заключается в поиске на соседних кадрах похожих областей и эффективном их кодировании, то есть представлении в виде соответствующего вектора перемещения и уточняющей информации.

По сути этот алгоритм можно представить в виде следующей последовательности шагов:

1. Разбиение текущего кадра на прямоугольные блоки (большинство алгоритмов использует блоки $16x16$ или $8x8$ пикселов);
2. Для каждого блока выполняется поиск наиболее похожего блока с предыдущего кадра в определённой окрестности;
3. Вычисление и сильное огрубление уточняющей информации (по сути разности интенсивностей между пикселами на текущем и найденном на предыдущем кадре блоками);
4. Кодирование блока в виде вектора перемещения и уточняющей информации. Уточняющая информация может эффективно кодироваться энтропийными алгоритмами, так как значения разностей интенсивностей в большинстве случаев будут близки к нулю.

Реализация этого алгоритма не должна вызывать трудностей за исключением второго шага. Что именно считать достаточно похожими блоками – хороший, но довольно сложный вопрос [Беляев Е.А., Чуйков А.В. Модифицированный алгоритм иерархической оценки движения в задачах сжатия видеоинформации: Научная сессия ГУАП, 2007. Сб. докл. / ГУАП. СПб. – с. 56-60.].

От выбора способа измерения близости сильно зависит не только скорость работы алгоритма, но и его эффективность (коэффициент сжатия). Наиболее распространены следующие подходы оценки похожести фрагментов: среднее квадратичное отклонение, сумма модулей разности, а также поиск и сравнение характерных точек блоков.

Использование методов компенсации движения приводит к следующей структуре видеопотока (Рис. 64) [Ричардсон Я. Видеокодирование. H/264 и MPEG-4 – стандарты нового поколения. –М.: Техносфера, 2005, - 366 с.]:

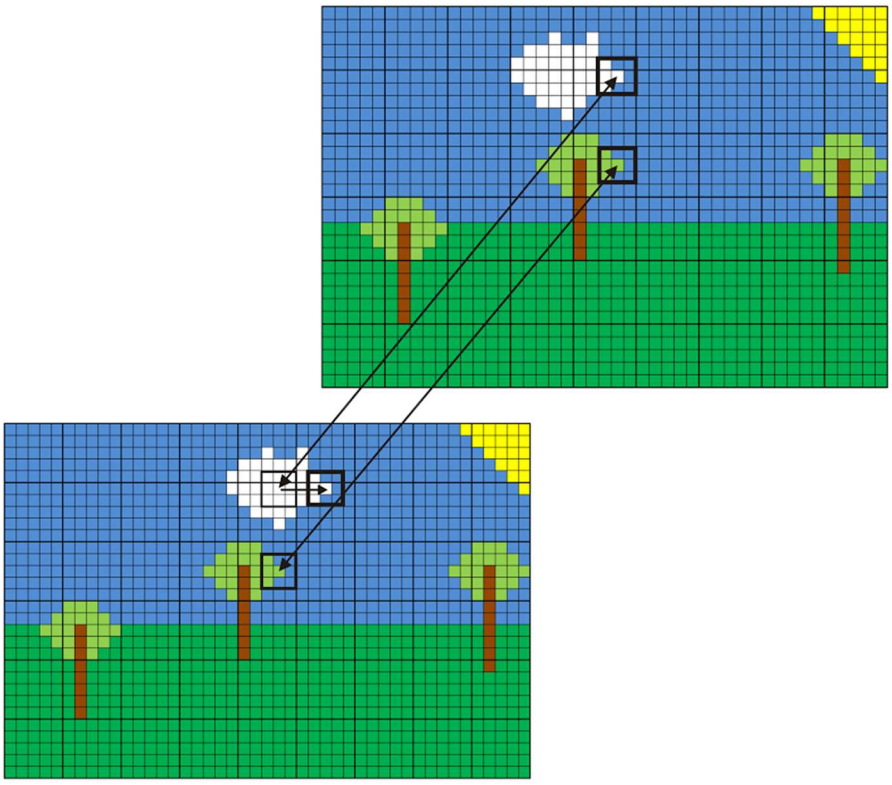


Рис. 64. Кодирование видеопотока с помощью компенсации движения

2 Фрагментарный метод сжатия видеопотока

Данный раздел посвящён подробному описанию предложенного авторами метода фрагментарного сжатия. При этом описании мы будем пользоваться следующими понятиями и определениями:

- *Пиксель* – минимальная единица изображения. Числовое значение пикселя выражает значение функции яркости изображения в одной точке экрана.

Количество бит, отводимое для кодирования яркости, называется глубиной цвета и обозначается в дальнейшем bpp;

- *Кадр* – набор всех пикселей в конкретный момент времени. Кадр представляется как двумерный массив пикселей высотой N_1 и шириной N_2 пикселей.
- *Видеопоток (фильм)* – последовательность кадров, упорядоченная по времени. Общее число кадров в фильме в дальнейшем обозначается M.
- *Окно*– прямоугольная область пикселей высотой n_1 и шириной n_2.
- *Фрагмент* – часть кадра, ограниченная окном.
- *Логическая разность* – результат применения операции сложения по модулю 2 (исключающего ИЛИ) к двум цифровым представлениям соответствующих фрагментов в соседних кадрах.
- *Арифметическая разность* – результат арифметического вычитания цифровых представлений соответствующих фрагментов в соседних кадрах.
- *Элемент* – в качестве элемента в зависимости от особенностей задачи выступает либо фрагмент, либо различные виды разностей. Цифровым представлением элемента является битовая строка длиной k. В случае фрагментов или логических разностей $k = n_1 * n_2 * bpp$, а в случае арифметических разностей $k = n_1 * n_2 * bpp + n_1 * n_2$, так как на каждый пиксел окна требуется дополнительный знаковый бит.
- *Объём фильма* (N_ϕ) – общее количество элементов в фильме. $N_\phi = \frac{N_1 * N_2 * M}{n_1 * n_2}$
- *Частота элемента* – отношение количества появлений данного элемента в фильме к объёму фильма.
- *База элементов* – набор всех присутствующих в видеопотоке элементов и их частот. Мощность базы элементов обозначается N_6.
- *Код элемента*– двоичный код, позволяющий однозначно идентифицировать элемент в базе.

2.1 Основная идея фрагментарного метода сжатия

Основная идея фрагментарного метода сжатия заключается в представлении видеопотока в виде цепочки элементов длиной $N_ф$ из базы элементов. Поскольку видеопоток представляет собой набор осмысленных изображений (кадров), достаточно медленно меняющихся во времени, следует ожидать существенной корреляции как между соседними элементами одного кадра, так и между соответствующими элементами на соседних кадрах. Это должно приводить к двум эффектам:

- Сравнительно небольшой мощности базы ($N_б$) относительно мощности множества всех возможных значений фрагментов (2^{k_f}) даже при достаточно большом объёме фильма;
- Значительной неравномерности частот появлений различных элементов из базы элементов в фильме. Это должно позволить применять эффективные энтропийные алгоритмы кодирования-сжатия цепочки элементов в видеопотоке.

Таким образом основная идея метода фрагментарного сжатия состоит в представлении видеопотока в виде хорошо сжимаемой последовательности элементов из некоторой специально составленной базы элементов. Проведённые авторами эксперименты подтверждают эту гипотезу.

Концепция метода позволяет осуществлять сжатие видеопотока как без потерь, так и с потерями. В случае сжатия с потерями возможна и предобработка (фильтрация), позволяющая сгладить исходное изображение, и постобработка, заключающаяся в анализе базы элементов и последующем выделении и удалении из видеопотока как случайных помех, так и визуально избыточной информации. Совместное использование пред- и постобработки позволяет формировать новый сжатый видеопоток с нужными свойствами.

Метод фрагментарного сжатия можно представить в виде последовательности четырёх шагов:

- Формирование базы элементов. В случае сжатия с потерями возможна предварительная фильтрация кадров видеопотока;
- Анализ полученной базы элементов и её частотных характеристик;
- Построение коротких кодов для элементов базы;
- Формирование схемы сжатой передачи фильма.

Ниже эти шаги рассмотрены более подробно.

2.1.1 Формирование базы элементов

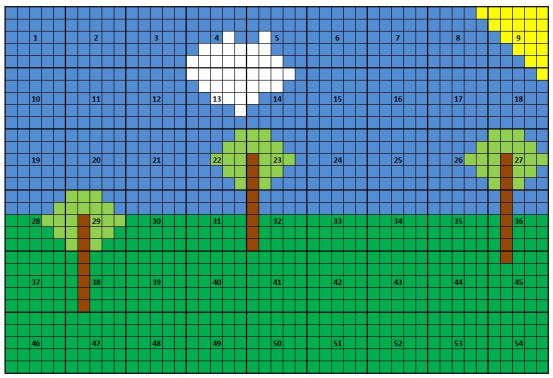
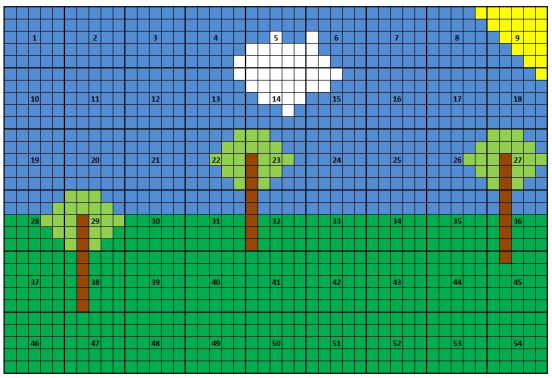
Процесс формирования базы элементов – один из самых ресурсоёмких шагов фрагментарного метода сжатия. Итоговая скорость компрессии во многом зависит от того, насколько эффективно выполняется этот этап.

Ввиду крайне высоких требований к скорости, алгоритм формирования базы элементов выполняется в несколько этапов, при этом на каждом шаге используются различные алгоритмы и структуры данных.

Формирование базы элементов кадра – первый и самый простой шаг. На этом этапе происходит получение очередного кадра из видеопотока, его предварительная обработка в случае необходимости, и, в зависимости от вида элемента, формируется соответствующий «псевдокадр»[41].

[41] При построении базы фрагментов «псевдокадр» совпадает с очередным кадром видеопотока, при построении базы логических или арифметических разностей «псевдокадр» является результатом применения соответствующей операции к двум последовательным кадрам.

Табл. 90. Два последовательных кадра

Кадр i	**Кадр $i+1$**
	

При просмотре псевдокадра» окно сдвигается на n_2 пикселов по ширине и на n_1 пикселов по высоте. В Табл. 90 приведён пример кадра $N_1 = 30$ строк на $N_2 = 45$ столбцов, просмотр выполняется с помощью квадратного окна $n_1 = n_2 = 5$. Тонкими линиями на рисунке выделены отдельные пикселы, толстыми линиями показаны границы окна, а цифры внутри окна – порядковый номер при просмотре.

Если в качестве элементов выступают логические или арифметические разности, то в результате сканирования получится следующий набор разностей:

Табл. 91. База элементов кадра

Номер окна	**Разность**	**Количество**
1	0	1
2	0	1
3	0	1
4	P4	1
5	P5	1
6	P6	1
7	0	1
...	0	1
13	P13	1
14	P14	1
15	P15	1
16	0	1
...	0	1
54	0	1

Для большинства реальных видеофильмов выполняется предположение о существенной корреляции между соответствующими элементами на соседних

кадрах. Это видно и в Табл. 91: большинство элементов соответствует нулевым разностям. Для экономии памяти полученная база элементов кадра уплотняется. Сначала выполняется сортировка каким-нибудь эффективным алгоритмом, а затем совпадающие элементы объединяются в одну запись. Т.е. вместо 48 нулевых разностей в уплотнённой базе будет только одна запись. Пример уплотнённой базы приведён в Табл. 92

Табл. 92. Уплотнённая база элементов кадра

Разность	Количество
0	48
P4	1
P5	1
P6	1
P13	1
P14	1
P15	1

На втором этапе уплотнённые базы элементов кадра записываются в специальный буфер элементов – отдельную область памяти, где множество баз кадров накапливаются, а потом одним большим блоком записываются в глобальное хранилище. Особенностью буфера является то, что к любому его элементу возможен произвольный доступ по индексу, т.е. по сути это просто большой массив элементов, размещённый в ОЗУ:

Также стоит отметить, что базы элементов кадра записываются в буфер «как есть» без всякой дополнительной пересортировки. Это приводит к тому, что буфер содержит в себе множество упорядоченных последовательностей элементов, но при этом сам не упорядочен:

Рис. 65. Примерная структура частичной базы

Когда для записи очередной базы кадра в буфере недостаточно места, вызывается процедура обновления списка реально встретившихся элементов.

На третьем этапе накопленные в буфере элементы перемещаются в специальное глобальное хранилище, которое представляет собой односвязный список элементов, упорядоченный по их значениям. Представление глобального хранилища в виде списка не позволяет выполнять произвольный доступ за константное время, но зато позволяет выполнять добавление и удаление элементов за константное время.

Перед тем, как добавить элементы из буфера в глобальное хранилище, выполняется уже рассмотренная ранее процедура сортировки и уплотнения буфера. Это позволяет выполнить добавление элементов за линейное относительно размера списка время:

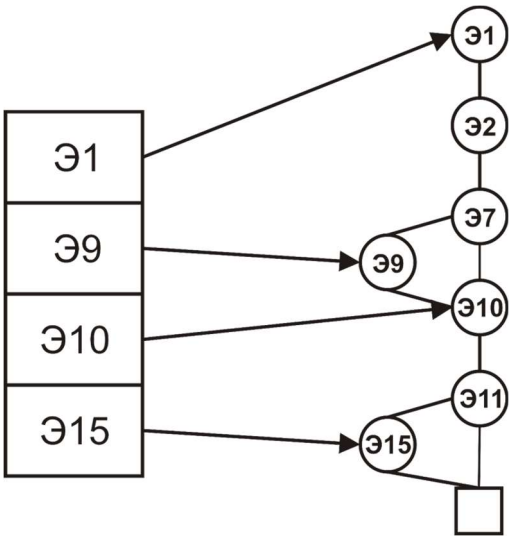


Рис. 66. Добавление элементов из буфера в глобальный список

Три рассмотренных этапа оперируют данными, размещёнными в ОЗУ. Заключительный этап предполагает работу с данными, размещёнными на жёстких дисках. В подавляющем большинстве случаев достаточно мощный современный компьютер позволяет полностью собрать базу элементов в ОЗУ, но существуют ситуации, когда размера оперативной памяти недостаточно, чтобы разместить там всю базу (например, если выполняется формирование общей базы для сотен часов съёмки). В этом случае необходимо сохранять полученные базы на диск по мере исчерпания оперативной памяти, а потом выполнить процедуру слияния баз.

Таким образом в процессе формирования базы элементов выполняется следующая последовательность шагов:

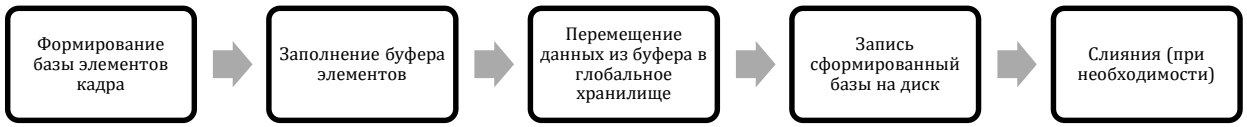


Рис. 67. Процесс формирования базы элементов

2.1.2 Анализ полученной базы элементов и её частотных характеристик

Анализ базы элементов предполагает анализ как индивидуальных значений каждого элемента, так и частотных характеристик базы. Это позволяет, во-первых, выбирать эффективные комбинации методов сжатия (построения коротких кодов и способов их передачи), а, во-вторых, в случае сжатия с потерями, осознанно проводить сокращение базы (фильтрацию).

2.1.3 Построение коротких кодов для элементов базы

База элементов включает в себя массив из $N_б$ строк длины k бит и таблицу частот появления этих строк в видеопотоке (фильме). Этой информации достаточно для применения хорошо известных методов энтропийного кодирования (дерево Хаффмана, арифметическое кодирование и т.д.). Перечисленные алгоритмы имеют очень высокие оценки эффективности, но их реальное использование связано с существенными трудностями реализации.

Для построения коротких кодов элементов авторы предлагают и используют предложенный ими ранее метод построения префиксных кодов с помощью секущих функций. В этом методе короткие коды элементов базы строятся по содержанию строк самих элементов. В работе Огнева А.И. доказано, что средняя длина таких кодов не превышает величины $H_б * 1{,}049$, где $H_б$ – энтропия базы элементов, вычисляемая по частотам появления элементов в видеопотоке. В численных экспериментах с реальными фильмами $H_б$ никогда не превышала 20, а это значит, что $H_б * 1{,}049 < H_б + 1$, откуда следует, что средняя длина кодов секущих укладывается в те же априорные оценки избыточности, что коды Шеннона-Фано и Хаффмана. При этом кодирование с помощью секущих функций лишено тех недостатков, которые возникают при кодировании и декодировании

вышеназванными методами цифровых массивов больших размеров ($N_б \approx 10^7 - 10^9$).

2.1.4 Формирование схемы сжатой передачи фильма

В сжатом видеопотоке элементы заменяются полученными ранее короткими кодами. При воспроизведении сжатого видеопотока возникает проблема декодирования. Для декодирования необходимо иметь информацию о соответствии коротких кодов и элементов базы. Самым простым способом передачи такой информации является непосредственная передача элемента и его частоты (количества появлений). Такой способ подразумевает передачу самой базы, а также дополнительных 64 бит на каждый элемент. Принимающая сторона после получения данных сама должна построить кодовое дерево по заранее сообщённому алгоритму.

Авторами был предложен способ записи и передачи дерева, в листьях которого записана вся база элементов. При этом объём передачи равен объёму базы плюс два дополнительных бита на каждый элемент базы ($(k + 2) * N_б$).

Договоримся о порядке обхода дерева: например, будем использовать обход в ширину. Если встречается делящий узел дерева, то передаётся бит 0, если в процессе обхода был обнаружен лист, то передаётся бит 1 и соответствующий элемент базы. В итоге легко показать, что на каждый элемент базы дополнительно передаётся всего 2 лишних бита. На Рис. 68 показано представление двоичного дерева в виде двоичной строки с помощью описанного метода:

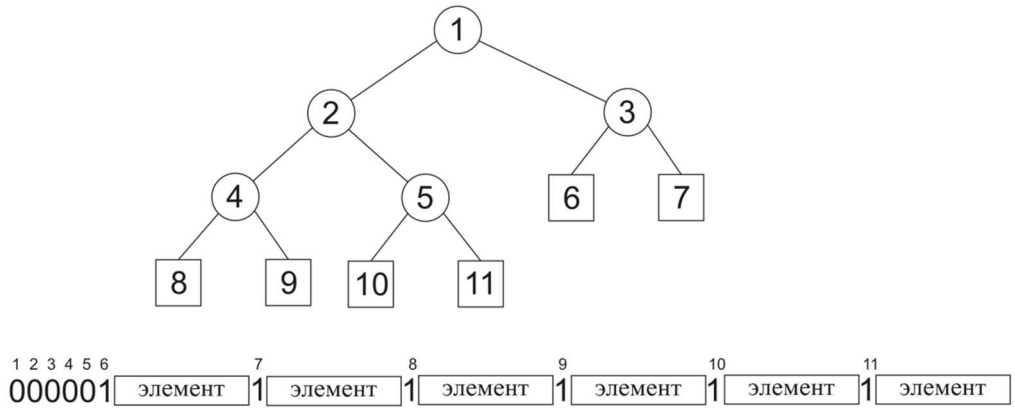


Рис. 68. Эффективная передача кодового дерева

Указанное представление обратимо: для каждой строки, построенной описанным способом, можно построить исходное кодовое дерево. Помимо очевидной экономии памяти описанный способ передачи избавляет от необходимости строить кодовое дерево при декодировании (оно восстанавливается естественным образом). То есть кодовое дерево строится только один раз во время сжатия.

Таким образом в работе предлагается схема передачи видеопотока, состоящая из двух частей:

- Кодовое дерево с элементами базы;
- Цепочка сжатых кодов элементов.

Примерная структура сжатого видеопотока приведена на Рис. 69:

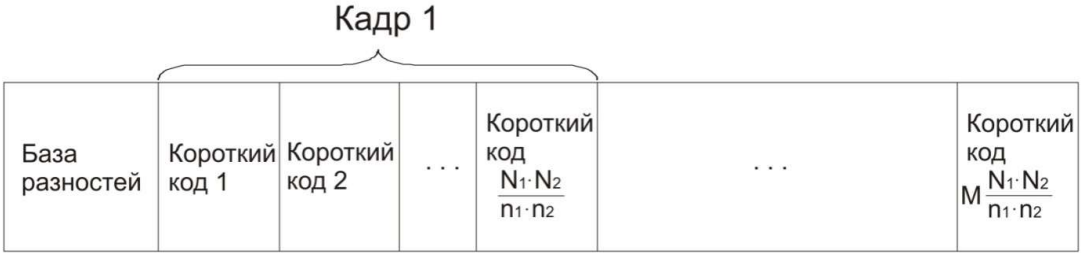


Рис. 69. Структура сжатого видеопотока

2.2 Выбор параметров алгоритма

Степень сжатия, обеспечиваемая алгоритмом фрагментарного сжатия, во многом зависит от свойств и характеристик сформированной базы элементов.

Основными характеристиками базы являются размер (количество элементов) и частотные характеристики (в первую очередь энтропия).

Для оценки достижимой степени сжатия воспользуемся следующими формулами. Пусть $l_{\text{ср}}$ – средняя длина кодов элементов базы, тогда объём сжатой передачи равен $V_{\text{сж}} = N_{\text{б}} * (k+2) + l_{\text{ср}} * N_{\text{ф}}$, а объём несжатого фильма при этом равен $V_{\text{ф}} = N_{\text{ф}} * \text{k}$.

Степенью сжатия назовём величину:

$$R_{\text{сж}} = \frac{V_{\text{сж}}}{V_{\text{ф}}} = \frac{N_{\text{б}}}{N_{\text{ф}}} * \frac{k+2}{k} + \frac{l_{\text{ср}}}{k}$$

где $N_{\text{ф}} = \frac{N_1 * N_2 * M}{n_1 * n_2}$, а $k = n_1 * n_2 * bpp$. Обратная степени сжатия величина называется коэффициентом сжатия и обозначается $K_{\text{сж}}$.

Основная цель, которой надо стремиться, - максимизация $K_{\text{сж}}$. В формуле, приведённой выше, N_1, N_2, M, bpp – характеристики исходного видеопотока, на которые мы влиять не можем. А величины $l_{\text{ср}}, N_{\text{базы}}$ зависят от площади и конфигурации окна сканирования (n_1, n_2). Т.е. единственными вариабельными параметрами в методе фрагментарного сжатия являются геометрические размеры окна.

Для любого размера окна существует максимально возможный объём базы, равный $2^{bpp * n_1 * n_2}$, и эта величина растёт чрезвычайно быстро. Конечно, в реальных фильмах встречаются далеко не все возможные элементы, но с увеличением размера окна растёт и количество бит, необходимое для представления соответствующего элемента базы. Кроме того, появляется существенное количество (более 70% от общего числа) элементов, которые встречаются в видеопотоке не более одного раза. В конечном итоге все перечисленные эффекты приводят к уменьшению коэффициента сжатия.

С другой стороны, слишком маленький размер окна приведёт к росту $N_{\text{ф}}$ и ограничению $N_{\text{б}}$, что в свою очередь приведёт к тому, что элементы будут повторяться более равномерно и "перекос" частот сгладится. Чем равномернее

повторяются элементы базе, тем ближе её энтропия к $\log_2(N_б)$. Это в свою очередь тоже приводит к уменьшению коэффициента сжатия.

Рассмотрим два крайних случая окно $1 * 1$ и окно $N_1 * N_2$. И в первом, и во втором случае $K_{сж} \approx 1$. В процессе исследований было установлено, что зависимость $K_{сж}$ от площади окна сканирования ($n_1 * n_2$) имеет примерно следующий вид:

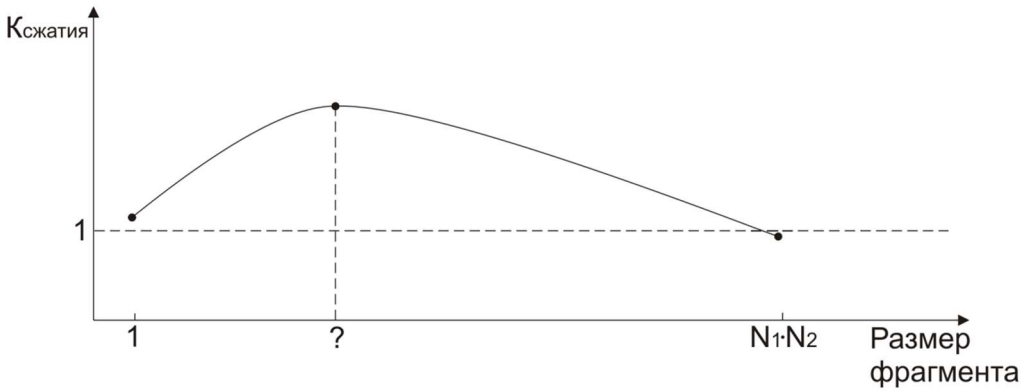


Рис. 70. Зависимость коэффициента сжатия от площади окна сканирования

Задача заключается в том, чтобы найти точку, где уровень сжатия максимальный. Т.к. уровень сжатия во многом зависит и от характеристик фильма, понятно, что не существует "идеальной" конфигурации, которая будет давать оптимальное сжатие для всех фильмов. Но эксперименты позволили найти сравнительно небольшую область, где сжатие оптимально.

Все анализируемые видеопотоки были разделены на две группы в зависимости от их особенностей:

- Естественные съёмки. К этой группе относится большинство фильмов, телепередач и любительских съёмок. Для фильмов этой группы характерно плавное изменение кадров с течением времени. При этом изменения достаточно плавные и сконцентрированы в центре кадра. Фон в большинстве случаев однородный, но возможны небольшие колебания яркости.
- Искусственные видеопотоки. Отличительной особенностью видеопотоков, отнесённых ко второй группе, является неестественность их происхождения.

Ярким примером являются мультфильмы. Несмотря на то, что фон может содержать множество мелких деталей, этот тип видеопотоков лучше других поддаётся сжатию фрагментарным методом ввиду отсутствия шума сенсоров и помех, связанных с дрожанием камеры.

Для выбора оптимальной конфигурации окна видеопотоки, принадлежащие каждой группе, были сжаты с использованием фрагментарного метода сжатия. В ходе исследований были проанализированы все возможные конфигурации окон площадью от одного до восьми пикселей (в качестве элементов использовались логические разности). В сумме было проанализировано около десяти тысяч часов видео, что составляет около миллиарда кадров. Полученные результаты представлены на Рис. 71 и Рис. 72.

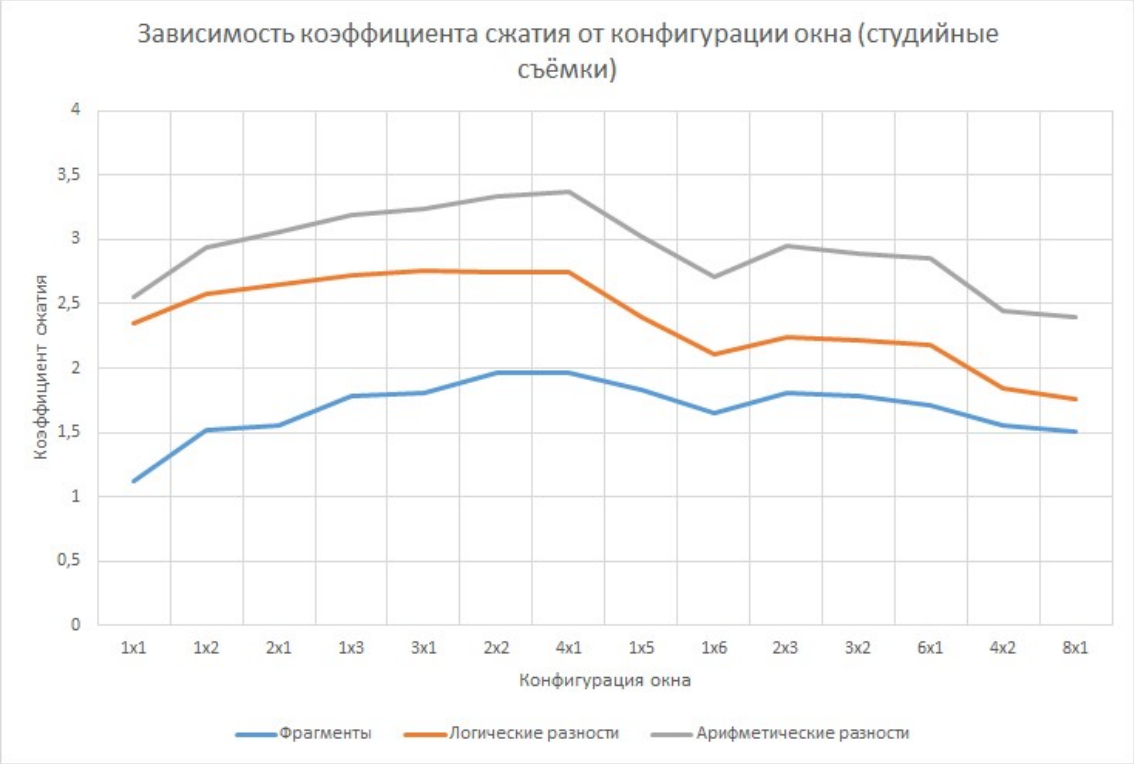


Рис. 71. Коэффициент сжатия (естественные съёмки)

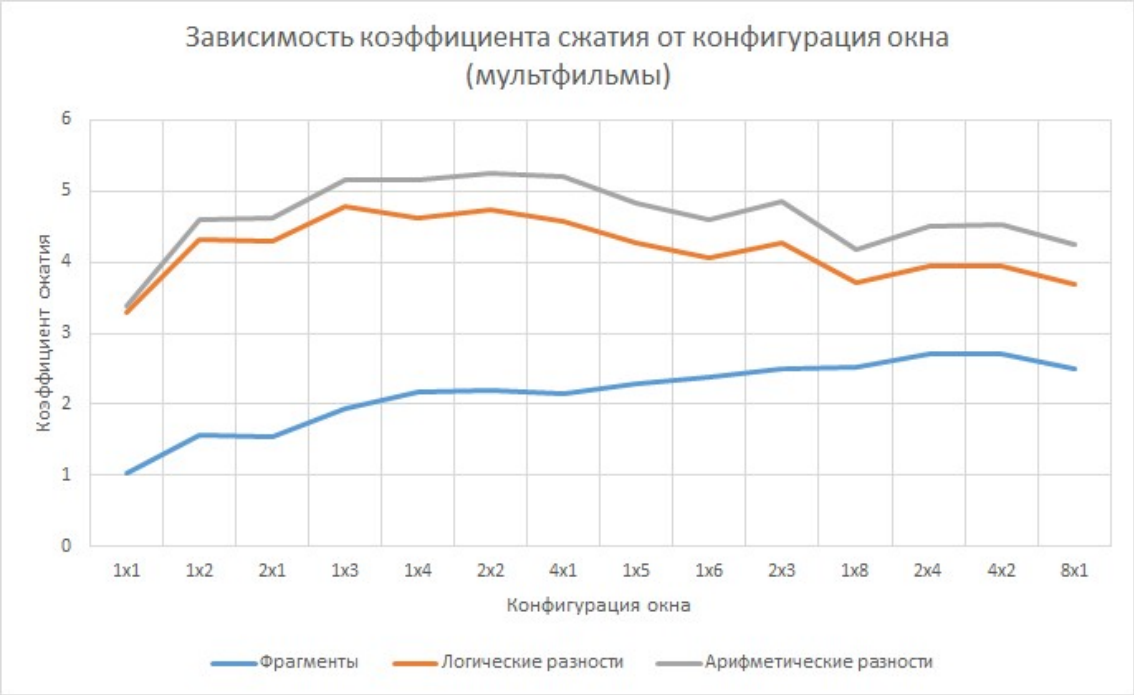


Рис. 72. Коэффициент сжатия (мультфильмы)

Из представленных данных очевидно, что максимального коэффициента сжатия можно достичь при использовании окон площадью в три, четыре или пять пикселей. Также при практическом использовании метода фрагментарного сжатия

видеопотока надо учитывать, что при фиксированной площади окна наибольшая эффективность сжатия наблюдается при использовании прямоугольного окна с соотношением сторон близким $\frac{2}{1}$.

2.2.1 Кодирование видеопотока в различных цветовых пространствах

Рассмотренные ранее результаты относились к ситуации кодирования видеопотока, содержащего только яркостную компоненту (так называемого монохромного видеопотока). Больший практический интерес представляют методы, позволяющие кодировать цветные видеопотоки.

Ранее были рассмотрены способы кодирования информации о цвете (см. Глава I3). Самым простым способом кодирования цветного видеопотока является независимое кодирование цветовых каналов. Был также рассмотрен способ, при котором базы элементов, собранные для независимых цветовых каналов, объединялись в одну общую базу. Несмотря на то, что доля базы в передаче при этом уменьшалась, рост эффективности сжатия был незначительным [M39]. Также было показано, что эффективность метода фрагментарного сжатия в пространстве YIQ существенно выше, чем в пространстве RGB (см.Рис. 73, Рис. 74). В первую очередь этот эффект связан с потерями, возникающими при преобразовании цветовых пространств.

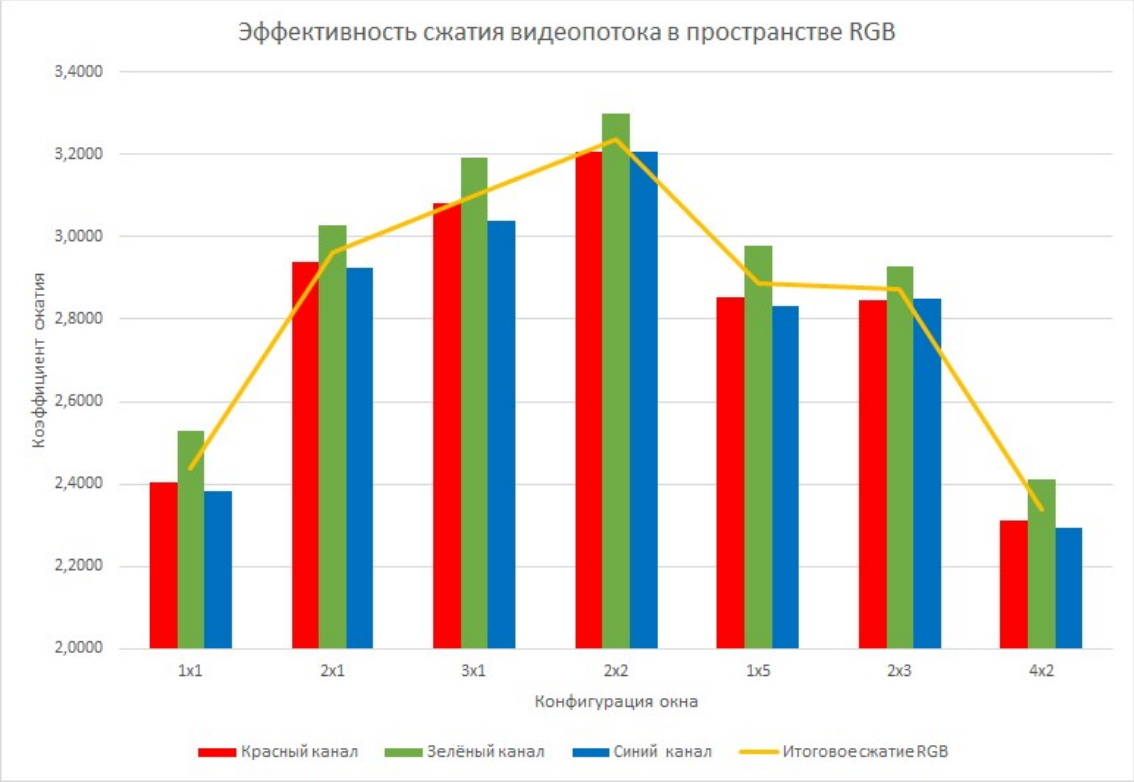


Рис. 73. Эффективность сжатия видеопотока в цветовом пространстве RGB

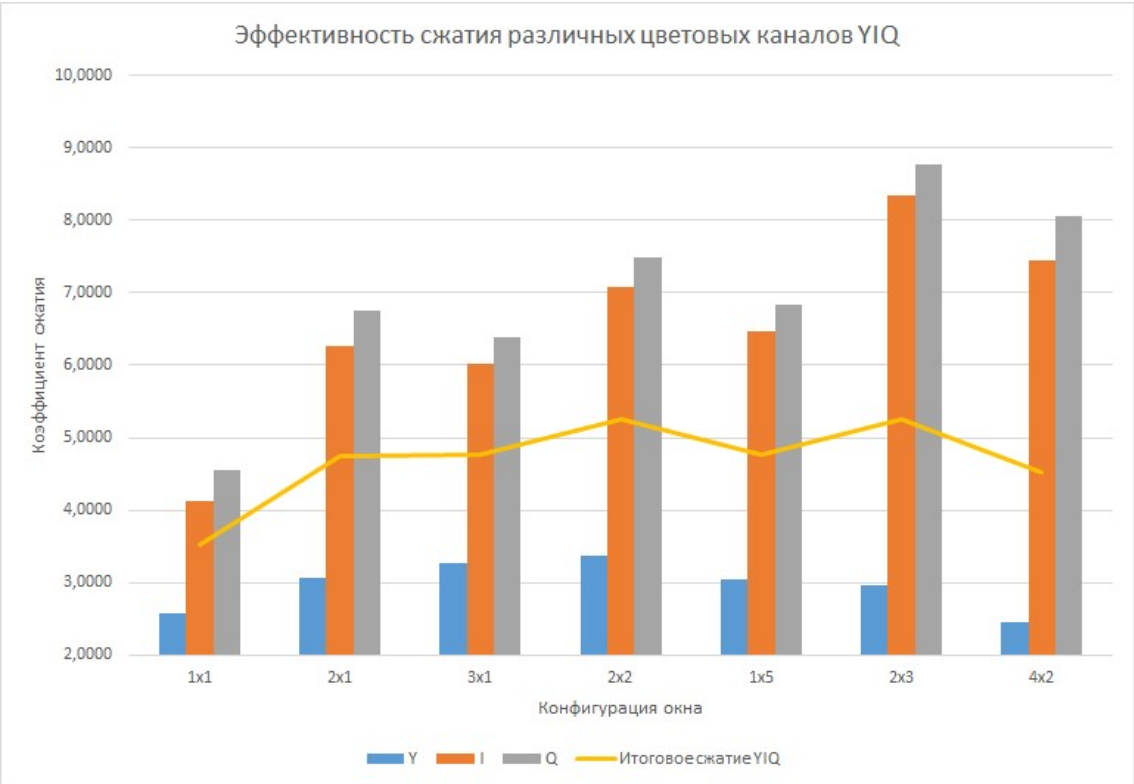


Рис. 74. Эффективность сжатия видеопотока в цветовом пространстве YIQ

Стоит также отметить, что эффективность сжатия в пространстве YIQ может быть существенно повышена за счёт грубого квантования цветоразностных каналов.

2.3 Сравнение эффективности метода фрагментарного сжатия видеопотока с используемыми методами

На сегодняшний день существуют методы, позволяющие сжимать видеопотоки без потерь. Некоторые из них узкоспециализированы и эффективны только на одном типе видеопотоков (например, на записи экрана), другие методы оказываются эффективны на различных видеопотоках. Данный раздел посвящён сравнению эффективности предложенного метода фрагментарного сжатия видеопотока с известными методами.

В Graphics & Media Lab Video Group при МГУ им. Ломоносова было выполнено обширное сравнение алгоритмов сжатия видео без потерь по множеству параметров (скорости компрессии, расходу ресурсов, эффективности сжатия и т.д.). Наиболее важной характеристикой является эффективность сжатия. По этому критерию были проанализированы следующие кодеки:

- Alpary;
- ArithYuv;
- AVIzlib;
- CamStudio GZIP;
- CorePNG;
- FastCodec;
- FFV1;
- Huffyuv;
- Lagarith;
- LOCO;
- LZO;
- MSU Lab;
- PICVideo;
- Snow;
- x264;
- YULS;

Сравнение выполнялось на стандартных тестовых последовательностях:

Табл. 93. Используемые при сравнении видеопоследовательности

Тестовая последовательность	Количество кадров	Разрешение
Foreman	300	352x288
Susi	374	704x576
Tennis	373	704x576
BBC	374	704x576
Battle	1599	704288

News	32	720x480
Da	262	720x352
Mi	261	640x272
Bankomat	376	704x352

Каждая видеопоследовательность была сжата каждым кодеком независимо. Полученные результаты приведены на Рис. 75:

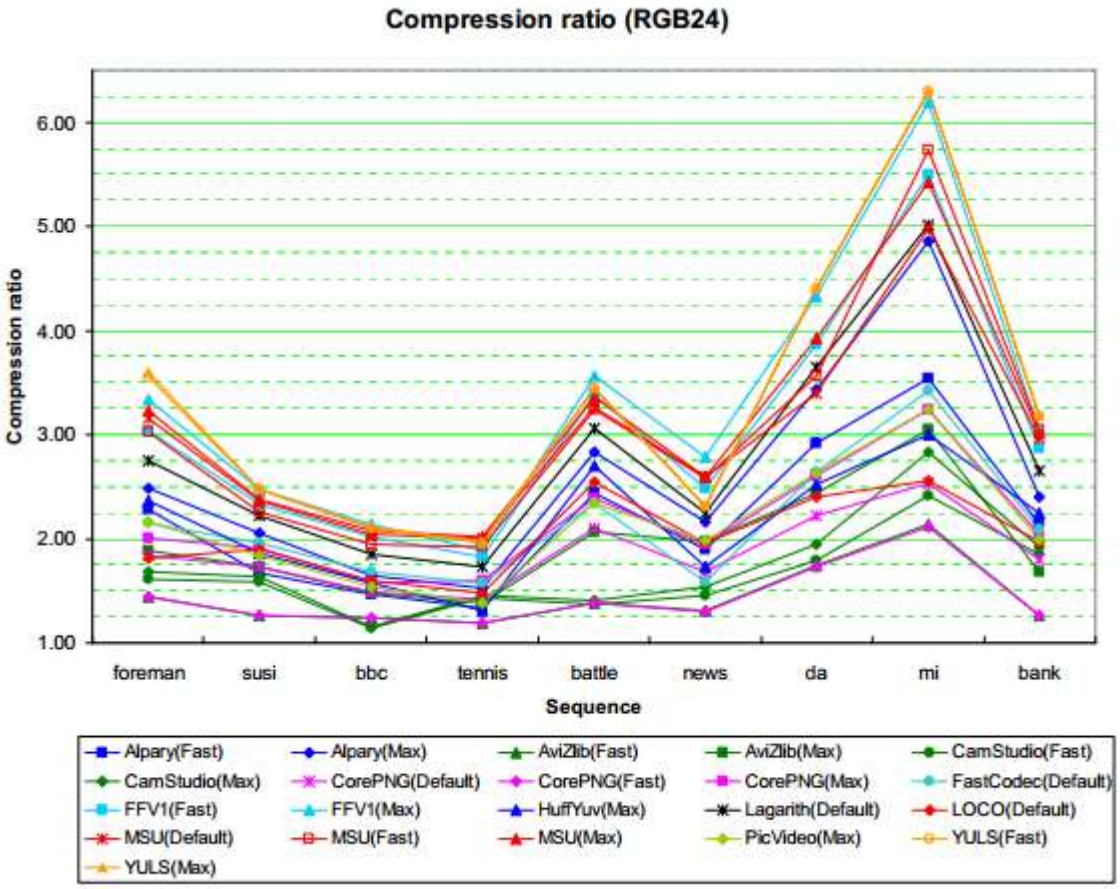


Рис. 75. Эффективность сжатия рассматриваемых в сравнении алгоритмов

Для удобства анализа полученные значения коэффициентов сжатия были усреднены, и полученные средние оценки приведены на Рис. 76:

Рис. 76. Усреднённые коэффициенты сжатия исследуемых методов

Прямое применение метода фрагментарного сжатия на этих тестовых видеопоследовательностях оказывается малоэффективным. Это объясняется тем, что длительность тестовых видеопотоков очень мала и существенную часть в сжатом методом фрагментарного сжатия фильме составляет база элементов (см. Рис. 77).

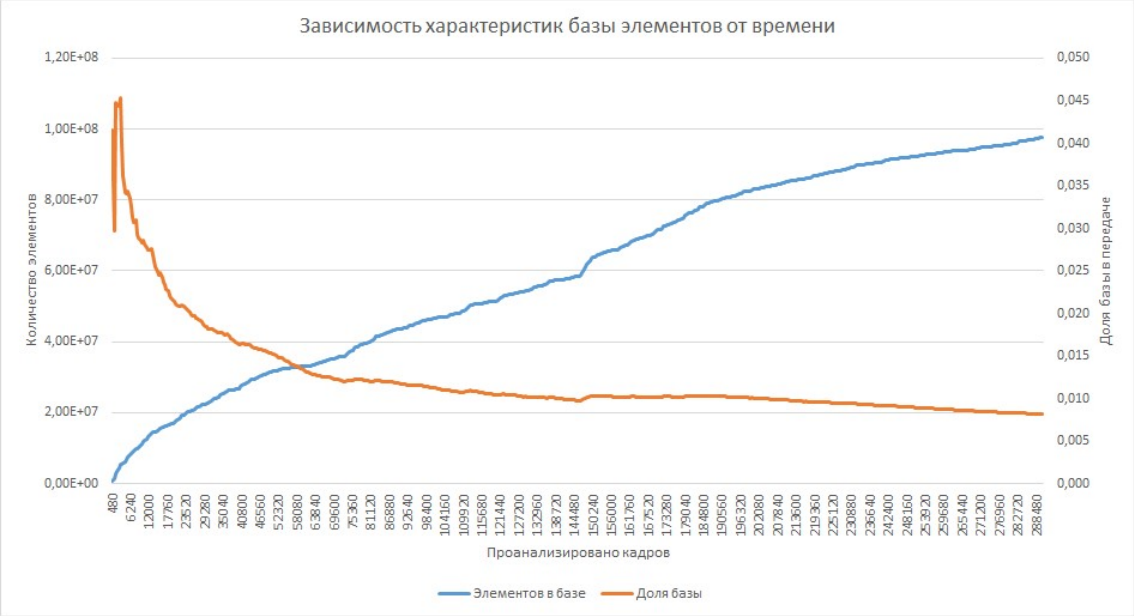


Рис. 77. Зависимость характеристик базы элементов от времени

При достаточной длительности видеопотока (от 5000 кадров) средний коэффициент сжатия метода фрагментарного сжатия составляет 3,38, что сравнимо (и даже немного превышает) с коэффициентами сжатия лучших из рассмотренных методов.

2.4 Способы повышения эффективности сжатия фрагментарного метода без потерь

2.4.1 Метод фрагментарного сжатия битовых плоскостей видеопотока

При увеличении площади окна и фиксированной глубине цвета отношение $\frac{N_б}{N_ф}$ растёт так быстро, что доля кодового дерева в передаче становится доминирующей и ведёт к ухудшению степени сжатия независимо от средней длины получаемых кодов. Обойти эту проблему можно двумя способами:

1. Исследования показали, что при фиксированном размере окна увеличение продолжительности фильма (M) приводит, во-первых, к уменьшению отношения $\frac{N_б}{N_ф}$ (за счёт того, что $N_б$ растёт гораздо медленнее $N_ф$), во-вторых, к уменьшению $H_б$ (что априори совсем не очевидно, так как сама база растёт). Отсюда следует, что гораздо выгоднее собирать базу элементов и кодировать её

сразу для видеопотоков большой длительности, например, вместо кодирования записей видеонаблюдения по дням гораздо эффективнее закодировать запись за всю неделю или месяц. Кроме того, не исключена возможность создания некой универсальной базы элементов, общей для всех существующих фильмов с фиксированной глубиной цвета bpp. Эта теоретическая база может иметь размеры, позволяющие работать с ней на обычных персональных ЭВМ (особенно при небольших значениях bpp), а соответствующее ей кодовое дерево может быть заранее известно всем пользователям, что позволит ограничится только передачей последовательности коротких кодов элементов.

2. Второй способ связан с разбиением исходного видеопотока на bpp двоичных видеопотоков (битовых плоскостей). Метод разложения на битовые плоскости заключается в разделении одного изображения с 2^{bpp} уровнями яркости на bpp бинарных изображений. При этом i-ое изображение формируется путём выделения i-ых битов из каждого пикселя исходного изображения. Если применить такое разложение ко всем кадрам видеопотока, то будет сформировано bpp бинарных видеопотоков с глубиной яркости в один бит на пиксель, то есть каждый пиксель которых имеет всего два возможных значения яркости.

Метод фрагментарного сжатия битовых плоскостей видеопотока состоит в том, что каждый из bpp сформированных бинарных видеопотоков сжимается по отдельности. При этом для каждого бинарного видеопотока могут быть выбраны свои оптимальные параметры метода. Также при этом легко вычисляется степень сжатия исходного видеопотока как среднее арифметическое степеней сжатия всех b_p видеопотоков, так как объёмы бинарных видеопотоков одинаковы.

Большим преимуществом этого метода является то, что достаточно вычислить базы элементов для каждого бинарного видеопотока и нет необходимости вычислять общую базу элементов исходного фильма. Это позволяет резко увеличить размеры

окна и исследовать степени сжатия видеопотока при ранее недостижимых размерах окон.

Была проведена серия экспериментов [м40], в которых видеопотоки, полученные путём выделения битовых плоскостей, кодировались фрагментарным методом сжатия. Результаты экспериментов показывают, что фрагментарное сжатие битовых потоков позволяет переходить к окнам большого размера. Ниже приведена Табл. 94, показывающая степень сжатия без потерь для каждой из восьми битовых плоскостей для различных окон (в качестве элемента выступают фрагменты). Для каждого окна также подсчитана средняя степень сжатия, то есть степень сжатия исходного видеопотока.

Табл. 94. Зависимость степени сжатия от конфигурации окна и битовой плоскости (фрагменты)

Окно	Битовая плоскость								Сжатие	
	1	2	3	4	5	6	7	8	CR	CC
1x1	1	1	1	1	1	1	1	1	1	1
1x2	0,984	0,97	0,931	0,864	0,775	0,685	0,606	0,522	0,792	1,263
2x1	0,97	0,952	0,91	0,843	0,757	0,672	0,599	0,521	0,778	1,285
1x3	0,97	0,949	0,895	0,807	0,692	0,575	0,472	0,362	0,715	1,398
3x1	0,966	0,943	0,884	0,792	0,678	0,565	0,466	0,361	0,707	1,415
2x2	0,946	0,917	0,843	0,735	0,605	0,481	0,378	0,278	0,648	1,543
4x1	0,952	0,924	0,855	0,752	0,627	0,505	0,396	0,282	0,661	1,512
1x5	0,958	0,93	0,86	0,753	0,62	0,485	0,364	0,235	0,651	1,537
1x6	0,955	0,925	0,851	0,739	0,602	0,462	0,337	0,203	0,634	1,577
2x3	0,925	0,89	0,805	0,685	0,545	0,413	0,303	0,197	0,595	1,68
3x2	0,937	0,901	0,813	0,689	0,546	0,413	0,302	0,197	0,6	1,667
6x1	0,944	0,912	0,832	0,717	0,581	0,447	0,327	0,202	0,62	1,612
4x2	0,917	0,878	0,783	0,654	0,508	0,374	0,262	0,157	0,567	1,765
8x1	0,928	0,891	0,805	0,687	0,548	0,413	0,29	0,162	0,591	1,693
3x3	0,912	0,871	0,774	0,642	0,494	0,357	0,246	0,142	0,555	1,803
1x10	0,941	0,906	0,822	0,702	0,559	0,413	0,281	0,14	0,596	1,679
2x5	0,905	0,865	0,77	0,64	0,494	0,356	0,241	0,133	0,55	1,817
2x6	0,9	0,859	0,761	0,629	0,481	0,342	0,226	0,117	0,539	1,855
4x3	0,89	0,847	0,744	0,609	0,461	0,325	0,215	0,115	0,526	1,903
6x2	0,906	0,863	0,761	0,625	0,475	0,337	0,223	0,116	0,538	1,858
12x1	0,925	0,885	0,793	0,668	0,524	0,383	0,256	0,123	0,57	1,756
1x15	0,931	0,892	0,804	0,68	0,535	0,387	0,252	0,109	0,574	1,744
3x5	0,89	0,845	0,74	0,601	0,449	0,311	0,2	0,099	0,517	1,936
4x5	0,869	0,822	0,712	0,572	0,421	0,286	0,178	0,082	0,493	2,03

Из Табл. 94 были выбраны оптимальные конфигурации окна для каждого значения плоскости и на их основе был построен график, показанный на Рис. 78:

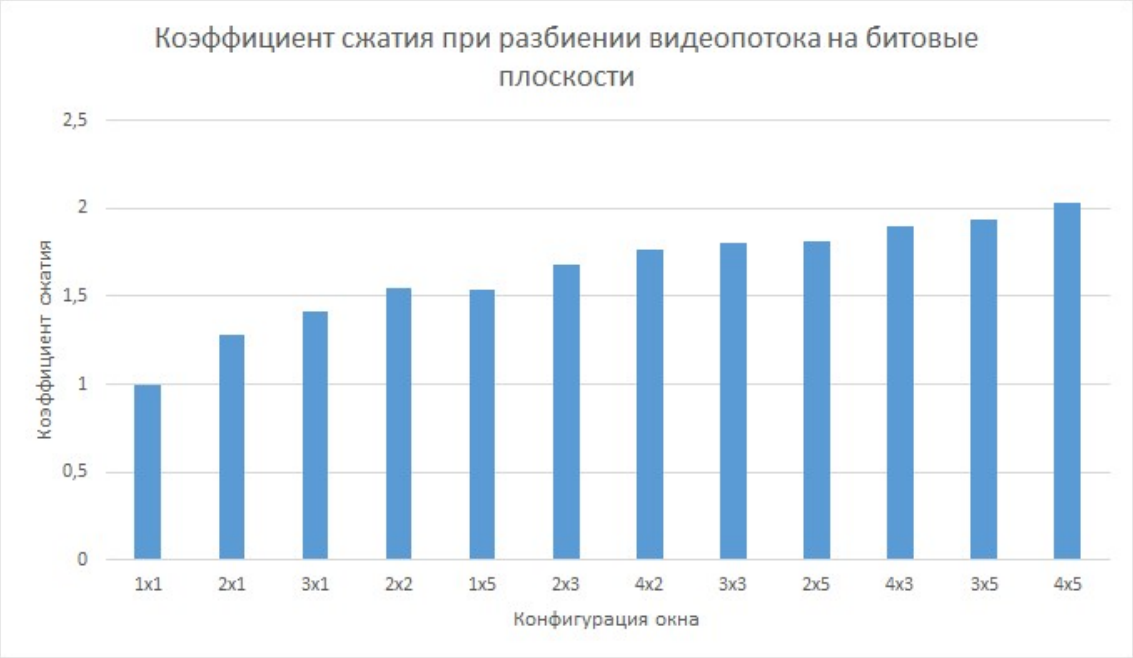
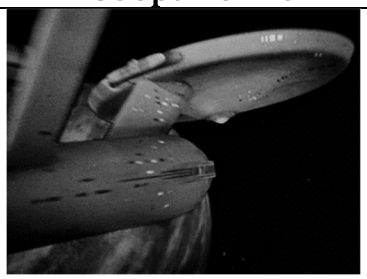


Рис. 78. Коэффициент сжатия при разбиении видеопотока на битовые плоскости

Очевидно, что в случае битовых плоскостей степень сжатия растёт вместе с увеличением окна. Также видно, что старшие битовые плоскости сжимаются существенно лучше младших (Глава V2.2.1). Это объясняется тем, что видеопотоки, получаемые в старших битовых плоскостях, практически полностью состоят из нулевых элементов, в то время как элементы, получаемые в младших битовых плоскостях, сильно зашумлены. Табл. 95 наглядно иллюстрирует это:

Табл. 95. Псевдокадры, получаемые в различных битовых плоскостях

Исходное изображение	**8 битовая плоскость**	**1 битовая плоскость**
		

Отсюда можно сделать вывод, что различные битовые плоскости должны сжиматься разными методами, это позволит резко улучшить степень сжатия. Например, для каждой из битовых плоскостей можно выбирать свою оптимальную конфигурацию окна сканирования. Из Табл. 94 получаем, что оптимальной

конфигурацией окна для каждой битовой плоскости является окно $4x5$, но в общем случае оптимальным для старших битовых плоскостей является окно с большей площадью, чем для младших.

Улучшение коэффициента сжатия ($CV = \frac{1}{CR}$) при разбиении на битовые плоскости по сравнению со сжатием без разбиения на битовые плоскости показано на Рис. 79:

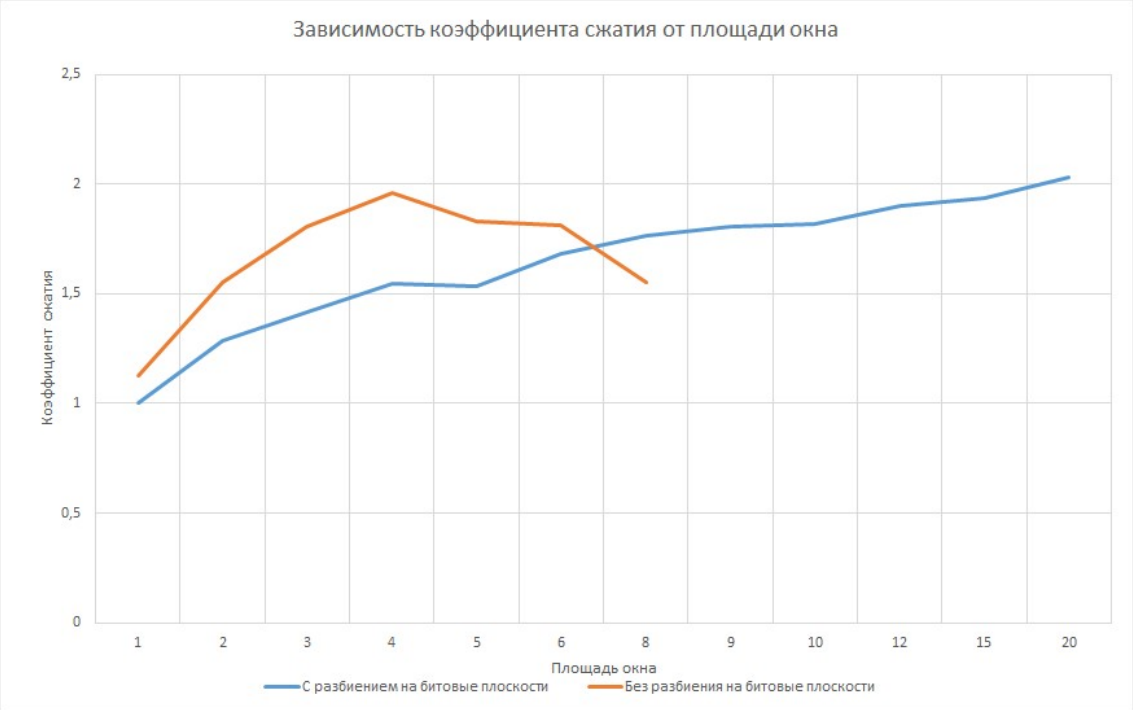


Рис. 79. Улучшение коэффициента сжатия при разбиении на битовые плоскости

Показанные степени сжатия для общих баз данных и для битовых плоскостей вовсе не являются предельными. Они могут быть значительно улучшены применением к однобитовым видеопотокам индивидуальных дополнительных методов сжатия.

2.4.2 Предварительное преобразование видеопотока в коды Грея

Ранее уже было показано, что младшие битовые плоскости сильно зашумлены. Шум является случайным плохо сжимаемым сигналом. Это приводит к плохой сжимаемости всего видеопотока в младших плоскостях. Бороться с этим можно, пытаясь выделить из видеопотоков осмысленный сигнал. Одним из способов добиться этого является преобразование яркости пикселей в коды Грея [м41].

Битовые плоскости, полученные с помощью кода Грея, более монотонны и в общем случае лучше поддаются сжатию. Преобразование яркости в коды Грея позволяет существенно улучшить степень сжатия метода фрагментарного кодирования видеопотока. Улучшение сжатия происходит во всех битовых плоскостях см. Рис. 80.

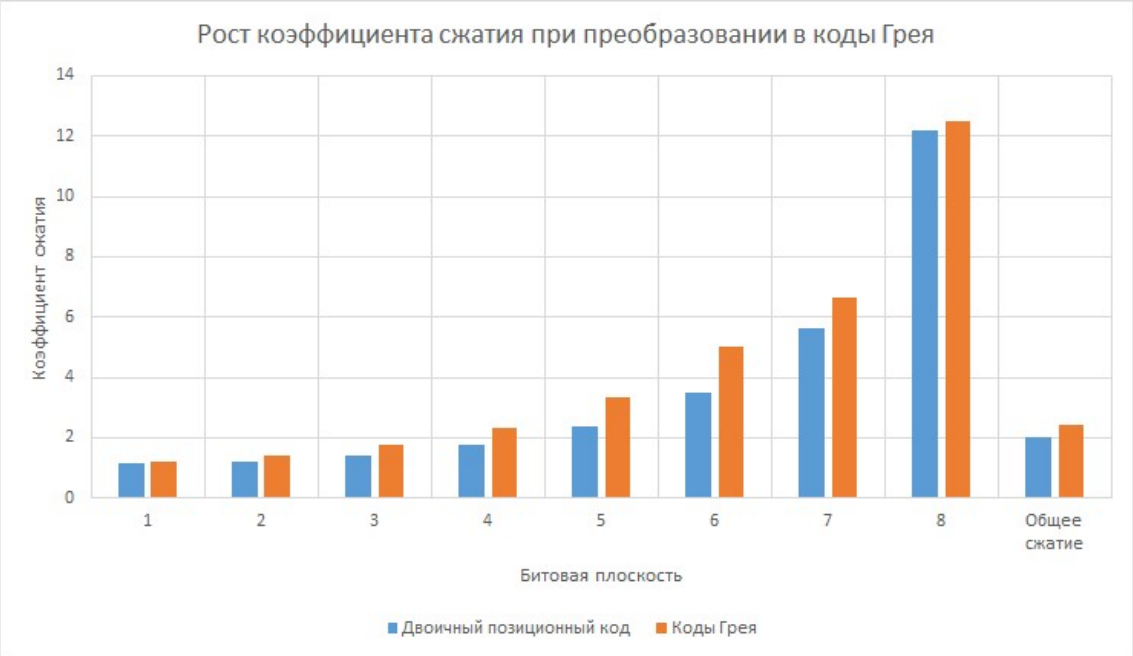


Рис. 80. Зависимость коэффициента сжатия по битовым плоскостям от преобразования в коды Грея

Из приведённого графика очевидно, что использование предварительного кодирования с помощью кодов Грея позволяет добиться существенного улучшения сжатия во всех битовых плоскостях, а значит, и общей степени сжатия.

На Рис. 81 показан прирост коэффициента сжатия для логических разностей, а на Рис. 82 показан прирост коэффициента сжатия для фрагментов.

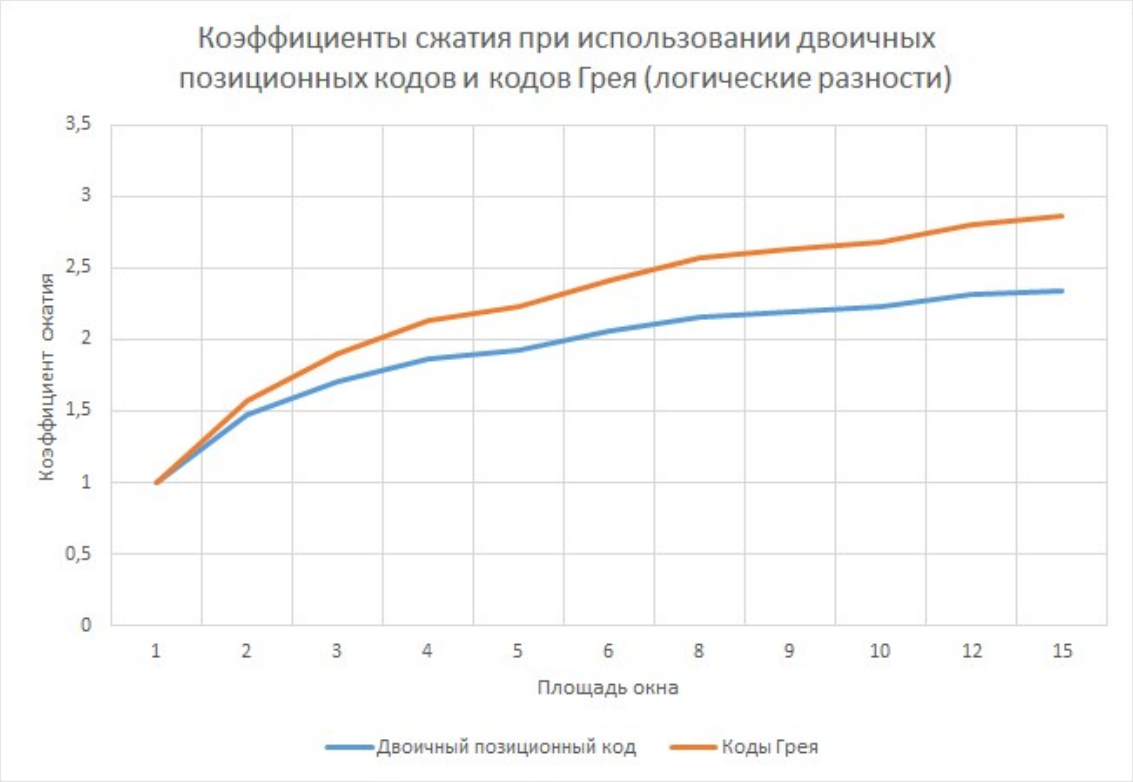


Рис. 81. Прирост коэффициента сжатия для логических разностей

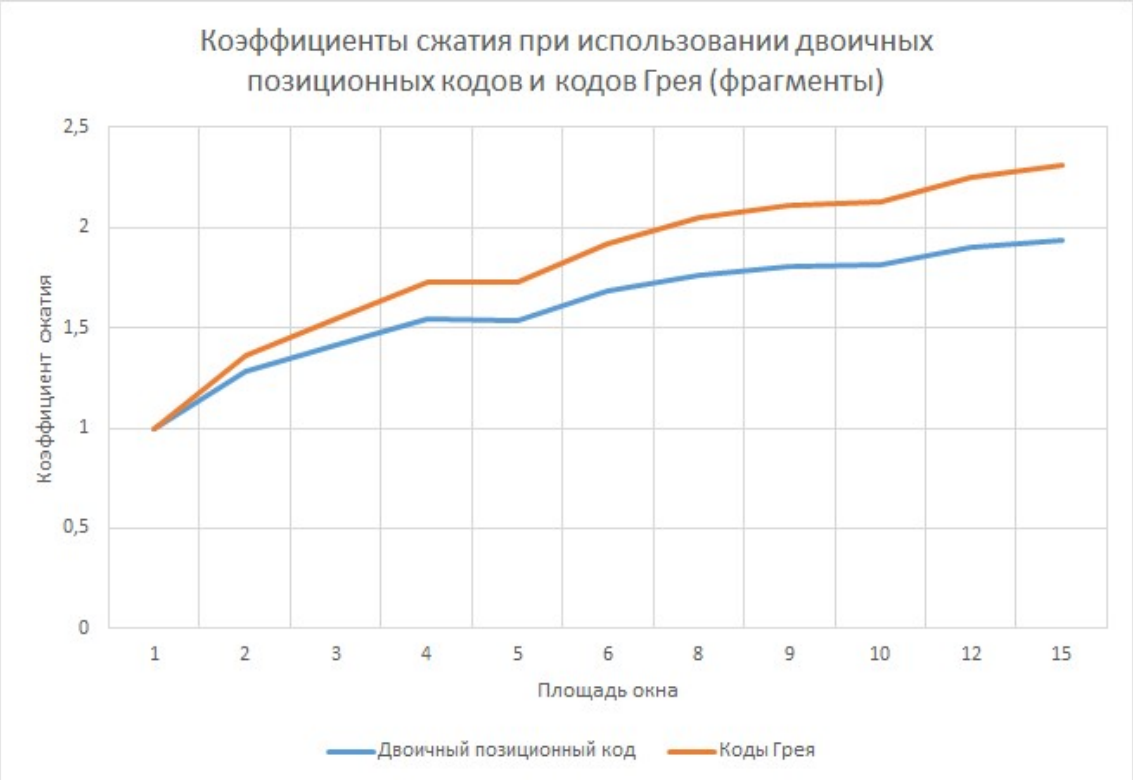


Рис. 82. Прирост коэффициента сжатия для фрагментов

2.4.3 Перестройка кодового дерева для более эффективной передачи базы элементов

Ранее был предложен метод представления двоичного дерева в виде битовой строки. При этом предложенная схема передачи позволяет передавать как элементы (хранящиеся в листьях дерева), так и структуру дерева, используя только два дополнительных бита на каждый элемент.

Предложенная схема значительно эффективнее непосредственной передачи таблицы кодов (или частот), но она может быть существенно улучшена. Двоичное дерево используется исключительно для получения префиксных кодов, важнейшей характеристикой которых является средняя длина. Легко показать, что существует множество деревьев, обеспечивающих одну и ту же среднюю длину кодов. Простейшим способом получения таких деревьев является перестановка узлов в пределах уровня. Например, деревья на Рис. 83 и Рис. 84 существенно различаются в получаемых на их основе префиксных кодах, но средняя длина кодов и обеспечиваемое сжатие при этом одинаковы.

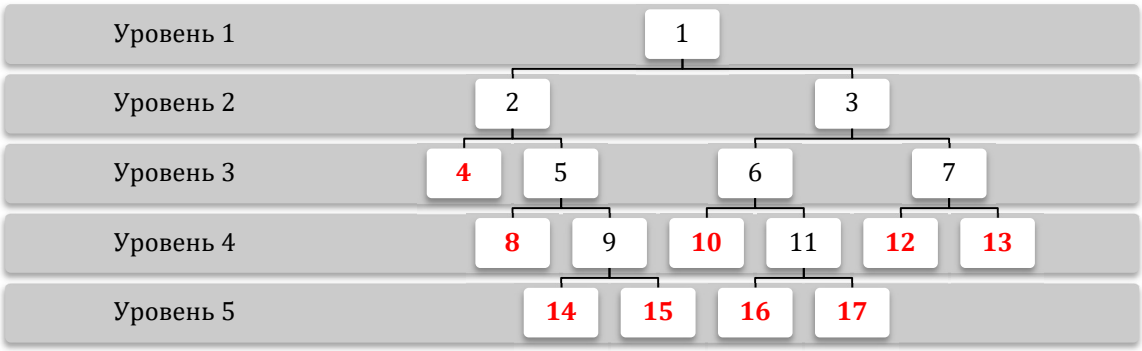


Рис. 83. Пример кодового дерева

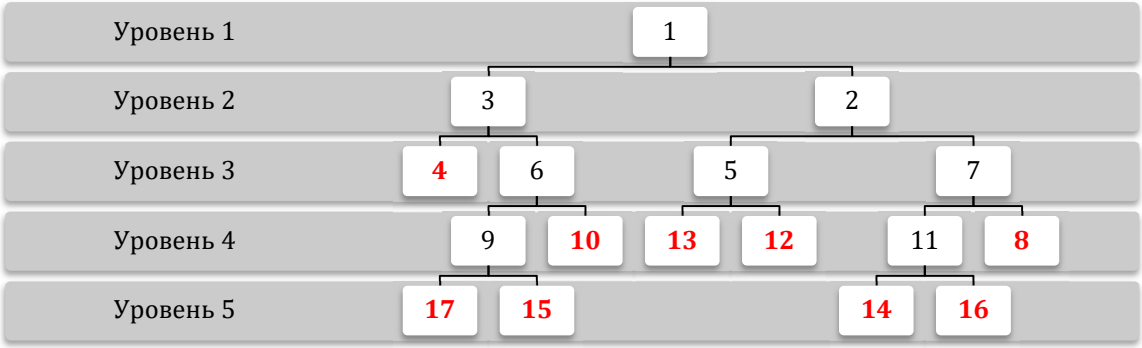


Рис. 84. Перестройка кодового дерева

Договоримся об определённом порядке перестановки узлов. Например, на каждом уровне все листья будем ставить левее всех делящих узлов. В этом случае дерево с Рис. 83 будет преобразовано в дерево, представленное на Рис. 85:

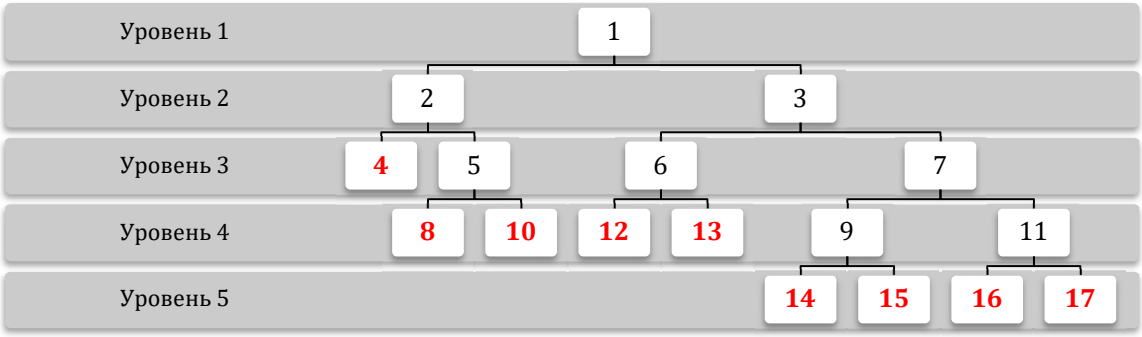


Рис. 85. Преобразованное дерево

Если дерево преобразовано подобным образом, то нулевой бит означает не просто один делящий узел, а то, что все узлы на обрабатываемом уровне правее заданного тоже делящие. Таким образом, для передачи дерева используется менее, чем два бита на каждый элемент. Эффективность этой перестановки растёт вместе с ростом размера кодового дерева.

2.5 Повышение эффективности сжатия за счёт потерь

2.5.1 Удаление младших битовых плоскостей

При разбиении исходного видеопотока на битовые плоскости все бинарные видеопотоки формируются независимо друг от друга, и, соответственно, есть возможность проводить эксперименты с качеством изображения, применяя методы сжатия с потерями для каждого бинарного видеопотока независимо от остальных (вплоть до полного исключения некоторых бинарных потоков из общего потока).

Уже не раз упоминалось, что младшие битовые плоскости сильно зашумлены. Чтобы повысить степень сжатия в младших битовых плоскостях было предложено использовать коды Грея. Этот метод позволяет существенно повысить степень сжатия, но в ряде случаев, когда не требуется точное восстановление исходного видеопотока, можно применять другие методы. Очевидно, что амплитуда яркости сигнала в m младших плоскостях не превышает $2^m - 1$ ($m = 1,2,3,4,5$), что составляет незначительную часть общей амплитуды яркости, поэтому самым простым способом повышения степени сжатия является удаление младших битовых плоскостей из кодируемого видеопотока и замена их константными значениями, то есть постоянным сигналом.

Наименьшее среднеквадратичное отклонение такого сигнала от исходного будет, когда эта константа равна среднему значению сигнала в течение всего видеопотока.

При удалении младших битовых плоскостей наиболее эффективным элементом является логическая разность. Ниже приведена Табл. 96, показывающая степень сжатия без потерь для каждой из восьми битовых плоскостей для различных окон:

Табл. 96. Зависимость степени сжатия от конфигурации окна и битовой плоскости (логические разности)

Окно	Битовая плоскость								Сжатие	
	1	2	3	4	5	6	7	8	CR	CC
1x1	1	1	1	1	1	1	1	1	1	1
1x2	0,9	0,86	0,77	0,69	0,62	0,57	0,54	0,51	0,68	1,46
2x1	0,89	0,84	0,76	0,68	0,62	0,57	0,54	0,51	0,68	1,48
1x3	0,88	0,83	0,72	0,6	0,51	0,44	0,39	0,35	0,59	1,7
3x1	0,88	0,83	0,71	0,6	0,51	0,43	0,39	0,35	0,59	1,71
2x2	0,85	0,8	0,68	0,56	0,45	0,37	0,31	0,27	0,54	1,87
4x1	0,85	0,8	0,69	0,56	0,45	0,37	0,31	0,27	0,54	1,86
1x5	0,86	0,81	0,69	0,55	0,43	0,34	0,27	0,22	0,52	1,93
1x6	0,85	0,8	0,68	0,54	0,41	0,31	0,24	0,19	0,5	1,99
2x3	0,82	0,78	0,66	0,52	0,4	0,3	0,24	0,18	0,49	2,05
3x2	0,84	0,79	0,67	0,52	0,4	0,3	0,24	0,19	0,49	2,03
6x1	0,84	0,79	0,67	0,53	0,4	0,31	0,24	0,19	0,5	2,02
4x2	0,81	0,76	0,64	0,5	0,37	0,27	0,2	0,14	0,46	2,16
8x1	0,81	0,77	0,65	0,51	0,38	0,28	0,2	0,15	0,47	2,14
3x3	0,81	0,76	0,64	0,5	0,36	0,26	0,19	0,13	0,46	2,2
1x10	0,83	0,78	0,66	0,52	0,38	0,27	0,19	0,12	0,47	2,13
2x5	0,8	0,75	0,64	0,49	0,36	0,25	0,18	0,12	0,45	2,23
2x6	0,79	0,75	0,63	0,49	0,35	0,24	0,16	0,11	0,44	2,27
4x3	0,78	0,74	0,62	0,48	0,34	0,24	0,16	0,1	0,43	2,31
6x2	0,8	0,75	0,63	0,49	0,35	0,24	0,16	0,1	0,44	2,27
12x1	0,81	0,76	0,65	0,5	0,36	0,25	0,17	0,11	0,45	2,22
1x15	0,82	0,77	0,65	0,51	0,37	0,25	0,16	0,09	0,45	2,21
3x5	0,78	0,74	0,62	0,47	0,34	0,23	0,15	0,09	0,43	2,34
4x5	0,76	0,72	0,6	0,46	0,32	0,21	0,13	0,07	0,41	2,44

На Рис. 86 показан рост коэффициента сжатия с учётом замены младших битовых видеопотоков на стационарные:

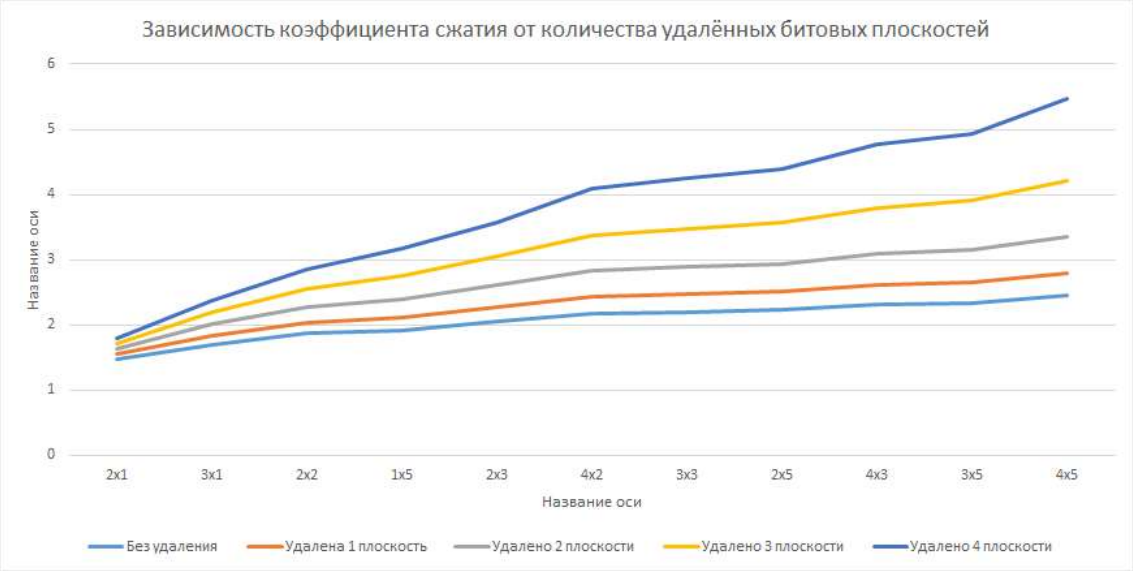
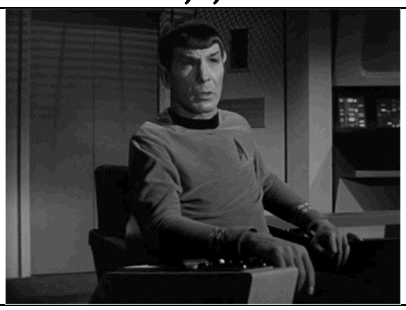
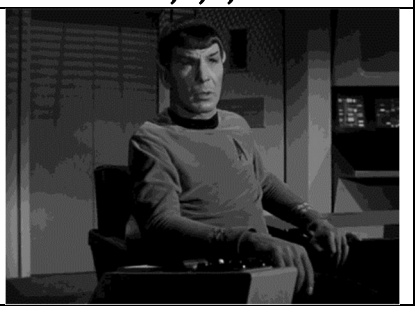


Рис. 86. Зависимость коэффициента сжатия от количества исключённых битовых плоскостей

Визуальные искажения, возникающие в сжимаемом видеопотоке, показаны в Табл. 97.

Табл. 97. Изображение после удаления младших битовых плоскостей

Исходное изображение	**1**	**1,2**
		
	1,2,3	**1,2,3,4**
		

Полученные данные позволяют выбирать количество заменяемых битовых плоскостей в зависимости от приоритета: минимального объёма передачи или качества изображения.

2.5.2 Предварительная фильтрация исходного видеопотока

Алгоритм фрагментарного сжатия видеопотока чувствителен к шумам на исходном изображении, так как зашумлённое окно скорее всего будет содержаться в базе элементов как отдельная запись.

Шумы могут быть удалены как на этапе предварительной обработки каждого кадра, так и после сбора базы элементов. С большой долей вероятности можно утверждать, что фрагменты с наименьшими частотами соответствуют зашумлённым окнам. Идея постобработки заключается в удалении самых редких элементов из базы и замене их наиболее близкими в визуальном смысле. Ниже более подробно рассмотрен первый подход.

Причин возникновения шумов довольно много: это и неправильная настройка или неисправность устройств получения изображений, и помехи, возникающие в каналах связи, и плохие условия съёмки (например, шум КМОП-матриц в темноте). Каждая из перечисленных выше причин приводит к появлению разных типов шумов. На данный момент не существует универсального алгоритма, способного бороться со всеми типами шума, поэтому существует множество специализированных алгоритмов.

Основной способ борьбы с шумами – это фильтрация. При фильтрации изображений для некоторых пикселей исходного изображения вычисляются новые значения. Новое значение пиксела зависит как от значения яркости обрабатываемого пиксела, так и от значений окрестных пикселов.

На практике для удаления шумов чаще всего используются усредняющие и порядковые фильтры. Именно эти фильтры предлагается использовать в методе фрагментарного сжатия видеопотока для подавления шумов.

Усредняющие фильтры вычисляют итоговую яркость пиксела как среднее (в определённом смысле) яркости всех пикселов окрестности. Все фильтры этой группы сглаживают резкие перепады яркости, что, с одной стороны, позволяет избавиться от шумов некоторого вида, а, с другой стороны, приводит к размытию границ объектов. В большинстве случаев размытие границ является побочным

малозаметным эффектом, но, если на изображении содержится большое количество маленьких деталей, усредняющие фильтры могут привести к их исчезновению. Эффективность фильтрации во многом зависит от способа вычисления «среднего».

В отличие от усредняющих фильтров, порядковые фильтры требуют предварительного ранжирования (упорядочивания) яркостей пикселов в пределах апертуры фильтра. Далее из упорядоченной последовательности выбирается определённый элемент, зависящий от целей фильтрации. Наиболее популярным порядковым фильтром является медианный фильтр.

Медианный фильтр приводит к замене сильно отличающегося от окрестности пиксела на медиану яркости окрестности. Такая замена позволяет эффективно удалять шум типа соль-и-перец (при условии, что шум занимает не более половины площади апертуры), при этом избегая сильного размытия контуров.

К недостаткам медианной фильтрации можно отнести низкую скорость работы (по сравнению с усредняющими фильтрами), т.к. для каждого пиксела необходимо выполнять упорядочивание пикселов окрестности по яркости.

Пороговая фильтрация – специально разработанный фильтр для подавления шума камеры и гранулярного шума, возникающего в процессе оцифровки изображений. Этот фильтр принципиально отличается от усредняющих и порядковых фильтров тем, что применяется не к одному изображению, а к двум последовательным кадрам видеопотока.

Для фильтра выбирается порог – модуль разности между двумя яркостями. Если для двух соответствующих пикселов на двух последовательных кадрах модуль разности не превышает заданного порога, то изменение яркости считается помехой и при кодировании яркость берётся из предыдущего кадра.

На первый взгляд этот фильтр кажется весьма грубым, но на самом деле искажения, вносимые данным фильтром (особенно при низком пороге), практически не заметны.

Рис. 87. Пояснение пороговой фильтрации (порог 10)

Наиболее заметные искажения возникают при движении больших объектов на слабоконтрастном фоне при высоком значении порога.

Использование предварительной фильтрации позволяет весьма сильно повысить эффективность сжатия (см. Рис. 88).

Рис. 88. Зависимость степени сжатия от используемого фильтра