УПРАВЛЕНИЕ СКОРОСТЬЮ И ОШИБКОЙ КОДИРОВАНИЯ В СИСТЕМЕ СЖАТИЯ И ПЕРЕДАЧИ ВИДЕОИНФОРМАЦИИ С ОГРАНИЧЕНИЯМИ НА ПАМЯТЬ ПЕРЕДАЮЩЕГО И ПРИНИМАЮЩЕГО УСТРОЙСТВ

Е.А. Беляев, А.М. Тюрликов
Санкт-Петербургский Государственный Университет Аэрокосмического Приборостроения

Аннотация

В работе рассматриваются алгоритмы сжатия видеоинформации в системах передачи, в которых общий объем памяти как передающего, так и принимающего устройств много меньше количества бит, необходимых для хранения одного кадра видеопоследовательности. Предложен алгоритм управления битовой скоростью и среднеквадратической ошибкой кодирования, обеспечивающий равномерное приемлемое визуальное качество с учетом ограничений на пропускную способность канала и объема памяти.

Введение

Системы передачи видеоинформации по проводным и беспроводным каналам связи широко применяются во многих сферах жизнедеятельности. При этом, подобные системы требуют высоких затрат как памяти, так и вычислительных ресурсов. Данная работа посвящена разработке алгоритмов сжатия видеоинформации с высоким разрешением и визуальным качеством, которые можно реализовать при достаточно небольших объемах памяти как передающего, так и принимающего устройств (общий объем памяти устройства много меньше количества бит, необходимых для хранения одного кадра видео последовательности).

Существенные ограничения на память устройства делают невозможным методы устранения временной избыточности, характерной для источников видеоинформации (компенсация движения, временная фильтрация и т.д.). Поэтому разрабатываемые алгоритмы, могут достигать определенной степени сжатия только за счет устранения пространственной избыточности. Решению поставленной задачи могут удовлетворять алгоритмы, основанные на дискретном косинусном преобразовании (JPEG, MPEG2 и H.264) [1], а также алгоритмы, базирующиеся на дискретном вейвлетом преобразовании (JPEG2000) [2].

Стандарт JPEG2000 является более адаптированным решением для данной задачи, так как обеспечивает управление битовой скоростью с достаточно высокой степенью точности (в отличие от алгоритмов, основанных на дискретном косинусном преобразовании). С другой стороны, при реализации JPEG2000 возникают следующие сложности. При ограничениях на объем памяти исходное изображение разбивается на независимо кодируемые сегменты (tile). Стандарт JPEG2000 обеспечивает примерно одинаковые битовые затраты на каждый сегмент. Но, так как статистические свойства сегментов могут существенно отличаться, возможен следующий эффект: некоторые сегменты сжимаются с высоким визуальным качеством, а некоторые сжимаются с очень плохим качеством.

В данной работе предлагается алгоритм, который обеспечивает приемлемое качество для всех сегментов для данной пропускной способности канала.

1. Управление битовой скоростью в стандарте JPEG2000

Схема кодирования, используемая в стандарте JPEG2000, кратко может быть описана следующим образом. После преобразования цветового пространства каждый сегмент изображения подвергается процедуре многоуровневого дискретного вейвлетного преобразования [3, 4]. Для этого используется два вейвлетных фильтра: низкочастотный фильтр $h_0(n)$ и высокочастотный фильтр $h_1(n)$. При разложении сигнала сегмента изображения вначале выполняется разложение по строкам, а затем по столбцам. Результатом разложения являются 4 матрицы: HH_0, HL_0, LH_0, LL_0, соответствующие фильтрации фильтром $h_1(n)$ по строкам и по столбцам; фильтром $h_1(n)$ по строкам и $h_0(n)$ по столбцам; фильтром $h_0(n)$ по строкам и $h_1(n)$ по столбцам; фильтром $h_0(n)$ по строкам и столбцам. Далее производится децимация (прореживание) полученных матриц по строкам и столбцам с коэффициентом 2. Затем матрица LL_0 подвергается дальнейшему вейвлетному разложению. Его результатом являются матрицы: HH_1, HL_1, LH_1, LL_1 (рис. 1). Такое разложение повторяется v раз. Результатом разложения является набор из $3v+1$ матриц уменьшающейся размерности.

Дальнейшее кодирование состоит из трех основных шагов. На первом шаге (Tier-1) происходит сжатие без потерь коэффициентов вейвлетного разложения. Каждая из матриц разложения делится на блоки, которые кодируются независимо друг от друга. Коэффициенты разложения блоков представляются в двоичном виде. Двоичные символы коэффициентов, относящиеся к одному разряду, образуют битовую плоскость. Кодирование осуществляется по битовым плоскостям: начиная от битовой плоскости, соответствующей старшему разряду коэффициентов. Каждая битовая плоскость (кроме самой старшей) кодируется в три прохода так, что кодирование может быть прервано после любого числа проходов. Общее количество проходов $p = 3(n_b - 1) + 1$, где n_b – количество битовых плоскостей в блоке.

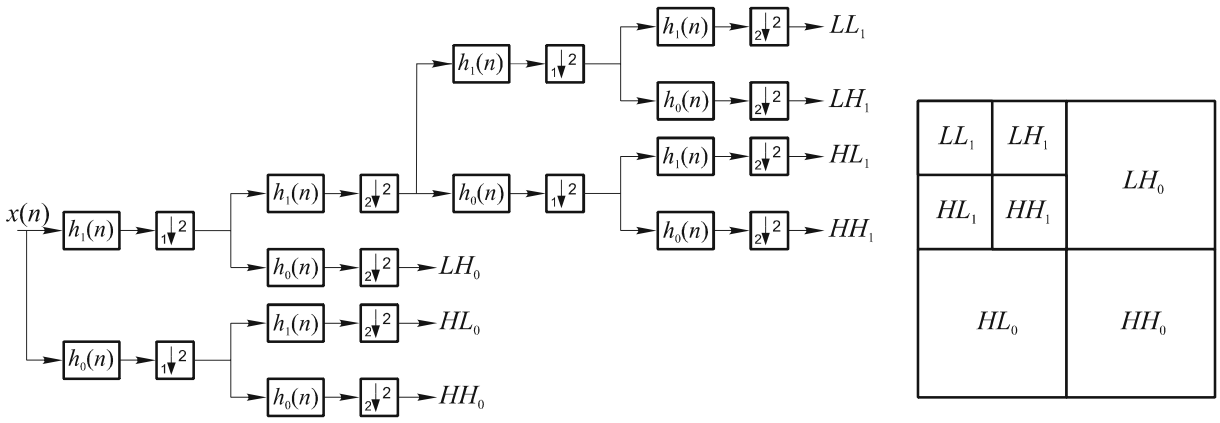

Рис. 1. Вейвлетное разложение изображения

На втором шаге кодирования (RD-Optimization), для каждого блока определяются проходы, которые будут переданы декодеру. Пронумеруем блоки, входящие в сегмент изображения $i = 1, 2, \ldots$. Блоку i соответствует целое число проходов, каждый из которых вносит свой вклад в общие битовые затраты и качество восстановленного сегмента изображения. Обозначим за r_i^k количество бит, а за d_i^k среднеквадратическую ошибку при передаче проходов с номерами $1, 2, \ldots, k$ (см. рис. 2), $k = 1, \ldots, p$. Декодеру передаются проходы с номерами 1, 2, ..., n_i. Номер последнего прохода n_i, который будет передан, будем называть *точкой отсечения блока*.

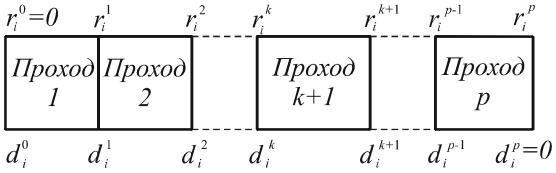

Рис. 2. Структура кодового блока

Обозначим через $\mathbf{n} = \{n_i\}$ *вектор отсечения сегмента*, компонента n_i которого означает, что $i-$ый блок отсекается в точке с номером n_i. Общая среднеквадратическая ошибка для сегмента

$$d(\mathbf{n}) = \sum_i d_i^{n_i},$$

где $d_i^{n_i}$ – среднеквадратическая ошибка для блока i после отсечения его в точке n_i.

Согласно [5] среднеквадратическая ошибка может быть вычислена как

$$d_i^{n_i} = w_i^2 \sum_{\{u,v\}} \left(s_i^{n_i}[u,v] - s_i[u,v] \right)^2,$$

где $s_i[u,v]$ – исходный вейвлетный коэффициент с координатами u и v в блоке i; $s_i^{n_i}[u,v]$ – восстановленный вейвлетный коэффициент с учетом отсечения блока в точке n_i, w_i^2 – весовой коэффициент матрицы вейвлетного разложения. Суммарные битовые затраты на сегмент

$$r(\mathbf{n}) = \sum_i r_i^{n_i},$$

где $r_i^{n_i}$ – битовые затраты на блок i после отсечения его в точке n_i. Оптимизационная задача заключается в поиске вектора отсечения $\mathbf{n}_{opt}$, такого, что

$$\begin{cases} d(\mathbf{n}_{opt}) = \min\limits_{\{\mathbf{n}\}} d(\mathbf{n}) \\ r(\mathbf{n}_{opt}) \le r_{\max}. \end{cases} \quad (1)$$

Для решения оптимизационной задачи (1) в стандарте JPEG2000 [2, приложение J.10] применяется метод множителей Лагранжа. Метод множителей Лагранжа подразумевает, что для каждого блока используется множество *подходящих точек отсечения*. Для любых двух подходящих точек отсечения j и k, $j < k$ блока i

$$\frac{d_i^j - d_i^k}{r_i^k - r_i^j} > 0$$

и

$$\min_{k<n_i} \frac{d_i^{n_i} - d_i^k}{r_i^k - r_i^{n_i}} > \min_{j>n_i} \frac{d_i^j - d_i^{n_i}}{r_i^{n_i} - r_i^j}.$$

Иными словами, множество подходящих точек отсечения есть множество точек отсечения, лежащих на выпуклой кривой скорость/искажение. Множество векторов отсечения, компоненты которого состоят из подходящих точек отсечения, обозначим за $\mathbf{N}$. Любой вектор отсечения $\mathbf{n} \in \mathbf{N}$ минимизирующий

$$r(\mathbf{n}) + \lambda d(\mathbf{n}) = \sum_i (r_i^{n_i} + \lambda d_i^{n_i}),$$

обладает таким свойством, что ошибка не может быть уменьшена без увеличения битовых затрат, λ – множитель Лагранжа. Так как все блоки кодируются независимо, можно считать, что

$$\min \sum_i (r_i^{n_i} + \lambda d_i^{n_i}) = \sum_i \min(r_i^{n_i} + \lambda d_i^{n_i}).$$

Это означает, что итоговое решение есть сумма оптимальных решений для каждого блока i, поэтому достаточно минимизировать значение $r_i^{n_i} + \lambda d_i^{n_i}$

для каждого блока отдельно. Решение оптимизационной задачи заключается в поиске такого значения λ, что $r(\mathbf{n}) = r_{\max}$. При этом каждому значению λ соответствует вектор отсечения $\mathbf{n}$, компонента которого $n_i = \arg\min_k \left\{ r_i^k + \lambda d_i^k \right\}$.

Поиск искомого значения λ может осуществляться следующим образом. Первоначально, из некоторых соображений, выбираются значения $\lambda_{\min}$ и $\lambda_{\max}$. Затем выполняется поиск значения λ путем деления пополам отрезка $[\lambda_{\min}, \lambda_{\max}]$.

На шаге 3 (Tier-2) кодирования, формируется сжатый сегмент изображения в соответствии с найденным вектором отсечения $\mathbf{n}_{opt}$.

2. Система передачи видеоинформации на основе алгоритма JPEG 2000

Система передачи на основе JPEG 2000 может быть описана следующим образом. Будем считать, что время в системе дискретно и разделено на окна единичной длины $[t, t+1)$, а под моментом времени t будем подразумевать окончание этого интервала времени. Источник видеоинформации через одинаковые интервалы времени подает на вход кодера очередной сегмент изображения. Кодер работает в реальном масштабе времени, поэтому на момент окончания окна $[t, t+1)$ в *сглаживающий буфер передатчика* [6] помещается сжатый сегмент, размером $r(\mathbf{x}_t)$ бит, $\mathbf{x}_t \in \mathbf{N}_t$. В зависимости от количества бит в буфере передатчика формируются требования к следующему сжимаемому сегменту. Количество бит в буфере передатчика $b(t)$ в момент времени t, после помещения в буфер очередного сжатого сегмента, объемом $r(\mathbf{x}_t)$ бит и передаче по каналу с постоянной скоростью r, изменяется следующим образом:

$$b(t) = \max\left\{0, b(t-1) - r\right\} + r(\mathbf{x}_t). \qquad (2)$$

Решение оптимизационной задачи (1) гарантирует, что буфер передатчика размером $B_0 = r$ бит никогда не переполнится, если для каждого передаваемого сегмента выбирается такой вектор деления $\mathbf{x}_t \in \mathbf{N}_t$, чтобы

$$\begin{cases} d(\mathbf{x}_t) = \min\limits_{\mathbf{n}\in\mathbf{N}_t} d(\mathbf{n}) \\ r(\mathbf{x}_t) \le r. \end{cases}$$

Таким образом, в системе передачи на основе JPEG2000 на каждый сегмент расходуется количество бит, максимально близкое к скорости канала r. Поэтому при сильных отличиях статистических свойств сегментов, описанная схема управления может привести к следующему эффекту: некоторые сегменты могут сжиматься с неоправданно высоким визуальным качеством, а некоторые могут сжиматься с очень плохим качеством.

3. Система передачи видеоинформации на основе модифицированного алгоритма JPEG2000

В данной работе предлагается управлять не только количеством бит на сжатый сегмент, но и визуальным качеством каждого сегмента. То есть сегменты предлагается сжимать с одинаковым, приемлемым визуальным качеством, с учетом ограничений на пропускную способность канала. Для того, чтобы обеспечить ошибку не более d, необходимо аналогично (1), найти такой вектор отсечения $\mathbf{x}_t \in \mathbf{N}_t$, чтобы

$$\begin{cases} r(\mathbf{x}_t) = \min\limits_{\mathbf{n}\in\mathbf{N}_t} r(\mathbf{n}) \\ d(\mathbf{x}_t) \le d. \end{cases}$$

При этом очевидно, что битовые затраты на каждый сегмент могут существенно отличаться и превосходить пропускную способность канала r. Поэтому объем буфера передатчика $B_0 > r$. При этом задержка, вносимая при использовании буфера, равна B_0 / r сегментов [7].

Будем предполагать, что, не смотря на то, что статистические свойства отдельных сегментов изображения могут существенно отличаться друг от друга, статистические свойства кадра в целом меняются незначительно. С учетом этого предположения оптимизационную задачу можно сформулировать более точно. Для каждого сегмента t необходимо выбрать вектор отсечения $\mathbf{x}_t$ так, чтобы

$$\begin{cases} \text{минимизировать } \max\limits_t d(\mathbf{x}_t) \\ b(t) \le B_0. \end{cases} \qquad (3)$$

Дальнейшее изложение построено следующим образом. Сначала описывается алгоритм динамического программирования, позволяющий получить решение оптимизационной задачи (3) при отсутствии ограничений на память и вычислительные ресурсы. Затем предлагается алгоритм последовательного поиска, для которого значение максимальной ошибки на сегмент совпадает с алгоритмом динамического программирования. Алгоритм последовательного поиска приводится для оценки характеристик предложенного ниже эвристического алгоритма управления, работающего в условиях ограничения на память.

Решение динамическим программированием

Если мы не имеем ограничений на вычислительные ресурсы и память, то поставленную задачу (3) можно решить динамическим программированием. При этом на каждом шаге алгоритма:

1. Отбрасываются пути, приводящие к переполнению буфера.
2. Из множества путей с одинаковой максимальной ошибкой d удаляются все пути кроме одного, для которого количество бит в буфере минимально.

3. Из множества путей с одинаковым количеством бит в буфере удаляются все пути кроме одного, для которого минимальна максимальная ошибка d.

Для наглядности рассмотрим пример из последовательности трех сегментов (см. рис. 3), каждому из которых соответствуют три вектора отсечения: $\{\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_3\}$ для сегмента a; $\{\mathbf{b}_1,\mathbf{b}_2,\mathbf{b}_3\}$ для сегмента b, $\{\mathbf{c}_1,\mathbf{c}_2,\mathbf{c}_3\}$ для сегмента c. Каждому вектору соответствует точка на плоскости «скорость-искажение».

На рис. 4 схематически показано решение оптимизационной задачи путем динамического программирования. Каждый шаг алгоритма для наглядности разбит на этап записи сжатого сегмента в буфер и этап передачи данных из буфера в канал. На шаге 1 пути не отбрасываются. На шаге 2 из возможных 9-ти путей отбрасываются 7. На шаге 3 из возможных 6-ти путей отбрасываются 3. Оптимальному решению соответствует выбор векторов отсечения $\mathbf{a}_2$ для сегмента a, $\mathbf{b}_2$ для сегмента b, $\mathbf{c}_2$ для сегмента c.

Решение последовательным поиском

Максимальное значение ошибки на сегмент, найденное динамическим программированием, может быть получено также и последовательным поиском. Обозначим за $\mathbf{D}$ множество возможных значений, переупорядоченных по возрастанию ошибок для всех сегментов, дополненное значениями 0 и ∞. За d_i обозначим элемент этого множества. Для рассмотренного выше примера такое множество имеет следующий вид $\mathbf{D}=\{0, d(\mathbf{b}_3), d(\mathbf{a}_3), d(\mathbf{a}_2), d(\mathbf{b}_2), d(\mathbf{c}_3), d(\mathbf{c}_2), d(\mathbf{b}_1), d(\mathbf{c}_1), d(\mathbf{a}_1), \infty\}$.

Работа алгоритма разбивается на этапы. На этапе i выбирается порог $\tilde{d}=d_i$. Для каждого сегмента t выбирается вектор отсечения $\mathbf{x}_t\in\mathbf{N}_t$, такой что

$$\mathbf{x}_t=\arg\max_{\mathbf{n}\in\mathbf{N}_t}\left\{d(\mathbf{n}):d(\mathbf{n})\le\tilde{d}\right\}, \quad (4)$$

а также согласно (2) вычисляется количество бит в буфере $\tilde{b}(t)$ при последовательной передаче сжатых сегментов. Если, на этапе i, в какой либо момент времени буфер объемом B_{con} переполнился, то производятся аналогичные действия на этапе $i+1$, в противном случае решение найдено и порог $\tilde{d}$ является значением решения оптимизационной задачи.

Описанный алгоритм будем называть *алгоритмом последовательного поиска* (см. рис. 5).

Утверждение 1. Не существует последовательности векторов отсечения, не приводящих к переполнению буфера и имеющих максимальное значение среднеквадратической ошибки меньше, чем $\tilde{d}$, где $\tilde{d}$ – решение, полученное алгоритмом последовательного поиска.

Доказательство. Предположим, что алгоритм последовательного поиска остановился на этапе i. Тогда для любого этапа j, предшествующего этапу i для кадра, содержащего L сегментов, в процессе работы алгоритма выбирается последовательность векторов $\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_{\mathrm{L}}$ таких, что $d(\mathbf{x}_t)\le d_j$, $t\in[1...\mathrm{L}]$, $\mathbf{x}_t\in\mathbf{N}_t$ и, из описания алгоритма, в некоторый момент времени τ буфер переполнился:

$$\tilde{b}(\tau)>B_{con}. \quad (5)$$

Теперь рассмотрим любую другую последовательность векторов $\mathbf{y}_1,\mathbf{y}_2,...,\mathbf{y}_{\mathrm{L}}$, таких, что $d(\mathbf{y}_t)\le d_j, t\in[1...\mathrm{L}], \mathbf{y}_t\in\mathbf{N}_t$ и обозначим за $b(t)$ количество бит в буфере в момент времени для данной последовательности. Из (4) следует, что для любого момента времени $d(\mathbf{x}_t)\ge d(\mathbf{y}_t)$. При этом известно, что каждому сегменту соответствует такое множество векторов отсечения, что количество бит, затрачиваемое на сегмент, убывает с ростом ошибки. Поэтому для любого момента времени

$$r(\mathbf{x}_t)\le r(\mathbf{y}_t). \quad (6)$$

С учетом условия (6) и выражения (2) получим, что при $\tilde{b}(0)=b(0)=b_0$, во все моменты времени $\tilde{b}(t)\le b(t)$. Поэтому, с учетом (5), в момент времени $\tau'<\tau$ значение $b(\tau')>B_{con}$.

Алгоритм управления скоростью и ошибкой кодирования при ограничениях на память

Два описанных выше метода подразумевают, что в системе нет ограничений на память. В данной работе предлагается эвристический алгоритм управления, который позволяет получить оценку значения $\tilde{d}$ алгоритма последовательного поиска в процессе работы системы, при таких ограничениях на память. Обозначим эту оценку $\hat{d}(t)$. Оценивать значение $\tilde{d}$ предлагается следующим образом. До тех пор пока количество бит в буфере не превысит некоторый порог B_H, все сегменты сжимаются с ошибкой не большей $\hat{d}(t)$. Превышение порога B_H означает, что поддержание ошибки на уровне $\hat{d}(t)$ невозможно для данной пропускной способности канала. Поэтому сначала буфер опустошается, после чего оценка ошибки $\hat{d}(t)$ увеличивается.

Алгоритм описывается при помощи следующих трех шагов.

Шаг 0.

0.1 Установить начальное значение $t=0$, $\hat{d}(0)=d_0$, $b(0)=0$.

0.2 Перейти к шагу 1.

Шаг 1. (Заполнение буфера)

1.1 $t=t+1$

1.2 Передать в канал $\min\{b(t-1),r\}$ бит.

1.3 Выполнить сжатие сегмента t без потерь и найти множество векторов отсечения $\mathbf{N}_t$.

1.4 Выполнить поиск $\mathbf{x}_t \in \mathbf{N}_t$, такого, что

$$\begin{cases} r(\mathbf{x}_t) = \min\limits_{\mathbf{n}\in\mathbf{N}_t} r(\mathbf{n}) \\ d(\mathbf{x}_t) \le \hat{d}(t). \end{cases}$$

1.5 Если $\max\{0, b(t-1)-r\} + r(\mathbf{x}_t) > B_H$ то перейти к шагу 2.4.

1.6 Поместить в буфер передатчика $r(\mathbf{x}_t)$ бит и перейти к шагу 1.1.

Шаг 2. (Опустошение буфера)

2.1 $t = t+1$

2.2 Передать в канал $\min\{b(t-1), r\}$ бит.

2.3 Выполнить сжатие сегмента t без потерь и найти множество векторов отсечения $\mathbf{N}_t$.

2.4 Вычислить количество бит, необходимое для освобождения буфера при передаче сегмента с ошибкой не более d_{empty}, путем поиска вектора отсечения $\mathbf{x}_t \in \mathbf{N}_t$, такого, что

$$\begin{cases} r(\mathbf{x}_t) = \min\limits_{\mathbf{n}\in\mathbf{N}_t} r(\mathbf{n}) \\ d(\mathbf{x}_t) \le d_{empty}. \end{cases}$$

2.5 Выполнить поиск $\mathbf{y}_t \in \mathbf{N}_t$, такого, что

$$\begin{cases} d(\mathbf{y}_t) = \min\limits_{\mathbf{n}\in\mathbf{N}_t} d(\mathbf{n}) \\ r(\mathbf{y}_t) \le \min\{B_0 - b(t), r(\mathbf{x}_t)\}. \end{cases}$$

2.6 Если значение $\max\{0, b(t-1)-r\} = 0$, то $\hat{d}(t+1) = \hat{d}(t) + \Delta d$, и перейти к шагу 1.4.

2.7 Поместить в буфер передатчика $r(\mathbf{y}_t)$ бит и перейти к шагу 2.1.

В качестве примера работы алгоритма на рис. 6 показана зависимость степени заполненности буфера от номера сегмента. Участки кривой, относящиеся к шагу 1 и шагу 2 алгоритма, показаны тонкой и толстой непрерывными линиями, соответственно.

Утверждение 2. Пусть алгоритму последовательного поиска с объемом буфера B_{con} соответствует максимальное значение среднеквадратической ошибки $\tilde{d}$. Тогда, при использовании предложенного алгоритма с порогом $B_H = B_{con}$ и начальным значением оценки порога ошибки $\hat{d}(0) < \tilde{d}$, для всех моментов времени t выполняется неравенство $\hat{d}(t) \le \tilde{d} + \Delta d$.

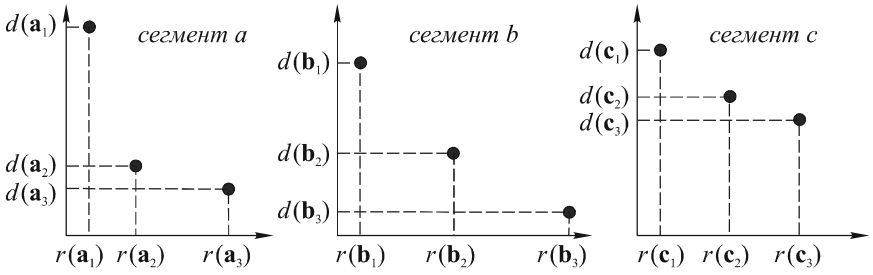

Рис. 3. Последовательность из трех сегментов

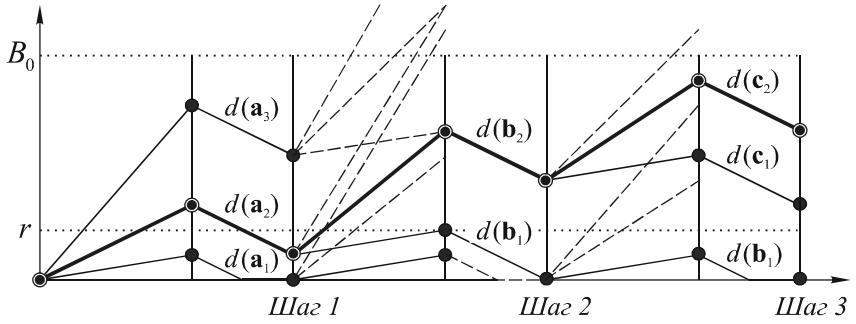

Рис. 4. Решение динамическим программированием

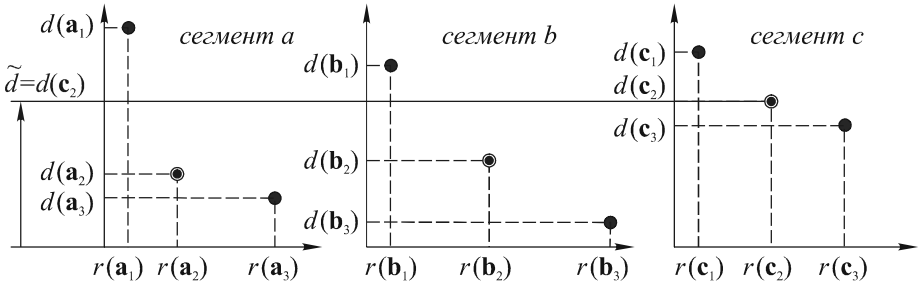

Рис. 5. Решение последовательным поиском

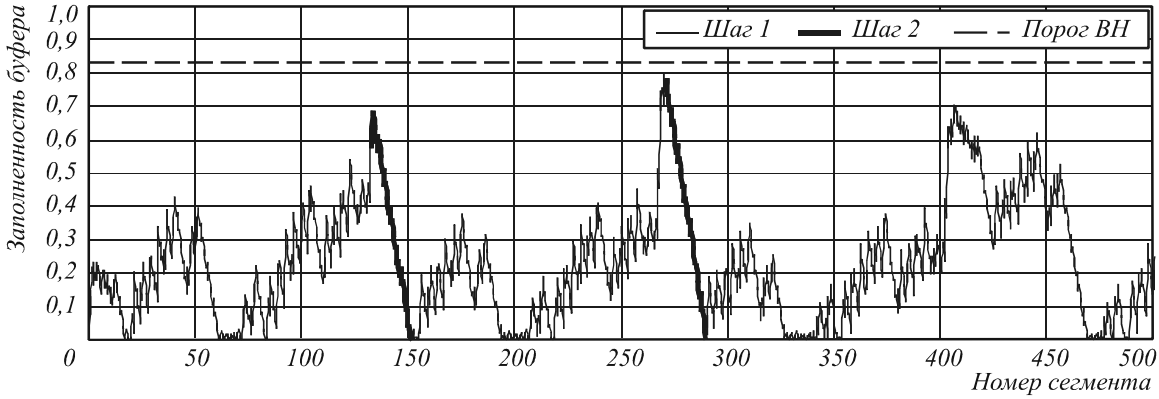

Рис. 6. Пример работы алгоритма

Доказательство. Обозначим за $\tilde{b}(t)$ количество бит в буфере объемом B_{con} при последовательном поиске. Количество бит, помещаемых в буфер в момент времени t при последовательном поиске, обозначим $\tilde{r}(t)$. Из описания алгоритма последовательного поиска, порог ошибки $\tilde{d}$ обеспечивает для всех моментов времени выполнение неравенства

$$\tilde{b}(t) \le B_{con} .$$

Пусть предложенный алгоритм начинает работать со значением оценки порога ошибки $\hat{d}(0) < \tilde{d}$ и в некоторый момент времени τ значение оценки порога ошибки первый раз попадет в диапазон (в противном случае, для всех моментов времени t, $\hat{d}(t) \le \tilde{d}$)

$$\tilde{d} < \hat{d}(\tau) \le \tilde{d} + \Delta d , \qquad (7)$$

где Δd – шаг изменения оценки порога ошибки. Количество бит в буфере передатчика (см. п. 2.6 алгоритма) перед помещением в буфер сжатого сегмента, в момент времени τ

$$b(\tau) = 0 , \qquad (8)$$

С учетом (7) в моменты времени $t \ge \tau$

$$r(\mathbf{x}_t) < \tilde{r}(t) , \qquad (9)$$

Из (2), (8) и (9) следует, что в моменты времени $t \ge \tau$

$$\max\{0, b(t-1) - r\} + r(\mathbf{x}_t) < \tilde{b}(t) \le B_{con} = B_H .$$

Таким образом, начиная с момента времени τ, условие $\max\{0, b(t-1) - r\} + r(\mathbf{x}_t) > B_H$ (см. п. 1.5 алгоритма) не будет выполняться, и, следовательно, алгоритм не попадет в шаг 2.6, тогда для всех сегментов значение порога ошибки

$$\begin{cases} \hat{d}(t) \le \tilde{d}, & \text{если } t < \tau \\ \hat{d}(t) \le \tilde{d} + \Delta d, & \text{если } t \ge \tau. \end{cases}$$

Из утверждения 2 следует, что при выборе порога $B_H = B_{con}$ в моменты времени $t \ge \tau$ среднеквадратическая ошибка на сегмент не превышает $\tilde{d} + \Delta d$.

4. Выбор параметров для алгоритма управления и практические результаты

Для получения практических результатов была использована открытая реализация JPEG2000 стандарта JASPER версии 1.701.0. Для сравнения использовались три тестовые видеопоследовательности разрешением 1920x1080 с различными статистическими свойствами: видеопоследовательность, содержащая компьютерную графику (test1); видеопоследовательность, содержащая фрагменты, как компьютерной графики, так и естественные (фотографические) фрагменты (test2); а также видеопоследовательность, содержащая только естественные фрагменты (test3). Для упрощения представления результатов каждая тестовая последовательность формировалась из одинаковых кадров.

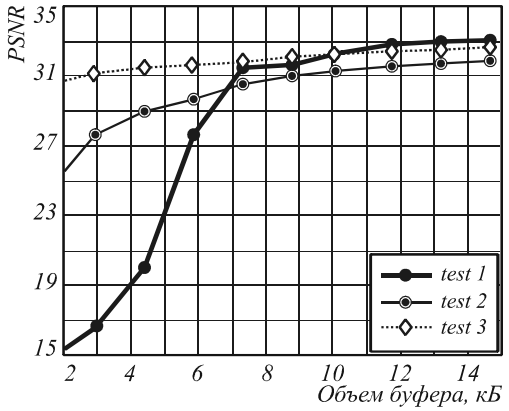

Рис. 7. Зависимость максимальной среднеквадратической ошибки восстановления для сегмента от объема буфера при последовательном поиске. Параметр rate=0,04

Будем полагать, что из ограничений на память устройства задан объем сглаживающего буфера B_0 и известна минимально возможная скорость канала. Необходимо выбрать значения B_H и d_{empty}. Для этого предлагается следующая последовательность действий. Для тестовых последовательностей строится зависимость максимальной среднеквадратиче-

ской ошибки восстановления для сегмента от объема буфера при последовательном поиске (см. рис. 7) для минимально возможной скорости канала. Исходя из полученных зависимостей, выбирается объем буфера B_{empty}, который обеспечивает максимальную, но приемлемую среднеквадратическую ошибку d_{empty} для всех тестовых последовательностей при опустошении буфера и значение $B_H = B_0 - B_{empty}$.

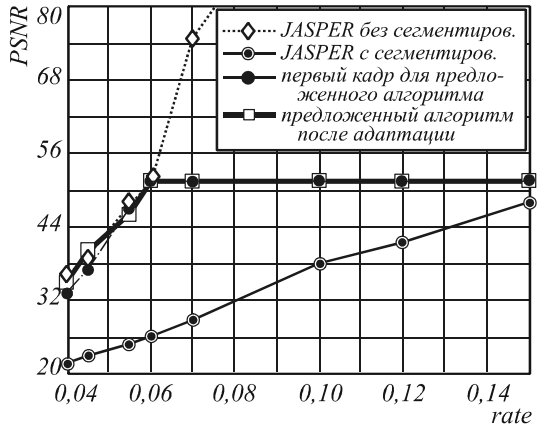

Рис. 8. Зависимость скорость/искажение для последовательности test1

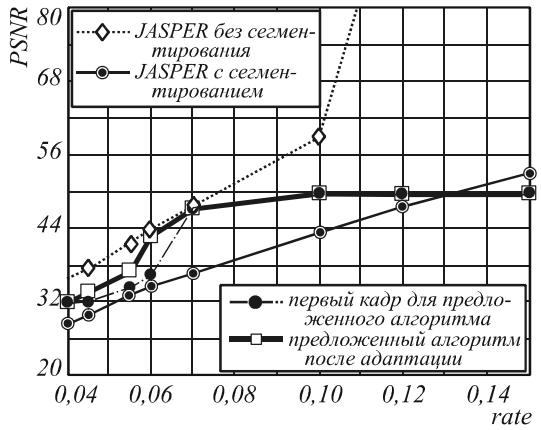

Рис. 9. Зависимость скорость/искажение для последовательности test2

На рис. 8-10, показаны результаты использования оригинального алгоритма JPEG2000 как с сегментированием, так и без сегментирования, а также предложенного алгоритма для первого кадра последовательности и для кадра после адаптации алгоритма к скорости канала. Объем сглаживающего буфера передатчика был выбран из расчета, что дополнительная задержка при передаче не будет превосходить величину, эквивалентную 0,15 кадра. Величина B_{empty} выбрана так, чтобы обеспечивать пиковое отношение сигнал/шум на сегмент не хуже 30 дБ.

На рис. 11 показана зависимость пикового отношения сигнал/шум от номера сегмента для видеопоследовательности, содержащей компьютерную графику. График наглядно демонстрирует, что при условии ограничений на память, использование алгоритма управления битовой скоростью, реализованном в JPEG2000 не подходит для сжатия изображений, содержащих компьютерную графику. Это связано с тем, что колебания битовых затрат на сегмент, при заданной среднеквадратической ошибке восстановления для изображений, содержащих компьютерную графику, существенно больше, чем для естественных изображений (тоже следует из рис. 7).

При постановке оптимизационной задачи (3) предполагалось, что статистические свойства кадров меняются незначительно. Если в процессе работы предполагается смена сцены так, что статистические свойства следующей сцены существенно отличаются от предыдущей, то предложенный алгоритм после адаптации к скорости канала выйдет на значение ошибки, соответствующей «наихудшему» кадру. Для избежания этого эффекта в предложенный алгоритм достаточно внести следующую модификацию. Если в течение целого кадра система находилась на шаге 1 и количество бит после передачи последнего сегмента равно нулю, то следует уменьшить оценку ошибки $\hat{d}(t+1) = \hat{d}(t) - \Delta d$.

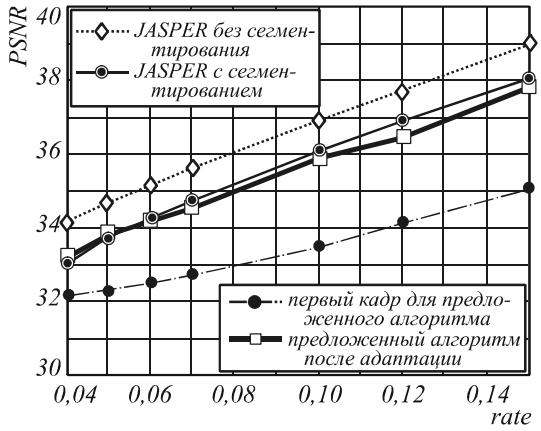

Рис. 10. Зависимость скорость/искажение для последовательности test3

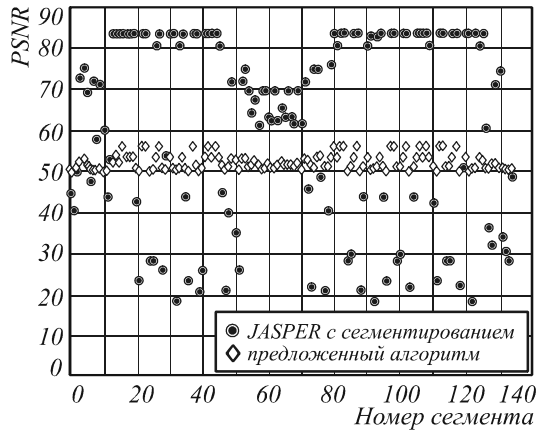

Рис. 11. Зависимость PSNR от номера сегмента для исходного алгоритма JPEG2000 с сегментированием и предложенного алгоритма для последовательности test1. Параметр rate=0,07

Заключение

В работе было показано (см. например рис. 8 и рис.11), что при условии ограничений на память, алгоритм управления битовой скоростью, используемый в алгоритме JPEG2000, не подходит для сжатия изображений, содержащих компьютерную графику. При таком управлении обеспечиваются постоянные битовые затраты на сегмент, но при этом могут существенно отличаться среднеквадратические ошибки для различных сегментов. Поэтому при передаче изображений, содержащих фрагменты компьютерной графики, необходимо управлять значением среднеквадратической ошибки на сегмент и использовать сглаживающий буфер. Применение данного подхода позволяет существенно повысить визуальное качество передаваемых видеопоследовательностей, содержащих компьютерную графику. При передаче последовательностей, содержащих только естественные изображения, данный подход показывает сравнимые с оригинальным алгоритмом результаты.

Литература

1. Marpe D., George V., Cycon H. L., and Barthel K. U. Performance Evaluation of Motion-JPEG2000 in Comparison with H.264/AVC Operated in Intra Coding Mode // Proc. SPIE, vol. 5266, pp. 129-137, 2004.
2. ITU-T and ISO/IEC JTC 1. JPEG 2000 Image Coding System: Core Coding System, ITU-T Recommendation T.800 and ISO/IEC 15444-1 // JPEG 2000 Part 1, 2000.
3. Mallat S. A Theory of Multiresolution Signal Decomposition: The Wavelet Representation // IEEE Transactions on Pattern Anal. Mach. Intell., 1989. – V. 11. – Pp. 674-693.
4. Antonini M., Barlaud M., Mathieu P., Daubechies I. Image coding using wavelet transform Image Processing // IEEE Transactions on Image Processing, 1992. – Vol. 1. Issue 2. – Pp. 205-220,
5. Taubman D. High Performance Scalable Image Compression with EBCOT // IEEE Transactions on Image Processing, 2000. – Vol. 9. – No. 7. – Pp. 1158–1170,
6. Jelinek F. Buffer overflow in variable-length coding of fixed rate sources // IEEE Trans. Inform. Theory, 1968. Vol. IT-14, pp. 490-501.
7. Reibman A.R., Haskell B.G. Constraints on variable bitrate video for ATM networks // IEEE Transactions on Circuits and Systems for Video Technology, 1992. – V. 2. – Issue 4. – Pp. 361-372.