АЛГОРИТМЫ ОЦЕНКИ ДВИЖЕНИЯ В ЗАДАЧАХ СЖАТИЯ ВИДЕОИНФОРМАЦИИ НА НИЗКИХ БИТОВЫХ СКОРОСТЯХ

Е.А. Беляев[1], А.М. Тюрликов[1]
[1]Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия

Аннотация

В работе представлено описание ряда известных алгоритмов оценки движения, используемых в задачах сжатия видеоинформации. Особое внимание уделяется алгоритмам оценки движения, которые оптимизируют битовые затраты на векторы движения и разностные блоки. Предложен модифицированный алгоритм иерархической оценки движения. Приведены результаты практического сравнения, показывающие эффективность алгоритма.

Ключевые слова: сжатие видеоинформации, алгоритмы оценки движения.

Введение

Для источников видеоинформации характерны *пространственная (внутрикадровая) и временная (межкадровая)* избыточности, которые проявляются во взаимной статистической прогнозируемости значений яркости и цветности соседних пикселов внутри кадра и значений яркости и цветности соответствующих пикселов соседних кадров. Устранение этой статистической прогнозируемости лежит в основе сжатия видеоинформации [1].

Основными характеристиками кодера видеоинформации являются битовая скорость (количество бит, формируемых кодером в единицу времени) и уровень искажения восстановленной видеопоследовательности относительно исходной. Цель системы сжатия видеоинформации состоит в минимизации искажения при заданном уровне битовой скорости.

В настоящей работе рассматриваются задачи устранения временной избыточности видеоинформации при помощи *блоковой компенсации движения,* при которой кадр сначала разбивается на прямоугольные блоки. Затем выполняется процедура *оценки движения*, при которой в текущем кадре для каждого блока выполняется поиск в базовом кадре наиболее “похожего” блока, смещенного на вектор, называемый *вектором движения*. В качестве критерия наилучшего вектора движения может использоваться, например, минимум суммы квадратов разностей значений яркости соответствующих пикселов блока в текущем кадре и найденного блока в базовом кадре.

После выполнения процедуры оценки движения путем вычитания соответствующих значений яркостей и цветностей пикселов текущего блока и найденного блока формируется разностный блок, который кодируется вместе с вектором движения.

При разработке алгоритмов оценки движения возникает задача уменьшения вычислительной сложности поиска вектора движения, а также задача минимизации битовых затрат на векторы движения.

Дальнейшее изложение организовано следующим образом. В разделе 1 описывается базовый кодер видеоинформации, который применяется для получения сравнительных результатов сжатия тестовых видеопоследовательностей. Разделы 2 и 3 систематизируют ранее известные результаты, на основе которых в разделе 4 предлагается новый алгоритм оценки движения. Остановимся более подробно на структуре данных разделов.

В разделе 2 описываются так называемые алгоритмы «быстрой» оценки движения, которые уменьшают вычислительную сложность поиска векторы движения путем сокращения исходного множества проверяемых векторов.

В разделе 3 рассматриваются вопросы, связанные с алгоритмами оценки движения, которые учитывают битовые затраты на векторы движения. Обычно, при использовании суммы квадратов разностей в качестве критерия поиска векторы движения исходят из предположения, что минимизация данного критерия приводит к минимизации суммарных битовых затрат на представление видеоинформации. Однако при высоких степенях сжатия данное предположение не выполняется, так как в суммарных битовых затратах существенно возрастает доля затрат на векторы движения.

Один из способов учесть векторы движения, описанный в подразделе 3.1, заключается в использовании критерия поиска, состоящего из линейной комбинации суммы квадратов разностей и битовых затрат на вектор движения. Однако такой критерий не гарантирует, что суммарные битовые затраты будут минимальны для полученного уровня искажения. Поэтому в подразделе 3.2 приводится постановка задачи оценки движения с ограничением. При такой постановке задачи каждому блоку необходимо сопоставить такой вектор движения и шаг квантования, что будет минимизирован уровень искажения на кадр с учетом ограничения на максимальный объем сжатого кадра.

Следует принять во внимание, что для обеспечения требуемой битовой скорости сжатой видеоинформации применяются *алгоритмы управления битовой скоростью*. Управление битовой скоростью осуществляется, как правило, путем изменения шага квантования. В связи с этим существуют два подхода к реализации алгоритмов оценки движения. При первом подходе алгоритм управления сначала задает шаг квантования, после чего алгоритм оценки движения формирует множество векторов движения. При втором подходе алгоритм управления за-

дает требуемое количество бит на сжатый кадр, а вектор движения и шаг квантования для каждого блока определяется алгоритмом оценки движения с ограничением.

В соответствии с этими двумя подходами, в разделе 4 предлагается модифицированный алгоритм иерархической оценки движения. Идея алгоритма для случая, когда шаг квантования определен алгоритмом управления битовой скоростью, рассматривается в подразделах 4.1 – 4.4 и состоит в следующем. Во-первых, используется процедура, при которой кадр сначала делится на квадратные блоки большого размера. Каждый такой блок, в свою очередь, может быть разбит на четыре квадратных блока и так далее. В дальнейшем такую процедуру будем называть *квадратичным разбиением* кадра. Во-вторых, используется известный «быстрый» алгоритм иерархической оценки движения, с помощью которого формируется множество вариантов кодирования большого блока: вариант кодирования с одним вектором движения, с четырьмя векторами и так далее. В-третьих, для выбора лучшего варианта кодирования используется рекурсивная процедура, которая минимизирует битовые затраты на векторы движения при условии, что битовые затраты на разностные блоки будут минимальны из множества вариантов кодирования. При этом лучший вариант кодирования будет определяться значением шага квантования.

Для случая оценки движения с ограничением предложенный алгоритм в подразделе 4.5 обобщается следующим образом. Для каждого большого блока и шага квантования по описанной выше схеме формируется множество лучших вариантов кодирования. Затем формулируется оптимизационная задача минимизации искажения при ограничении на максимальный объем сжатого кадра.

В разделе 5 приводятся сравнительные результаты сжатия тестовых видеопоследовательностей, которые демонстрируют эффективность предложенного алгоритма.

1. Базовый кодер видеоинформации

В качестве базового кодера видеоинформации в настоящей работе рассматривается наиболее распространённый стандарт сжатия MPEG-2 [2] (см. рис.1), а алгоритмы, входящие в базовый кодер, заимствованы из официальной открытой реализации стандарта [3].

В базовом кодере все кадры входной видеопоследовательности разбиваются на *макроблоки* размером 16×16 пикселов и подразделяются на три типа: I (Intra), P (Predicted) и B (Bidirectional).

1.1. Кодирование I-кадров в базовом кодере

I-кадры кодируются независимо от других кадров. Каждый макроблок разбивается на четыре блока по 8x8 пикселов. Для каждого блока выполняется дискретное косинусное преобразование (ДКП) и скалярное квантование коэффициентов преобразования. Ненулевые квантованные коэффициенты преобразования сжимаются без потерь при помощи кодирования длин серий с последующим кодированием кодом Хаффмана.

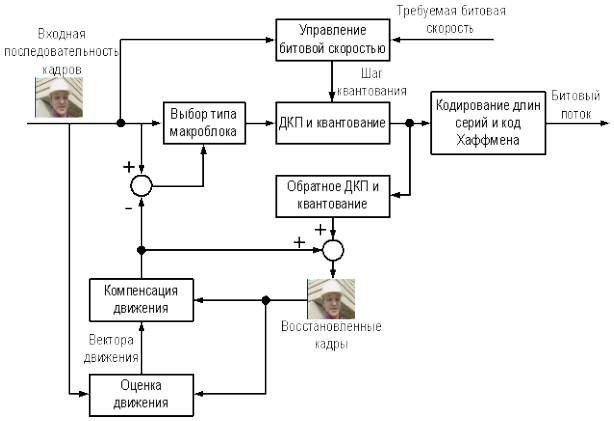


Рис.1. Схема базового кодера

1.2. Кодирование P-кадров в базовом кодере

Для кодирования P-кадров используется последовательность восстановленных кадров, которые будут получены на стороне декодера. Для получения восстановленного кадра на стороне кодера выполняется процедура обратного квантования и обратного ДКП.

Для кодирования P-кадра необходим *базовый кадр,* к которому относится предыдущий восстановленный, в порядке воспроизведения, I либо P-кадр. Для каждого макроблока S_n в P-кадре с номером n выполняется поиск блока S_k^* в базовом кадре с номером k, соответствующий минимуму следующего функционала:

$$J(\vec{v}) = \sum_{(x,y)\in S_n} \left| s_n(x,y) - s_k^*(x+v_x, y+v_y) \right|^p, \qquad (1)$$

где $\vec{v} = \{v_x, v_y\}$ - вектор движения, $|v_x| \le r$ и $|v_y| \le r$, где r - радиус поиска, $p \in \{1,2\}$ - параметр функционала, $s_n(x,y)$ - значение яркости пиксела с координатами x и y в макроблоке S_n, $s_k^*(x+v_x, y+v_y)$ - значение яркости пиксела с координатами $x+v_x$ и $y+v_y$ в макроблоке S_k^*.

После выполнения процедуры оценки движения путем вычитания значений яркостей и цветностей пикселов в макроблоке S_n и найденном блоке в базовом кадре S_k^* формируется разностный макроблок ΔS_n:

$$\Delta s_n(x,y,\vec{v}) = s_n(x,y) - s_k^*(x+v_x, y+v_y), \qquad (2)$$

который затем обрабатывается по той же схеме, что и макроблок в I-кадре.

Для восстановления P-кадра на стороне декодера необходима передача векторов движения. Как правило, векторы движения для соседних макроблоков статистически зависимы, поэтому кодируется не абсолютное значение компонент вектора v_x и v_y, а

разность компонент вектора текущего макроблока и предыдущего в порядке кодирования макроблока.

Вектор движения, соответствующий минимуму функционала (1), сначала определяется при помощи полного перебора допустимых значений компонент вектора v_x и v_y в радиусе поиска при условии, что данные компоненты принимают целочисленные значения. Поиск путем полного перебора обладает высокой вычислительной сложностью, так как для определения наилучшего вектора движения необходимо вычислить функционал (1) в количестве $(2\cdot r+1)^2$ раз.

После поиска в целом диапазоне значений выполняется уточнение вектора движения с точностью до ½ пиксела. Для этого путем интерполяции формируется три дополнительных базовых блока, смещенных относительно найденного блока S_k^* на ½ пиксела вправо, на ½ пиксела вниз и на ½ пиксела вправо и вниз. Затем из четырех векторов-кандидатов выбирается вектор, которому соответствует наименьшее значение функционала (1).

1.3. Кодирование B-кадров в базовом кодере

Для B-кадра с номером n существуют три способа формирования разностного макроблока (см. рис. 2).

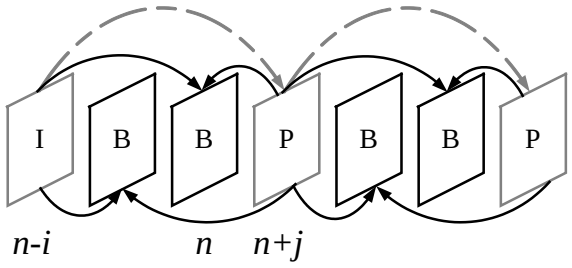


Рис.2. Пример положения базовых кадров для P-кадров (пунктирная линия) и B-кадров (сплошная линия)

В первом случае базовым кадром может быть только предыдущий в порядке воспроизведения восстановленный I либо P-кадр. Далее такой кадр будем называть *предыдущим базовым кадром* (номер $n-i$ на рис. 2).

Во втором случае базовым кадром может быть только следующий в порядке воспроизведения восстановленный I либо P-кадр. Далее такой кадр будем называть *следующим базовым кадром* (номер $n+j$ на рис.2).

При третьем способе формирования разностного макроблока используются как предыдущий, так и следующий базовые кадры:

$$\Delta s_n(x,y,\vec{v}^{\,f},\vec{v}^{\,b}) = s_n(x,y) - \left\lfloor \frac{s^*_{n-i}(x+v_x^f, y+v_y^f) + s^*_{n+j}(x+v_x^b, y+v_y^b)+1}{2} \right\rfloor \quad (3)$$

где $\vec{v}^{\,f} = \{v_x^f, v_y^f\}$ - вектор движения относительно предыдущего базового кадра, $\vec{v}^{\,b} = \{v_x^b, v_y^b\}$ - вектор движения относительно следующего базового кадра.

Из трех типов разностных макроблоков выбирается тот тип, для которого (1) принимает наименьшее значение с учетом того, что для третьего типа разностного макроблока (1) принимает вид:

$$J(\vec{v}^{\,f},\vec{v}^{\,b}) = \sum_{(x,y)\in S} \left|\Delta s_n(x,y,\vec{v}^{\,f},\vec{v}^{\,b})\right|^p. \quad (4)$$

Для минимизации (4) необходимо выполнить совместный поиск векторов $\vec{v}^{\,f}$ и $\vec{v}^{\,b}$. Поиск наилучшей пары векторов потребует вычисление функционала (4) в количестве $(2\cdot r+1)^4$ раз.

Для уменьшения вычислительной сложности в базовом кодере вместо совместного поиска пары векторов движения выполняется поиск векторов $\vec{v}^{\,f}$ и $\vec{v}^{\,b}$ по отдельности, а затем найденные векторы используются при формировании разностного блока (3) и при вычислении функционала (4).

2. «Быстрые» алгоритмы оценки движения

Как уже было отмечено выше, поиск полным перебором обладает высокой вычислительной сложностью. Поэтому возникает задача уменьшить количество вычислений при условии, что будет найден вектор движения, которому соответствует значение функционала, близкое к поиску, найденному полным перебором. Рассмотрим некоторые из известных алгоритмов «быстрой» оценки движения.

2.1. Логарифмический поиск

Одним из первых в качестве «быстрого» алгоритма оценки движения был предложен *логарифмический поиск* (см. рис. 3) в работе [4].

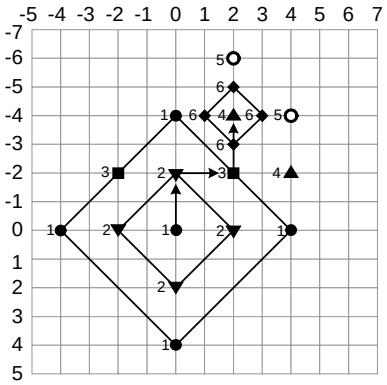


Рис.3. Логарифмический поиск, $v_x^0 = v_y^0 = 0$

Алгоритм состоит из трех шагов:

Шаг 1. $\vec{v} := (v_x^0, v_y^0), \vec{v}^* := \vec{v}$, $n := 4$, $J_{\min} := J(\vec{v})$, $\{\Delta\vec{v}\} := \{(0,n),(n,0),(0,-n),(-n,0)\}$.

Шаг 2. $\vec{v}^* := \vec{v} + \Delta\vec{v}_i$, где $i = \arg\min_{\vec{h}=\vec{v}+\Delta\vec{v}_i} J(\vec{h})$,

Если $J_{\min} > J(\vec{v}^*)$, то $J_{\min} := J(\vec{v}^*)$,

$\vec{v} := \vec{v}^*$ и перейти к Шагу 2.

Иначе перейти к Шагу 3.

Шаг 3. Если $n > 1$, то $n := n/2$,

$\{\Delta\vec{v}\} := \{(0,n),(n,0),(0,-n),(-n,0)\}$

и перейти к Шагу 2.

Иначе вектор движения $\vec{v}$ найден.

На рис. 3 цифрами обозначены векторы движения, для которых вычисляется функционал, при очередном попадании алгоритма на Шаг 2.

2.2. Градиентный поиск

Как правило, логарифмический поиск применяется при интенсивном движении объектов в видеопоследовательности. В случае, если движение не интенсивное, может быть использован *градиентный поиск* (см. рис. 4), предложенный в работе [5].

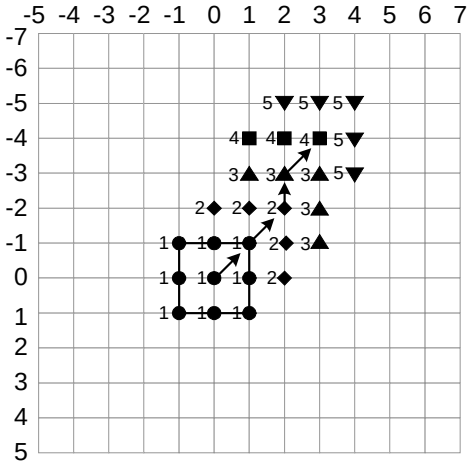


Рис.4. Градиентный поиск, $v_x^0 = v_y^0 = 0$

Алгоритм состоит из двух шагов:

Шаг 1.

$\vec{v} := (v_x^0, v_y^0), \vec{v}^* := \vec{v}, J_{\min} := J(\vec{v}),$

$\{\Delta\vec{v}\} := \{(0,1),(0,-1),(1,0),(-1,0),(1,1),(1,-1),(-1,-1),(-1,1)\}.$

Шаг 2. $\vec{v}^* := \vec{v} + \Delta\vec{v}_i$, где $i = \arg\min_{\vec{h}=\vec{v}+\Delta\vec{v}_i} J(\vec{h})$,

Если $J_{\min} > J(\vec{v}^*)$, то $J_{\min} := J(\vec{v}^*)$,

$\vec{v} := \vec{v}^*$ и перейти к Шагу 2.

Иначе вектор движения $\vec{v}$ найден.

2.3. «Быстрый» алгоритм для совместного поиска векторов движения

Предыдущие два алгоритма применяются для «быстрого» поиска одного вектора движения. Для поиска двух векторов движения, соответствующих минимуму (4), вместо совместного поиска полным перебором в работе [6] предлагается следующая итеративная процедура:

Шаг 1. Независимый поиск векторов $\vec{v}^f$ и $\vec{v}^b$ путем минимизации функционалов $J(\vec{v}^f)$ и $J(\vec{v}^b)$ соответственно.

Шаг 2. Минимизация $J(\vec{v}^f, \vec{v}^b)$ путем уточнения вектора $\vec{v}^f$ при фиксированном векторе $\vec{v}^b$.

Шаг 3. Минимизация $J(\vec{v}^f, \vec{v}^b)$ путем уточнения вектора $\vec{v}^b$ при фиксированном векторе $\vec{v}^f$.

Шаг 4. Если после выполнения Шага 2 и 3 величина $J(\vec{v}^f, \vec{v}^b)$ уменьшилась, то перейти к Шагу 2.

Иначе векторы движения $\vec{v}^f$ и $\vec{v}^b$ найдены.

Вычислительную сложность «быстрых» алгоритмов оценки движения точно определить затруднительно, поскольку количество вычислений функционала зависит от источника видеоинформации и заранее неизвестно. Однако эмпирические результаты [7] показывают, что для «быстрых» алгоритмов, схожих с логарифмическим и градиентным поиском, количество вычислений функционала при поиске вектора движения пропорционально радиусу поиска *r*.

3. Алгоритмы оценки движения, учитывающие битовые затраты на векторы движения

Как было уже отмечено выше, при использовании функционала (1) исходят из предположения, что минимизация (1) приводит к минимизации суммарных битовых затрат на представление видеоинформации.

Однако при высоких степенях сжатия в суммарных битовых затратах существенно возрастает доля векторов движения, которая может достигать 50% (см. рис. 5). Поэтому необходима разработка алгоритмов оценки движения, которые учитывают битовые затраты на векторы движения.

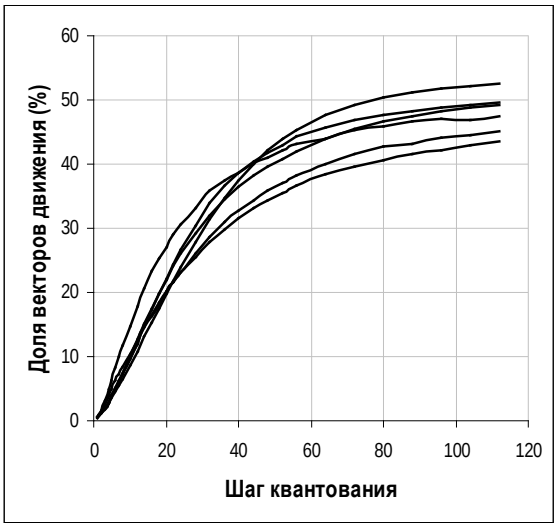


Рис.5. Доля битовых затрат на векторы движения в базовом кодере для шести тестовых видеопоследовательностей

3.1. Эвристический алгоритм, учитывающий битовые затраты на векторы движения

В работе [8] предлагается выполнять оценку движения последовательно для каждого макроблока путем минимизации следующего функционала:

$$\psi(\vec{v}, p) = J(\vec{v}) + \lambda r(\vec{v}), \qquad (5)$$

где $r(\vec{v})$ - битовые затраты на вектор движения, λ - множитель.

Для минимизации (5) необходимо знать значение λ. В работе [9] показано, что при некоторых допущениях величина λ может быть представлена в следующем виде:

$$\lambda = (c \cdot q^2)^{\frac{1}{2}p},$$

где q - шаг квантования, c - константа, которая принимает значение $c \approx 0{,}85$, p - параметр функционалов (1) и (4).

Преимуществом данного подхода является простота реализации, а также существенное улучшение эффективности кодирования на низких битовых скоростях по сравнению с методами, использующими функционалы (1) и (4). Однако следует заметить, что последовательная минимизация (5) в общем случае не приводит к общей минимизации битовых затрат как на векторы движения, так и на разностные блоки.

3.2. Постановка задачи оценки движения с ограничением

Вышеописанные алгоритмы оценки движения относятся к подходу, в рамках которого алгоритм управления битовой скоростью сначала задает шаг квантования, после чего алгоритм оценки движения формирует множество векторов движения.

Теперь рассмотрим случай, когда алгоритм управления задает максимально допустимое количество бит на сжатый кадр. При этом вектор движения и шаг квантования для каждого блока определяется алгоритмом оценки движения в соответствии с ограничением.

В работах [10-12] предложена постановка задачи блоковой оценки движения с ограничением. Пусть кодируемому кадру соответствует разбиение $f \in F$ на N_f блоков переменного размера $B_1,...,B_{N_f}$, где F - фиксированное множество разбиений. С данным разбиением f ассоциирован единственный обход блоков, который известен кодеру и декодеру. Каждому блоку B_i соответствует вектор движения $\vec{v}_i \in V_i$ и шаг квантования $q_i \in Q$, где V_i - конечное множество возможных векторов движения и Q - конечное множество возможных значений шага квантования. Для сокращения записи обозначим пару вектор движения и шаг квантования $\vartheta_i = \{\vec{v}_i, q_i\} \in \Theta_i$, где $\Theta_i = V_i \times Q$, битовые затраты и уровень искажения для блока B_i обозначим $r(\vartheta_i)$ и $d(\vartheta_i)$ соответственно.

Тогда, при ограничении количества бит на кадр, необходимо найти разбиение f и каждому блоку B_i сопоставить такое ϑ_i^f, чтобы:

$$\sum_{i=1}^{N_f} d(\vartheta_i^f) = \min_{j \in F} \min_{\{\vartheta_1^j,...,\vartheta_{N_j}^j\} \in \Theta_1^j \times ... \times \Theta_{N_j}^j} \sum_{i=1}^{N_j} d(\vartheta_i^j),$$

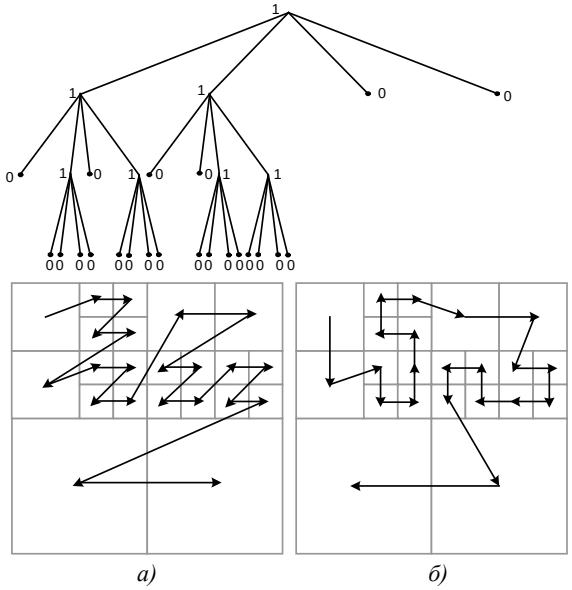
$$\text{при условии, что } \sum_{i=1}^{N_f} r(\vartheta_i^f) \le R_{\max}, \qquad (6)$$

где $R_{\max}$ - максимально допустимые битовые затраты на кодируемый кадр.

Оптимизационную задачу (6) можно решить при помощи динамического программирования, а также метода лагранжевых релаксаций (The Lagrangian relaxation method [13]). Однако вычислительная сложность такого решения чрезвычайно высока, так как потребуется перебор по множеству разбиений F, множеству векторов движения V_i и множеству шагов квантования Q.

3.3. Способы разбиения кадра на блоки переменного размера

Теперь рассмотрим способы формирования множества разбиений кадра F и соответствующие методы кодирования векторов движения. В работах [10-12] для изображения размером $2^L \times 2^L$ применяется квадратичное разбиение, представляющее кадр в виде блоков размера $2^l \times 2^l, (0 \le l \le L)$, где l - *уровень иерархии*. Для передачи разбиения декодеру используется четверичное иерархическое дерево (см. рис. 6), узлы которого принимают значения 0 (нет разбиения, используется блок размером $2^l \times 2^l$) и 1 (есть разбиение на четыре блока размером $2^{l-1} \times 2^{l-1}$). Дерево кодируется с корневой вершины. При этом если значение текущего узла равно 0, то дочерние узлы не кодируются.



а) *б)*

Рис.6. Четверичное иерархическое дерево и соответствующий растровый обход (а) и обход при помощи Гильбертовых кривых (б)

При оценке движения с переменным размером блока возникает задача обхода блоков так, чтобы предыдущий в порядке обхода вектор движения был максимально коррелирован с кодируемым вектором движения. При этом на передачу обхода не должна тратиться дополнительная информация. В связи с этим, в работах [10-12] прелагается использовать рекурсивно генерируемые кривые Гильберта вместо распространенного растрового обхода (см. рис. 6).

В работе [14] используется более гибкое разбиение, при котором кроме блоков размером $2^l \times 2^l$ возможно разбиение на блоки, например, $2^{l+1} \times 2^l$, $2^l \times 2^{l+1}$. При этом векторы движения, которые ис-

пользуются в качестве предсказателя, формируются из векторов, полученных для предыдущего кадра.

4. Модифицированный алгоритм иерархической оценки движения

В настоящей работе предлагается модифицированный алгоритм иерархической оценки движения, основанный на квадратичном разбиении, при котором кадр сначала делится на квадратные блоки большого размера (например, 128×128 пикселов), которые в дальнейшем будем называть *сегментами*. Затем каждый сегмент, в свою очередь, может быть поделен на четыре квадратных блока и так далее.

Алгоритм может быть использован как в режиме оценки движения при уже известном шаге квантования, так и в режиме оценки движения с ограничением.

Дальнейшее изложение организовано следующим образом. Вначале описывается режим использования оценки движения при уже известном шаге квантования. Для этого сначала описывается «быстрый» алгоритм иерархической оценки движения, схожий с алгоритмами в работах [15-16], с помощью которого формируется множество вариантов кодирования сегмента: вариант кодирования с одним вектором движения, с четырьмя векторами и так далее.

Затем описывается рекурсивный алгоритм разбиения Р-кадра на блоки переменного размера, который минимизирует битовые затраты на иерархическое дерево при условии, что битовые затраты на разностные блоки будут минимальны из множества затрат вариантов кодирования сегмента.

После этого описывается процедура кодирования векторов движения Р-кадра и показывается, что при таком способе кодирования рекурсивный алгоритм разбиения также минимизирует битовые затраты на векторы движения при условии, что битовые затраты на разностные блоки будут минимальны из множества вариантов кодирования сегмента.

Затем описываются особенности использования алгоритма для В-кадров.

В конце приводится обобщение, которое позволяет использовать предложенный алгоритм в режиме оценки движения с ограничением.

4.1. Иерархическая оценка векторов движения для Р-кадра

При иерархической оценке движения кодируемый и базовый кадры сначала уменьшаются в два раза по высоте и ширине. Полученные изображения снова уменьшаются и так далее. Всего такая процедура выполняется L раз, где L - параметр алгоритма. В результате формируются $L+1$ уровней иерархии, причем исходные (кодируемый и базовый) кадры соответствуют нулевому уровню.

Значение яркости пиксела на уровне l с координатами x и y вычисляется следующим образом:

$$s(l,x,y)=\left\lfloor \frac{1}{4}\left(\sum_{i=2\cdot x}^{2\cdot x+1}\sum_{j=2\cdot y}^{2\cdot y+1} s(l-1,i,j)+2\right)\right\rfloor.$$

Для описания алгоритма оценки движения введем следующие обозначения. Полученные на всех уровнях изображения разобьем на блоки одинакового размера (для базового кодера это блоки 16×16 пикселов). В результате такого деления блоку $B(l,i,j)$ с координатами (i,j) на уровне l соответствуют четыре блока на $l-1$ уровне и так далее. При этом сегментом будет являться совокупность блоков на нулевом уровне, соответствующих одному блоку $B(L,i,j)$.

Обозначим за $\vec{v}(l,i,j)$ вектор движения, соответствующий блоку с координатами (i,j) на уровне l. Вектор движения, который получен в результате использования градиентного поиска, обозначим за $g\{l,v_x^0,v_y^0\}$, где $\{v_x^0,v_y^0\}$ - начальное значение вектора движения, с которого начинается поиск, l - уровень иерархии, на котором выполняется поиск.

Тогда алгоритм иерархической оценки движения может быть описан при помощи трех шагов:

Шаг 1. Для всех блоков (i,j) на уровне L:

$\vec{v}(L,i,j):=g\{L,0,0\}$. $l:=L-1$.

Шаг 2. Для всех блоков (i,j) на уровне l:

$$\vec{v}_1^*:=g\{l,0,0\},$$

$$\vec{v}_2^*:=g\left\{l,2\cdot v_x\left(l+1,\left\lfloor\frac{i}{2}\right\rfloor,\left\lfloor\frac{j}{2}\right\rfloor\right),2\cdot v_y\left(l+1,\left\lfloor\frac{i}{2}\right\rfloor,\left\lfloor\frac{j}{2}\right\rfloor\right)\right\},$$

$$\vec{v}_3^*:=g\{l,v_x(l,i,j-1),v_y(l,i,j-1)\},$$

$$\vec{v}(l,i,j):=\arg\min_{\vec{h}=\{\vec{v}_1^*,\vec{v}_2^*,\vec{v}_3^*\}} J(\vec{h}).$$

Шаг 3. $l:=l-1$,

Если $l\ge0$ перейти к Шагу 2.

Иначе поиск окончен.

В результате формируется множество векторов $\{\vec{v}(l,i,j)\}$. Затем множество векторов движения $\{\vec{v}(0,i,j)\}$ уточняется до ½ пиксела по аналогии с уточнением в базовом кодере.

Из описания алгоритма иерархической оценки движения следует, что для 4^L макроблоков, входящих в сегмент, алгоритм градиентного поиска выполняется $1+3\cdot(4^1+4^2+...+4^L)=4^{L+1}-3$ раз. Следовательно, количество вычислений функционала при иерархической оценке движения в четыре раза больше, чем при градиентном поиске, и пропорционально $4\cdot r$.

4.2. Квадратичное разбиение Р-кадра

После выполнения алгоритма иерархической оценки движения выполняется квадратичное разбиение изображения на блоки переменного размера. При этом разбиение производится независимо для каждого сегмента. Для передачи разбиения сегмента аналогично [10-12] используется четверичное иерархическое дерево (см. рис. 6). Обозначим $t(l,i,j)$ значение в узле этого дерева на уровне l для блока (i,j).

Тогда для кодирования иерархического дерева для каждого сегмента (i, j) необходимо выполнить рекурсивный алгоритм $Tree(L, i, j)$, определенный следующим образом:

$Tree(l, i, j)$
  кодировать $t(l, i, j)$;
  если $t(l, i, j) = 1$, ***то***
    выполнить $Tree(l-1, 2\cdot i, 2\cdot j)$;
    выполнить $Tree(l-1, 2\cdot i, 2\cdot j+1)$;
    выполнить $Tree\ (l-1, 2\cdot i+1, 2\cdot j)$;
    выполнить $Tree(l-1, 2\cdot i+1, 2\cdot j+1)$; (7)

Теперь обозначим за $r_{diff}\left(\vec{v}(l, i, j), q\right)$ битовые затраты на разностные блоки, формируемые при помощи вектора $2^l\vec{v}(l, i, j)$ для соответствующих блоков на нулевом уровне с использованием шага квантования q.

Тогда для формирования четверичного иерархического дерева для каждого сегмента (i, j) необходимо выполнить рекурсивный алгоритм разбиения $Div\ (L, i, j, q)$, определенный следующим образом:

$Div\ (l, i, j, q)$
  $R := r_{diff}\left(\vec{v}(l, i, j), q\right)$;
  если $l = 0$ ***вернуть*** R;
  $R_{00} := Div\ (l-1, 2\cdot i, 2\cdot j, q)$;
  $R_{01} := Div\ (l-1, 2\cdot i, 2\cdot j+1, q)$;
  $R_{10} := Div\ (l-1, 2\cdot i+1, 2\cdot j, q)$;
  $R_{11} := Div\ (l-1, 2\cdot i+1, 2\cdot j+1, q)$; (8)
  если $R \le R_{00} + R_{01} + R_{10} + R_{11}$, ***то***
    $t(l, i, j) := 0$;
    вернуть R;
  иначе
    $t(l, i, j) := 1$;
    вернуть $R_{00} + R_{01} + R_{10} + R_{11}$;

Величина $r_{diff}\left(\vec{v}(l, i, j), q\right)$ используется в качестве критерия разбиения. Если количество бит, необходимое для передачи разностных блоков на нулевом уровне, при разбиении блока на текущем уровне уменьшается, то блок разбивается на четыре части. В противном случае блок не разбивается и для всех соответствующих блоков на нулевом уровне используется один вектор движения.

Таким образом, для фиксированного множества векторов $\{\vec{v}(l, i, j)\}$ вид разбиения будет определяться величиной шага квантования q (см. рис. 7-8).

Из алгоритма (7) следует, что если справедливо $t(l, i, j) = 0$, то значения иерархического дерева в дочерних узлах не кодируются. Из алгоритма (8) следует, что операция $t(l, i, j) := 0$ выполняется в случае, если количество бит, необходимое для передачи разностных блоков на нулевом уровне, при разбиении блока на текущем уровне не уменьшается. Поэтому из (7) следует, что алгоритм (8) минимизирует битовые затраты на иерархическое дерево при условии, что битовые затраты на разностные блоки будут минимальны из множества вариантов кодирования сегмента.

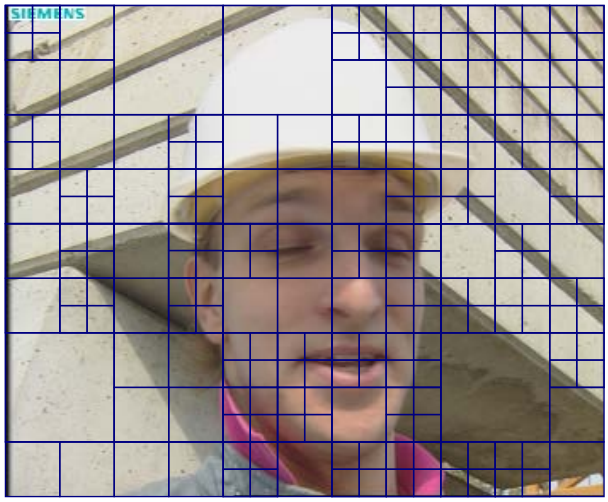


Рис.7. Пример иерархического разбиения при $q = 64$

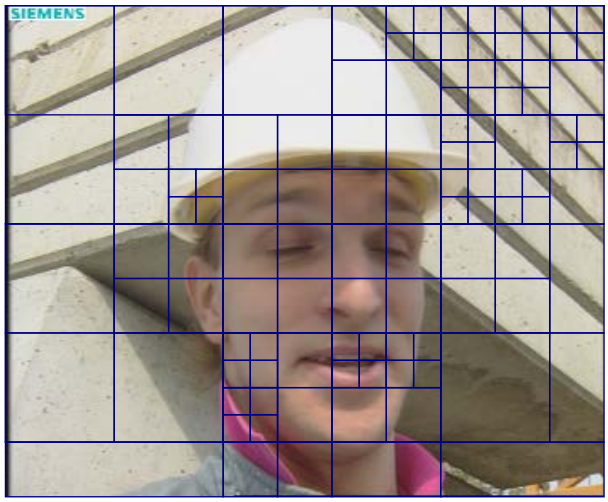


Рис.8. Пример иерархического разбиения при $q = 112$

4.3. Кодирование векторов движения P-кадра

При выполнении иерархической оценки движения на уровне l в качестве начального вектора движения, с которого начинается градиентный поиск, используется вектор из уровня $l+1$ (вектор $\vec{v}_2^*$). Поэтому вектор движения $\vec{v}(l, i, j)$ статистически зависим от вектора $\vec{v}\left(l+1, \lfloor i/2 \rfloor, \lfloor j/2 \rfloor\right)$.

В связи с этим в настоящей работе предлагается *иерархическое кодирование* векторов движения. При таком кодировании для каждого сегмента (i, j) необходимо выполнить рекурсивный алгоритм $Vect(L, i, j, 0, 0)$, определенный следующим образом:

$$
\begin{aligned}
&Vect(l,i,j,v_x^p,v_y^p)\\
&v_x^* := v_x(l,i,j);\\
&v_y^* := v_y(l,i,j);\\
&\textbf{кодировать } v_x^* - v_x^p;\\
&\textbf{кодировать } v_y^* - v_y^p;\\
&\textbf{если } t(l,i,j)=1, \textbf{ то}\\
&\quad \textbf{выполнить}\\
&\qquad Vect(l-1,2\cdot i,2\cdot j,2\cdot v_x^*,2\cdot v_y^*);\\
&\quad \textbf{выполнить}\\
&\qquad Vect(l-1,2\cdot i,2\cdot j+1,2\cdot v_x^*,2\cdot v_y^*);\\
&\quad \textbf{выполнить}\\
&\qquad Vect\ (l-1,2\cdot i+1,2\cdot j,2\cdot v_x^*,2\cdot v_y^*);\\
&\quad \textbf{выполнить}\\
&\qquad Vect(l-1,2\cdot i+1,2\cdot j+1,2\cdot v_x^*,2\cdot v_y^*);
\end{aligned}
\qquad (9)
$$

Из описания алгоритмов (8) и (9) следует, что алгоритм (8) также минимизирует битовые затраты на векторы движения при условии, что битовые затраты на разностные блоки будут минимальны из множества вариантов кодирования сегмента.

Сложность алгоритмов (7), (8) и (9) определяется сложностью вычисления $r_{diff}\left(\vec{v}(l,i,j),q\right)$, которое определяется $L+1$ раз для каждого макроблока. При этом выполняются процедуры:

- формирования разностного блока;
- дискретного косинусного преобразования;
- квантования;
- определения длин серий и использования кодовых таблиц для подсчета битовых затрат.

Можно показать, что количество операций при вычислении величины $r_{diff}\left(\vec{v}(l,i,j),q\right)$ сопоставимо с вычислением функционала (1) при p=2. В результате количество операций, выполняемых предложенным алгоритмом с учетом иерархической оценки движения, пропорционально $4r+L$, что несколько больше, чем при использовании «быстрых» алгоритмов и существенно меньше, чем при поиске полным перебором.

4.4. Оценка движения, разбиение и кодирование векторов движения для B-кадров

Особенность обработки B-кадра заключается в том, что для каждого сегмента алгоритмом (8) формируются три варианта разбиения. Отличие этих трех вариантов заключается в способе формирования разностных макроблоков. В первом варианте разностные блоки формируются согласно (2) с использованием предыдущего базового кадра. Во втором варианте разностные блоки формируются согласно (2) с использованием следующего базового кадра. В третьем варианте разностные блоки формируются согласно (3) с использованием предыдущего и следующего базовых кадров.

Затем из трех вариантов разбиений выбирается вариант, обеспечивающий минимум битовых затрат на разностные макроблоки.

4.5. Оценка движения с ограничением

Как уже было отмечено выше, предложенный алгоритм можно обобщить для случая оценки движения с ограничением. Заметим, что битовые затраты на векторы движения, иерархическое дерево и разностные блоки для сегмента i полностью определяются шагом квантования $q_i \in Q$. Обозначим $r_i(q)$ и $d_i(q)$ суммарные битовые затраты и уровень искажения на сегмент i.

Тогда, при ограничении количества бит на кадр для каждого сегмента i, необходимо выбрать шаг квантования q_i^*, чтобы

$$
\sum_{i=1}^{N} d(q_i^*) = \min_{\{q_1,\ldots,q_N\},\, q_i \in Q} \sum_{i=1}^{N} d(q_i)
$$
$$
\text{при условии, что } \sum_{i=1}^{N} r(q_i^*) \le R_{\max}. \qquad (10)
$$

Оптимизационную задачу (10) можно решить при помощи динамического программирования, а также метода лагранжевых релаксаций. При этом сложность решения этой задачи существенно меньше, чем для задачи (6). Это связано с тем, что в данном случае шаг квантования q определяет как иерархическое дерево, так и множество используемых векторов движения. Поэтому перебор по этим величинам не требуется.

Количество операций для обобщенного алгоритма возрастает за счет перебора по множеству значений шага квантования и пропорционально $4r+|Q|L$, где $|Q|$- число используемых шагов квантования. Отметим, что наиболее часто используемые значения радиуса поиска при оценке движения $16 \le r \le 64$, число уровней иерархии $2 \le L \le 5$, величина $|Q| \le 32$ для базового кодера. Поэтому сложность обобщенного алгоритма существенно меньше, чем при полном переборе. При этом обобщенный алгоритм не только минимизирует битовые затраты на векторы движения, но и обеспечивает заданное количество бит на кадр.

5. Практические результаты

Практические результаты были получены для известных тестовых видеопоследовательностей: «akiyo», «container», «foreman», «hall», «news», «silent» (см. рис. 9-14) длительностью 300 кадров, разрешением 352x288, с кадровой скоростью 30 кадров в секунду.

В качестве меры искажения использовалось пиковое отношение сигнал/шум:

$$
PSNR(d) = 10\log_{10}\left(\frac{255^2}{d}\right),
$$

где d- среднеквадратическая ошибка,

$$
d = \frac{1}{K\cdot W\cdot H}\sum_{k\in K}\sum_{(x,y)\in k}\left(s_k(x,y)-s_k^*(x,y)\right)^2,
$$

где H, W - высота и ширина кадра в пикселах, K - число кадров в видеопоследовательности, $s_k(x,y)$, $s_k^*(x,y)$ - значения яркостей пикселов с координатами x и y в исходном и восстановленном кадре с номером k соответственно.

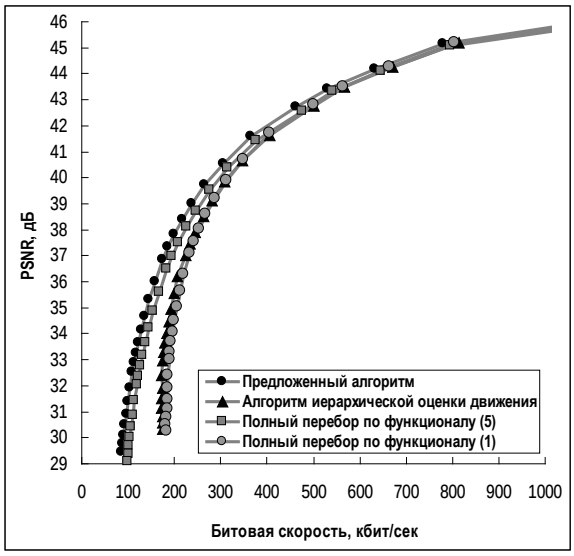


Рис.9. Результаты сравнения для «akiyo»

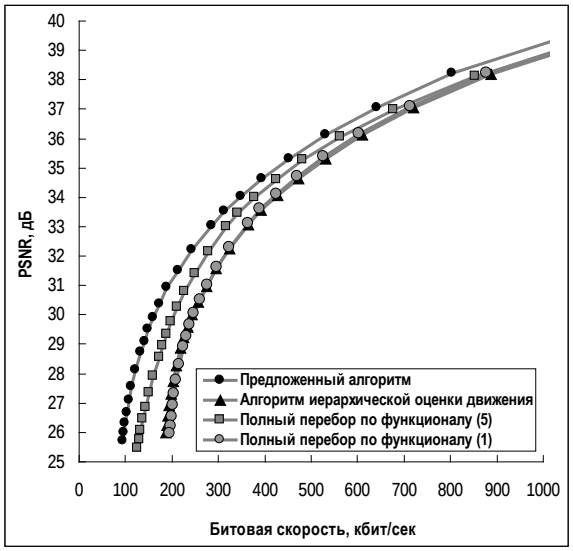


Рис.10. Результаты сравнения для «container»

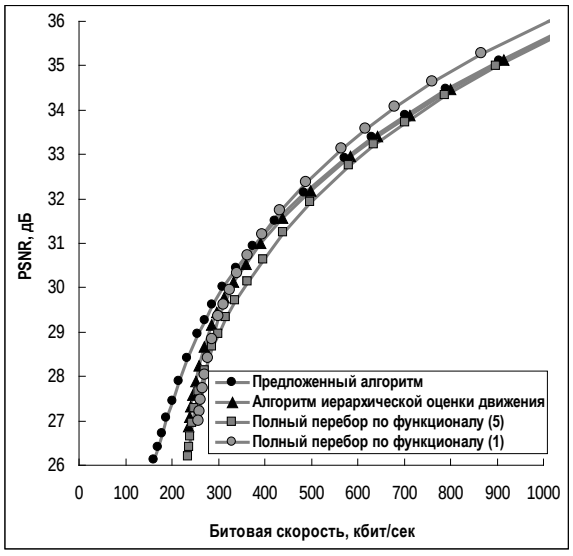


Рис.11. Результаты сравнения для «foreman»

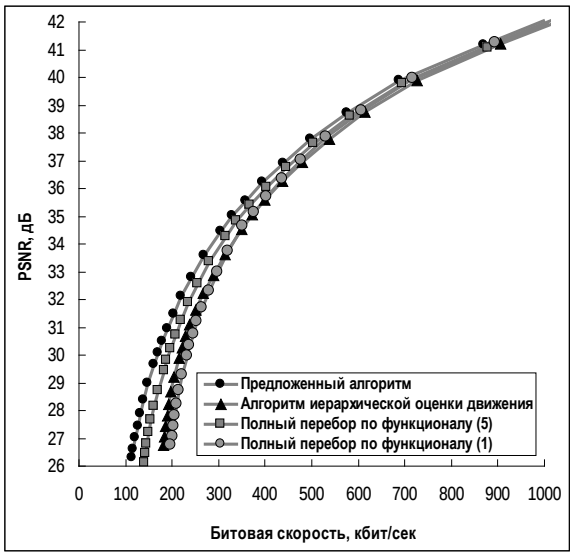


Рис.12. Результаты сравнения для «hall»

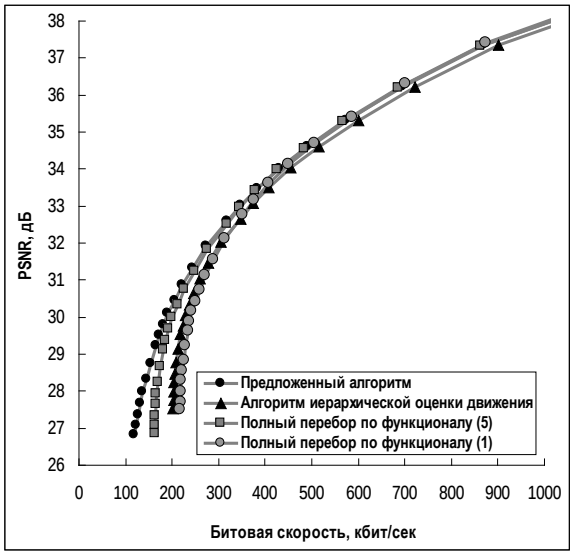


Рис.13. Результаты сравнения для «news»

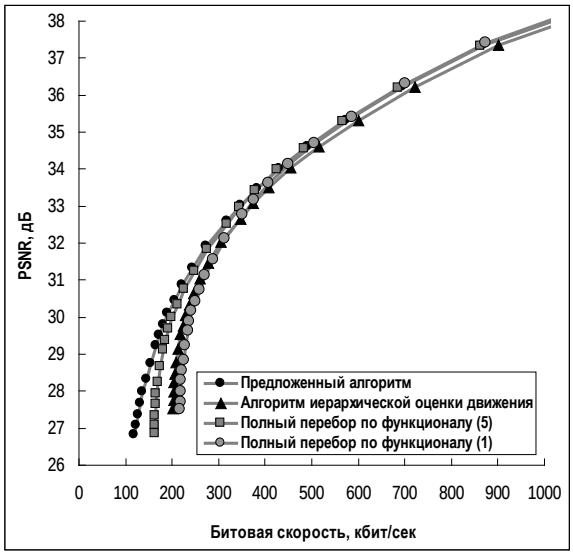


Рис.14. Результаты сравнения для «silent»

Результаты были получены для базового кодера, использующего:

- оценку движения полным перебором по функционалу (1);
- оценку движения полным перебором по функционалу (5);
- алгоритм иерархической оценки движения;
- предложенный модифицированный алгоритм иерархической оценки движения.

При этом сравнение проводилось для случая, когда шаг квантования задан до выполнения оценки движения.

Результаты сравнения демонстрируют высокую эффективность предложенного алгоритма, который позволяет на низких битовых скоростях на 5-40% сократить битовую скорость видеопоследовательности при фиксированном значении пикового отношения сигнал/шум. При этом сложность алгоритма существенно меньше, чем при поиске полным перебором.

Литература

1. **Оппенгейм, Э.** Применение цифровой обработки сигналов, /Э. Оппенгейм; пер. с англ.-.М.:Мир, 1980. – 552 с. (A.V. Oppenheim. Applications of Digital Signal Processing. Massachusetts Institute of Technology Cambridge, Mass. 1978)
2. **ISO/IEC** 13818 (MPEG-2): Generic coding of moving pictures and associated audio information, Nov. 1994.
3. http://www.mpeg.org/mpeg/
4. **Jain, J.R.** Displacement Measurement and Its Application in Interframe Image Coding / J.R. Jain, A.K. Jain // IEEE Transactions on Communications, 1981, vol. 29, No. 12, pp.1799-1808.
5. **Lurng-Kuo, L.** A block-based gradient de-scent search algorithm for block motion estimation in video coding / L. Lurng-Kuo, E. Feig // IEEE Transactions on Circuits and Systems for Video Technology, 1996, vol.6, pp.419-422.
6. **Wu, Siu-Wai.** Joint Estimation of Forward and Backward Motion Vectors for Interpolative Prediction of Video/ Siu-Wai Wu, A. Gersho // IEEE Transactions on Image Processing, 1994, vol. 3, No. 5, pp.684-687.
7. **Zhu, C.** A novel hexagon-based search algorithm for fast block motion estimation/ C. Zhu [and other] // 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001, vol. 3, pp.1593-1596.
8. **Kossentini, F.** Predictive RD Optimized Motion Estimation for Very Low Bit-Rate Video Coding/ F. Kossentini [and other] // IEEE Journal On Selected Areas in Communications, 1997, vol. 15, No. 9, p.1752-1763.
9. **Wiegard, T.** Lagrange multiplier selection in hybrid video coder control/ T.Wiegard, B. Girod // 2001 International Conference on Image Processing, 2001, vol. 3, pp. 542-545.
10. **Schuster, G. M.** A Theory for the Optimal Bit Allocation Between Displacement Vector Field and Displaced Frame Difference/ G. M. Schuster, A.K. Katsaggelos // IEEE Journal On Selected Areas in Communications, 1997, vol. 15, No. 9, pp.1739-1751.
11. **Schuster, G. M.** A Video Compression Scheme with Optimal Bit Allocation Among Segmentation, Motion, and Residual Error/ G. M. Schuster, A.K. Katsaggelos // IEEE Transactions On Image Processing, 1997, vol.6, No. 11, pp.1487-1502.
12. **Schuster, G. M.** An Optimal Quadtree-Based Motion Estimation and Motion-Compensated Interpolation Scheme for Video Compression / G. M. Schuster, A.K. Katsaggelos // IEEE Transactions On Image Processing, 1998, vol. 6, No. 11, pp.1505-1523.
13. **Fisher, M. L.** The Lagrangian relaxation method for solving integer programming problems/ M. L. Fisher // Management Sci., 1981, vol. 27, pp. 1–18.
14. **Injong, R.** Quadtree-Structured Variable-Size Block-Matching Motion Estimation with Minimal Error / R. Injong [and other] // IEEE Tranactions On Circuits And Systems for Video Technology, 2000, vol. 10, No. 1, pp. 42-50.
15. **Accame, M.** Hierarchical motion estimator (HME) for block-based video coders/ M. Accame [and other] // IEEE Transactions on Consumer Electronics, 1997, vol.43, pp.1320-1330.
16. **Lopes, F.** Hierarchical motion estimation with spatial transforms/ F. Lopes, M. Ghanbari // 2000 International Conference on Image Processing, 2000, vol.2, pp.558-561.

MOTION ESTIMATION ALGORITHMS FOR LOW BIT-RATE VIDEO COMPRESSION

E.A. Belyaev[1], A.M. Turlikov[1]
[1]Saint-Petersburg State University of Aerospace Instrumentation, Saint-Petersburg, Russia

Abstract

Several known motion estimation algorithms for video compression are described. A particular attention is given to motion estimation algorithms which minimize the bit-rate of motion vectors and inter frame blocks. A modified algorithm of hierarchical motion estimation is proposed. Comparison results which show the practical efficiency of the proposed algorithm are presented.

Key words: video compression, motion estimation algorithms.

Citation: Belyaev EA, Turlikov AM. Motion estimation algorithms for low bit-rate video compression. Computer Optics 2008; 32(4): 403-12.

References

[1] Oppenheim AV. Applications of digital signal processing [Russian translation]. Moscow: “Mir” Publisher 1980; 552 p.
[2] ISO/IEC 13818 (MPEG-2): Generic coding of moving pictures and associated audio information, Nov. 1994.
[3] http://www.mpeg.org/mpeg/
[4] Jain JR, Jain AK. Displacement Measurement and Its Application in Interframe Image Coding. IEEE Transactions on Communications 1981; 29(12): 1799-1808.
[5] Lurng-Kuo L, Feig E. A block-based gradient de-scent search algorithm for block motion estimation in video coding. IEEE Transactions on Circuits and Systems for Video Technology 1996; 6: 419-422.
[6] Wu Siu-Wai, Gersho A. Joint Estimation of Forward and Backward Motion Vectors for Interpolative Prediction of Video. IEEE Transactions on Image Processing 1994; 3(5): 684-687.
[7] Zhu C. A novel hexagon-based search algorithm for fast block motion estimation. IEEE International Conference on Acoustics, Speech, and Signal Processing 2001; 3: 1593-1596.
[8] Kossentini F. Predictive RD Optimized Motion Estimation for Very Low Bit-Rate Video Coding. IEEE Journal On Selected Areas in Communications 1997; 15(9): 1752-1763.
[9] Wiegard T, Girod B. Lagrange multiplier selection in hybrid video coder control. International Conference on Image Processing 2001; 3: 542-545.
[10] Schuster GM, Katsaggelos AK. A Theory for the Optimal Bit Allocation Between Displacement Vector Field and Displaced Frame Difference. IEEE Journal On Selected Areas in Communications 1997; 15(9): 1739- 1751.
[11] Schuster GM, Katsaggelos AK. A Video Compression Scheme with Optimal Bit Allocation Among Segmentation, Motion, and Residual Error. IEEE Transactions On Image Processing 1997; 6(11): 1487-1502.
[12] Schuster GM, Katsaggelos AK. An Optimal Quadtree-Based Motion Estimation and Motion-Compensated Interpolation Scheme for Video Compression. IEEE Transactions On Image Processing 1998; 6(11): 1505-1523.
[13] Fisher ML. The Lagrangian relaxation method for solving integer programming problems. Management Sci. 1981; 27: 1–18.
[14] Injong R. Quadtree-Structured Variable-Size BlockMatching Motion Estimation with Minimal Error. IEEE Tranactions On Circuits And Systems for Video Technology 2000; 10(1): 42-50.
[15] Accame M. Hierarchical motion estimator (HME) for block-based video coders. IEEE Transactions on Consumer Electronics 1997; 43: 1320-1330.
[16] Lopes F, Ghanbari M. Hierarchical motion estimation with spatial transforms. International Conference on Image Processing 2000; 2: 558-561.