Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра сетей и устройств телекоммуникаций

АЛГОРИТМЫ СЖАТИЯ ПОЛУТОНОВЫХ ИЗОБРАЖЕНИЙ

МЕТОДИЧЕСКОЕ ПОСОБИЕ
по курсу
«Цифровая обработка речи и изображений»
для студентов специальности
«Сети телекоммуникаций»
всех форм обучения

Минск БГУИР 2013

УДК 004.627(076)
ББК 32.973.26-018.2я73
А45

А в т о р ы:
А. А. Борискевич, В. Ю. Цветков, П. Л. Полещук, И. А. Борискевич

Р е ц е н з е н т:

декан факультета непрерывного и дистанционного обучения учреждения образования «Белорусский государственный университет информатики и радиоэлектроники», кандидат технических наук, доцент В. М. Бондарик

Алгоритмы сжатия полутоновых изображений : метод. пособие
А45 по курсу «Цифровая обработка речи и изображений» для студ. спец. «Сети телекоммуникаций» всех форм обуч. / А. А. Борискевич [и др.]. – Минск : БГУИР, 2013. – 38 с. : ил.

ISBN 978-985-488-801-9.

Рассмотрены алгоритмы сжатия полутоновых изображений на основе вложенного кодирования древовидных структур вейвлет-коэффициентов. Приведено лабораторное задание по исследованию характеристик вейвлет-сжатия полутоновых изображений с потерями и без потерь.

Может быть использовано при курсовом и дипломном проектировании.

УДК 004.627(076)
ББК 32.973.26-018.2я73

ISBN 978-985-488-801-9

Содержание

Введение 4
1 Алгоритмы сжатия изображений в вейвлет-области 5
1.1 Обобщенная структурная схема алгоритма сжатия с потерями и без потерь 5
1.2 Дискретное вейвлет-преобразование 6
1.2.1 Дискретное вейвлет-преобразование на основе цифровой фильтрации 6
1.2.2 Дискретное вейвлет-преобразование на основе лифтинг-схемы 7
1.2.3 Расширение последовательности коэффициентов 15
1.2.4 Двумерное лифтинг вейвлет-преобразование 17
1.3 Компрессия коэффициентов вейвлет-матрицы изображения 19
1.3.1 Компрессия на основе учета межсубполосной избыточности 20
1.3.2 Компрессия на основе учета внутриполосной избыточности 24
1.3.3 Оценка эффективности алгоритма сжатия полутонового изображения 35
2 Лабораторное задание 36
3 Содержание отчета 36
4 Контрольные вопросы и задания 36
Литература 37

Введение

Сжатие графических данных – это эффективный способ повышения производительности систем передачи мультимедийной информации, позволяющий увеличить скорости передачи и распределения, а также уменьшить затраты памяти за счет сокращения объема данных.

В связи с ростом мультимедийного трафика в сетях телекоммуникаций, системах дистанционного обучения, видеонаблюдения технологических процессов значительное внимание уделяется разработке новых методов сжатия графических данных, позволяющих получить приемлемое качество изображения при минимальном битовом потоке. Основу этих методов составляет сжатие неподвижных полутоновых изображений, поскольку они являются компонентами цветных неподвижных и подвижных изображений. Сжатие изображений заключается в удалении или сокращении их избыточности.

Перспективным на настоящий момент является прогрессивное сжатие с потерями, которое позволяет получить высокие коэффициенты сжатия за счет уменьшения визуальной избыточности при малом искажении исходного изображения. Прогрессивное сжатие обеспечивает восстановление изображения по части принятых данных. Таким образом, приняв часть информации, можно с определенным искажением восстановить все изображение.

1 АЛГОРИТМЫ СЖАТИЯ ИЗОБРАЖЕНИЙ В ВЕЙВЛЕТ-ОБЛАСТИ

1.1 Обобщенная структурная схема алгоритма сжатия с потерями и без потерь

Задачей кодека изображений является компактное представление (сжатие) данных, содержащихся в изображении, и их восстановление из компактной формы без потерь или с заданными потерями в качестве.

Термин «сжатие» означает уменьшение объема данных, используемого для представления изображения. Избыточность является центральным понятием при компрессии данных любого типа. При сжатии изображений различают три основных вида избыточности данных: кодовая избыточность, межэлементная избыточность и визуальная избыточность. Сжатие данных достигается в том случае, когда сокращается или устраняется избыточность одного или нескольких видов.

Если изображение кодируется некоторым способом, требующим большего числа символов чем это необходимо, то говорят, что изображение имеет кодовую избыточность. Кодовая избыточность возникает всегда, когда при выборе кодовых слов, присваиваемых некоторым событиям, знания о вероятности событий не используются в полной мере.

Поскольку значения элементов изображения могут быть достаточно точно предсказаны по значениям их соседей, то информация, содержащаяся в отдельном элементе, оказывается относительно малой. Для отражения подобной межэлементной связи были введены различные термины, такие, как пространственная избыточность, геометрическая избыточность.

Визуальная избыточность принципиально отличается от других видов. В отличие от кодовой или межэлементной визуальная избыточность связана с реальной информацией, содержащейся в изображении. Ее удаление возможно лишь постольку, поскольку такая информация не является существенной (не воспринимается) при обычном визуальном восприятии. Уменьшение данного вида избыточности характерно для сжатия с потерями.

С математической точки зрения сжатие изображения равнозначно преобразованию некоторого двумерного массива данных в статистически некоррелированный массив.

Важным свойством современного кодека полутонового изображения является прогрессивность. Прогрессивность заключается в возможности восстановления образа изображения, имеющего меньшие размеры (масштабирование по разрешению) или более низкое качество (масштабирование по качеству), по части сжатых данных. Используя часть сжатых данных, можно получить некоторую промежуточную форму изображения, которая будет приближаться к исходному пропорционально объему используемых для восстановления данных.

Общая структурная схема вейвлет-кодека для прогрессивного сжатия полутоновых изображений, рассматриваемого в лабораторной работе, представлена на рисунке 1.1.

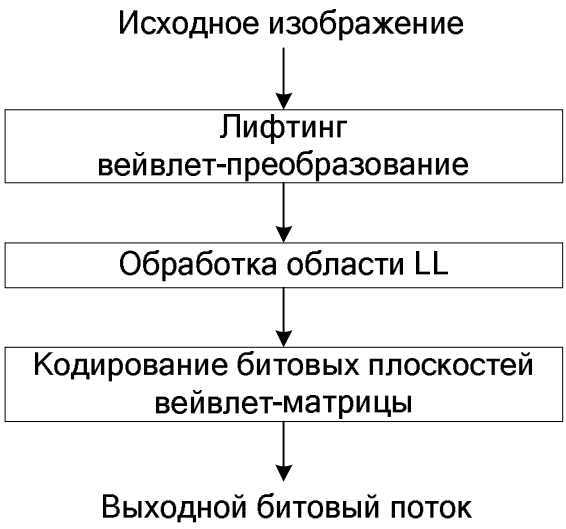


Рисунок 1.1 – Общая структура вейвлет-кодека для прогрессивного сжатия полутоновых изображений

Для уменьшения (устранения) пространственной избыточности, в рассматриваемом кодеке используется наиболее быстрый алгоритм дискретного вейвлет-преобразования – лифтинг вейвлет-преобразование.

Результатом лифтинг вейвлет-преобразования является вейвлет-матрица исходного изображения.

Вейвлет-матрица состоит из областей, разлилчных по значимости. Это дает возможность прогрессивного восстановления изображения из его вейвлет-матрицы.

Низкочастотная область вейвлет-матрицы (область LL) схожа по своим статистическим характеристикам с исходным изображением и имеет только положительные коэффициенты. Для оптимального применения алгоритмов компрессии вейвлет-матрицы требуется дополнительная обработка области LL, заключающаяся в представлении низкочастотных коэффициентов как положительными, так и отрицательными числами, уменьшая тем самым динамический диапазон вейвлет-матрицы.

Для компрессии вейвлет-матрицы используются прогрессивные алгоритмы кодирования битовых плоскостей.

1.2 Дискретное вейвлет-преобразование

1.2.1 Дискретное вейвлет-преобразование на основе цифровой фильтрации

Дискретное вейвлет-преобразование (ДВП) может быть представлено как фильтрация исходного сигнала $x(n)$ при $n \in (0, T-1)$, где T – количество отсче-

тов исходного сигнала, двумя фильтрами – низкочастотным ($\tilde{h}$) и высокочастотным ($\tilde{g}$) с последующим прореживанием отсчетов выходных потоков.

В результате образуется два сигнала – низкочастотный $s(n)$ и высокочастотный $d(n)$, имеющих длину вдвое меньшую, чем длина исходного сигнала. Обратное дискретное вейвлет-преобразование представляется как дополнение низкочастотного и высокочастотного сигнала нулями, фильтрация низкочастотного и высокочастотного сигналов фильтрами h и g соответственно и последующее суммирование сигналов.

Фильтры $\tilde{h}$ и $\tilde{g}$ образуют банк фильтров анализа, а фильтры h и g – банк фильтров синтеза.

Зная импульсную характеристику фильтров анализа $\tilde{h}$ и $\tilde{g}$, значения отсчетов сигналов $s(n)$ и $d(n)$ вычисляем по следующим формулам:

$$s(n) = \sum_{j=0}^{\tau_L - 1} \tilde{h}(j) x(2n - j), \tag{1.1}$$

$$d(n) = \sum_{j=0}^{\tau_H - 1} \tilde{g}(j) x(2n - j),\ n \in (0, \frac{T}{2}), \tag{1.2}$$

где τ_L – длина низкочастотного фильтра;

τ_H – длина высокочастотного фильтра.

При многомасштабном вейвлет-анализе для получения следующего уровня разложения исходного сигнала ДВП применяется к низкочастотному сигналу.

Разложение можно повторить несколько раз для дальнейшего увеличения частотного разрешения с дальнейшим прореживанием коэффициентов после низкочастотной и высокочастотной фильтрации. Таким образом, применяя рекурсивно банк фильтров к низкочастотному сигналу, полученному на предыдущем шаге преобразования, получим разложение сигнала на разных масштабах.

Так как на каждом этапе разложения длина низкочастотного сигнала уменьшается вдвое, то максимальное количество уровней разложения $R_{\max}$ составит

$$R_{\max} = \log_2(T). \tag{1.3}$$

Так как $R_{\max}$ – целое число, то длина сигнала должна быть равной степени числа 2.

1.2.2 Дискретное вейвлет-преобразование на основе лифтинг-схемы

Классическое ДВП на основе фильтрации имеет ряд недостатков. Одним из них является высокая вычислительная сложность. Причина высокой вычислительной сложности классического подхода заключается в том, что в процессе фильтрации обрабатывается весь сигнал, а затем прореживается. Таким обра-

зом, обработанными оказываются и те отсчеты, которые далее удаляются из сигнала. Для того чтобы снизить количество операций, требуемое для преобразования, было предложено лифтинг вейвлет-преобразование (ЛВП).

Чтобы перейти от классического ДВП к лифтиг-схеме, для банка фильтров, используемого в (1.1) и (1.2), задается следующее условие наилучшего восстановления сигнала:

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2, \qquad (1.4)$$
$$h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0,$$

где $h(z)$, $\widetilde{h}(z)$, $g(z)$, $\widetilde{g}(z)$ – z-преобразование соответствующих коэффициентов импульсных характеристик (КИХ) фильтров.

При выполнении условия (1.4) банк фильтров позволяет абсолютно точно восстановить первоначальный сигнал из частотных поддиапазонов, полученных в ходе преобразования.

Данные фильтры можно представить в виде полиномов в z-области. Например, фильтр h можно представить следующим образом:

$$h(z) = \sum_{i=0}^{p} h_i z^{-i}, \qquad (1.5)$$

где p – степень полинома.

В полифазном представлении фильтр h можно представить как

$$h(z) = h_e(z^2) + z^{-1}h_o(z^2), \qquad (1.6)$$

где h_e – четные отсчеты импульсной характеристики фильтра;

h_o – нечетные отсчеты импульсной характеристики фильтра.

Таким же образом в полифазном виде можно представить и остальные фильтры:

$$g(z) = g_e(z^2) + z^{-1}g_o(z^2), \qquad (1.7)$$
$$\tilde{h}(z) = \tilde{h}_e(z^2) + z^{-1}\tilde{h}_o(z^2),$$
$$\tilde{g}(z) = \tilde{g}_e(z^2) + z^{-1}\tilde{g}_o(z^2).$$

Основываясь на формулах (1.7), определим полифазные матрицы:

$$\tilde{P}(z) = \begin{pmatrix} \tilde{h}_e(z) & \tilde{h}_o(z) \\ \tilde{g}_e(z) & \tilde{g}_o(z) \end{pmatrix}. \qquad (1.8)$$

В матричной форме условие (1.4) будет выглядеть следующим образом:

$$P(z)\tilde{P}(z^{-1})^T = I, \qquad (1.9)$$

где I – единичная матрица размером 2×2.

Теперь ДВП можно представить в виде умножения на полифазную матрицу:

$$\begin{pmatrix} s(z) \\ d(z) \end{pmatrix} = \tilde{P}(z) \begin{pmatrix} x_e(z) \\ z^{-1}x_0(z) \end{pmatrix}, \qquad (1.10)$$

$$\begin{pmatrix} x_e(z) \\ z^{-1}x_0(z) \end{pmatrix} = P(z) \begin{pmatrix} s(z) \\ d(z) \end{pmatrix}.$$

Если детерминант $P(z)$ равен единице, то, применяя правило Крамера, получим

$$\tilde{h}(z) = -z^{-1}g(-z^{-1}), \qquad (1.11)$$
$$\tilde{g}(z) = z^{-1}h(-z^{-1}).$$

Следовательно,

$$h(z) = -z^{-1}\tilde{g}(-z^{-1}), \qquad (1.12)$$
$$g(z) = z^{-1}\tilde{h}(-z^{-1}).$$

Если детерминант $P(z)$ является единичным, то пара фильтров синтеза (h,g) и пара фильтров анализа $(\tilde{h},\tilde{g})$ являются комплементарными. Если $(h,g)=(\tilde{h},\tilde{g})$, то вейвлет-преобразование называется ортогональным, иначе – биортогональным.

Из (1.6), (1.7) следует, что если $(\tilde{h},\tilde{g})$ – комплементарная пара фильтров, то можно применить алгоритм Евклида для факторизации матрицы $\tilde{P}(z)$ в произведение конечного числа треугольных матриц:

$$\tilde{P}(z) = \left\{ \prod_{i=1}^{m} \begin{pmatrix} 1 & \tilde{q}_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tilde{t}_i(z) & 1 \end{pmatrix} \right\} \begin{pmatrix} K & 0 \\ 0 & \dfrac{1}{K} \end{pmatrix}, \qquad (1.13)$$

где K – масштабирующий множитель;

$\tilde{q}_i(z)$, $\tilde{t}_i(z)$ – полиномы в z-области более низких порядков.

Аналогичным образом можно найти $P(z)$:

$$P(z) = \left\{ \prod_{i=1}^{m} \begin{pmatrix} 1 & 0 \\ -t_i(z^{-1}) & 1 \end{pmatrix} \begin{pmatrix} 1 & -q_i(z^{-1}) \\ 0 & 1 \end{pmatrix} \right\} \begin{pmatrix} \dfrac{1}{K} & 0 \\ 0 & K \end{pmatrix}. \qquad (1.14)$$

Умножение исходного сигнала на отдельные треугольные матрицы можно представить как шаги лифтинга. Прямое ЛВП представлено на рисунке 1.2.

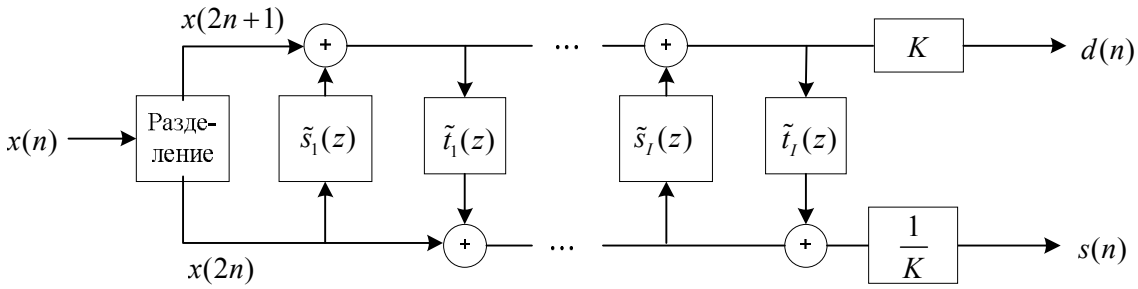


Рисунок 1.2 – Прямое лифтинг вейвлет-преобразование

Алгоритм обратного ЛВП полностью симметричен прямому. Обратное ЛВП представлено на рисунке 1.3.

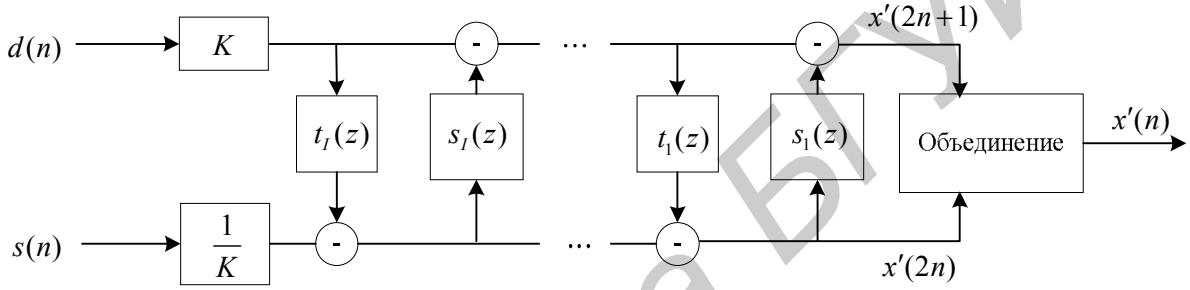


Рисунок 1.3 – Обратное лифтинг вейвлет-преобразование

Таким образом, лифтинг вейвлет-преобразование, основанное на факторизации матрицы, состоит из следующих этапов:

1) разложение отсчетов исходного сигнала на четные и нечетные. Данный шаг иногда называют Лэйзи вейвлет-преобразованием;

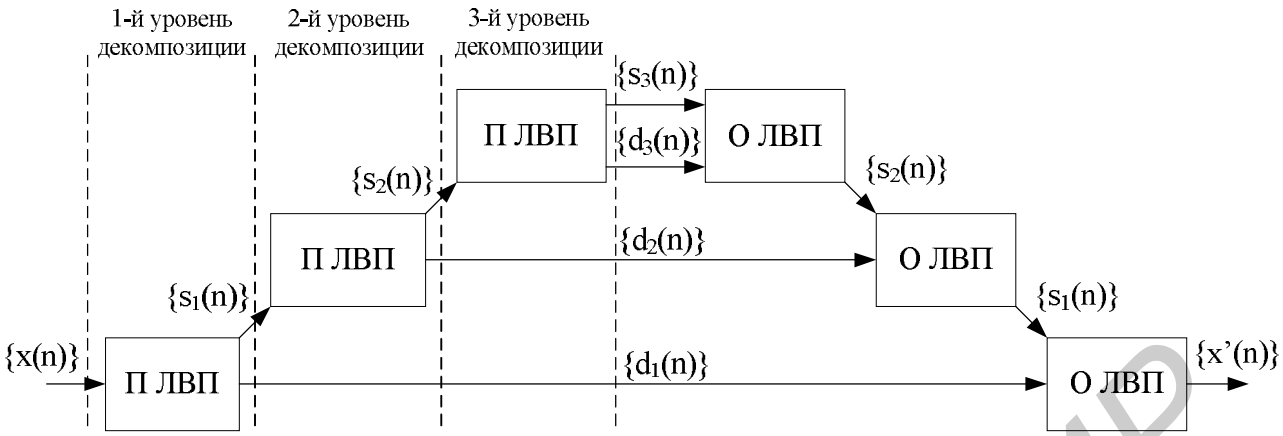
2) последовательность I шагов лифтинга, включающих в себя предсказание (умножение на нижнюю треугольную матрицу) и обновление (умножение на верхнюю треугольную матрицу);

3) масштабирование низкочастотного и высокочастотного сигнала.

Поскольку ЛВП будет применяться к последовательности значений пикселов изображения, то целесообразно далее употреблять термин «последовательность» вместо термина «сигнал».

Так как ЛВП по своей сущности близко к операции аппроксимации, то назовем низкочастотные коэффициенты $s(n)$ аппроксимационными коэффициентами, а высокочастотные $d(n)$ – детализирующими.

Последующие уровни разложения исходной последовательности получаются повторным преобразованием аппроксимационных коэффициентов, полученных на предыдущем уровне разложения. Схема прямого и обратного ЛВП для трех уровней разложения представлена на рисунке 1.4.



П ЛВП – прямое ЛВП,
О ЛВП – обратное ЛВП

Рисунок 1.4 – ЛВП для трех уровней разложения

Из (1.14) следует, что каждый шаг лифтинга можно определить набором коэффициентов полиномов $t(z),\tilde{t}(z),s(z),\tilde{s}(z)$ и коэффициентом масштабирования K.

Обозначим коэффициенты предсказания как $p_i(l)$, а коэффициенты обновления как $u_i(l)$. Эти коэффициенты равны коэффициентам соответствующего полинома ($\tilde{s}(z),\tilde{t}(z)$), стоящих при степени z, равной l. Далее будем считать, что ЛВП имеет одинаковые полиномы для синтеза и анализа. Максимальное значение l обозначим как ${}^{+}Z$. Величина ${}^{+}Z$ равна максимальной степени членов соответствующего полинома. Минимальное значение l обозначим как ${}^{-}Z$. Величина ${}^{-}Z$ равна минимальной степени членов соответствующего полинома. Для того чтобы различать максимальное и минимальное число коэффициентов для предсказания и обновления, а также для различных шагов лифтинга, введем обозначение Z_i^p для коэффициентов предсказания и Z_i^u для коэффициентов обновления i-го шага лифтинга.

Параметры одномерного ЛВП (базисную вейвлет-функцию) удобно задавать в виде таблицы коэффициентов предсказания и обновления (таблица 1.1).

Таблица 1.1 – Представление параметров ЛВП в табличной форме

	$l=-Z$	…	$l=-2$	$l=-1$	$l=0$	$l=1$	$l=2$	…	$l=Z$
p_1	$p_1({}^{-}Z_1^P)$	…	$p_1(-2)$	$p_1(-1)$	$p_1(0)$	$p_1(1)$	$p_1(2)$	…	$p_1({}^{+}Z_1^P)$
u_1	$u_1({}^{-}Z_1^u)$		$u_1(-2)$	$u_1(-1)$	$u_1(0)$	$u_1(1)$	$u_1(2)$		$u_1({}^{+}Z_1^u)$

Продолжение таблицы 1.1

p_2	$p_2(^{-}Z_2^p)$		$p_2(-2)$	$p_2(-1)$	$p_2(0)$	$p_2(1)$	$p_2(2)$		$p_2(^{+}Z_2^p)$
u_2	$u_2(^{-}Z_2^u)$		$u_2(-2)$	$u_2(-1)$	$u_2(0)$	$u_2(1)$	$u_2(2)$		$u_2(^{+}Z_2^u)$
p_I	$p_I(^{-}Z_I^p)$	…	$p_I(-2)$	$p_I(-1)$	$p_I(0)$	$p_I(1)$	$p_I(2)$	…	$p_I(^{+}Z_I^p)$
u_I	$u_I(^{-}Z_I^u)$		$u_I(-2)$	$u_I(-1)$	$u_I(0)$	$u_I(1)$	$u_I(2)$		$u_I(^{+}Z_I^u)$
K_p			$\frac{1}{K}$						
K_u			K						

В таблице 1.1 в строках p_i расположены коэффициенты, используемые для вычисления четных отсчетов, а в строках u_i – для вычисления нечетных.

Запишем математические выражения для прямого (1.15) и обратного (1.16) одномерного лифтинг вейвлет-преобразования:

$$\begin{cases} a_0(n) = s_{k-1}(2n), \\ b_0(n) = s_{k-1}(2n+1), \\ b_i(n) = b_{i-1}(n) - \left\lceil \sum_l p_i(l) a_{i-1}(n+l) \right\rceil, \\ a_i(n) = a_{i-1}(n) + \left\lceil \sum_l u_i(l) b_i(n+l) \right\rceil, \\ s_k(n) = \dfrac{a_I(n)}{K}, \\ d_k(n) = K b_I(n), \end{cases} \tag{1.15}$$

$$\begin{cases} a_I(n) = K s_j(n), \\ b_I(n) = d_j(n)/K, \\ a_{i-1}(n) = a_i(n) - \left\lfloor \sum_l u_i(l) b_i(n+l) \right\rfloor, \\ b_{i-1}(n) = b_i(n) + \left\lfloor \sum_l p_i(l) a_{i-1}(k+l) \right\rfloor, \\ s_{k-1}(2n) = a_0(n), \\ s_{k-1}(2n+1) = b_0(n), \end{cases} \tag{1.16}$$

где $k \in \overline{1, R_{\max}}$ – уровень разложения исходной последовательности;

i – шаг лифтинга;

I – количество шагов лифтинга для данного вейвлета.

Диапазон используемых коэффициентов предсказания и обновления на i-м шаге лифтинга задается следующими выражениями:

$$l \in [{}^{-}Z_i^p, {}^{+}Z_i^p], \quad (1.17)$$

$$l \in [{}^{-}Z_i^u, {}^{+}Z_i^u]. \quad (1.18)$$

В соответствии с табличным представлением ЛВП в таблицах 1.2–1.5 приведены параметры некоторых базисных вейвлет-функций.

Таблица 1.2 – Параметры лифтинг вейвлет-функции Хаар

l	0
p_1	-1
u_1	0,5
K_p	1
K_u	1

Таблица 1.3 – Параметры лифтинг вейвлет-функции 5/3

l	-1	0	1
p_1		$-0,5$	$-0,5$
u_1	0,25	0,25	
K_p	1		
K_u	1		

Таблица 1.4 – Параметры лифтинг вейвлет-функции 9/7-М

l	-1	0	1	2
p_1	0,0625	$-0,5625$	$-0,5625$	0,0625
u_1	0,25	0,25		
K_p	1			
K_u	1			

Таблица 1.5 – Параметры лифтинг вейвлет-функции 13/11

l	-2	-1	0	1	2	3
p_1	0,01171875	0,09765625	$-0,5859375$	$-0,5859375$	0,09765625	0,01171875
u_1		0,25	0,25			
K_p	1					
K_u	1					

Размер используемой окрестности и количество шагов лифтинга определяет вычислительную сложность преобразования на основе той или иной базисной вейвлет-функции. Размер используемой окрестности влияет на качество аппроксимации коэффициентов.

Граф-схема вычисления прямого лифтинг вейвлет-преобразования для расширения нулями и ${}^{-}Z = {}^{+}Z = 1$ представлена на рисунке 1.5.

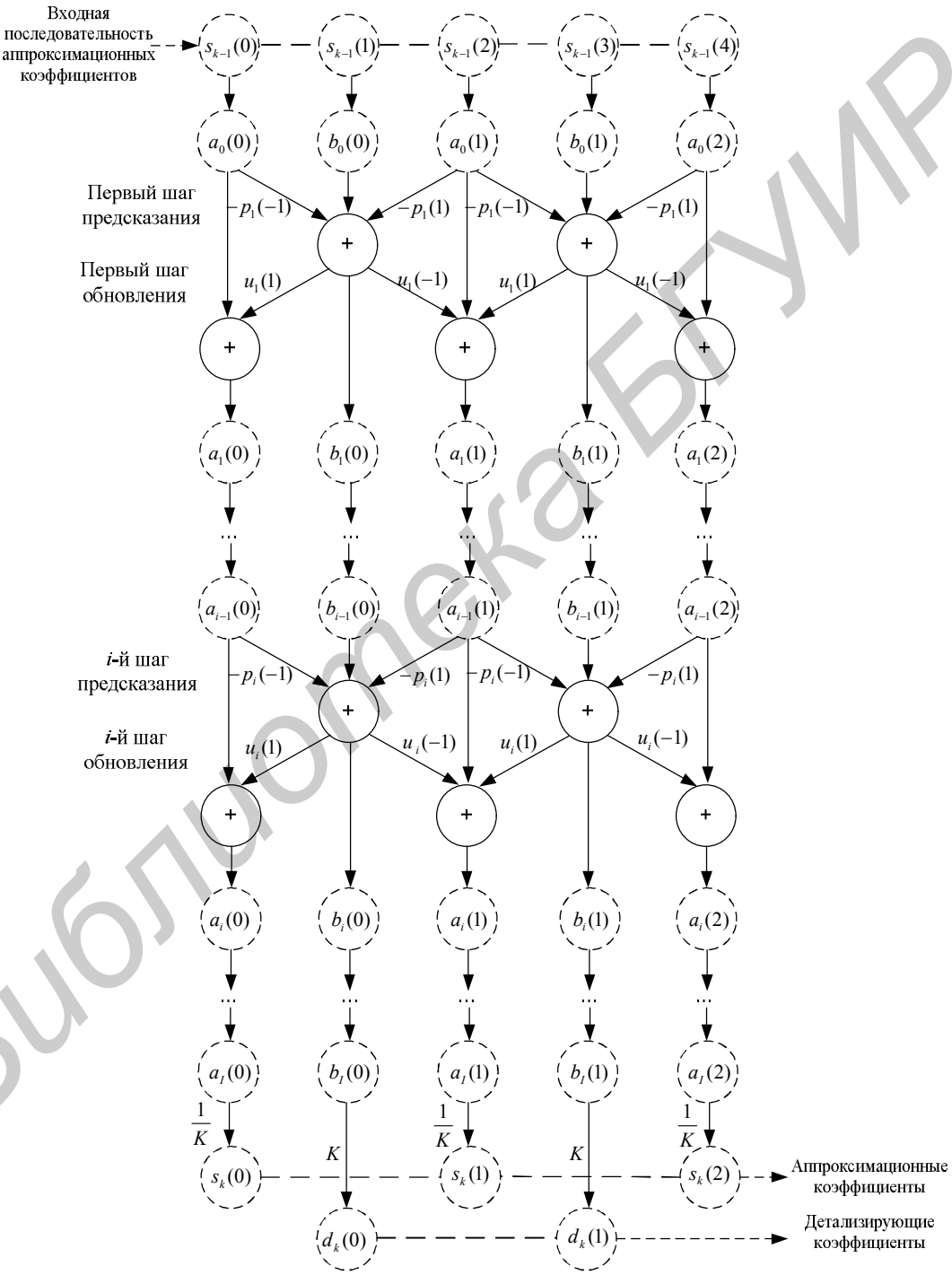


Рисунок 1.5 – Граф-схема вычисления прямого лифтинг вейвлет-преобразования для расширения нулями

1.2.3 Расширение последовательности коэффициентов

Так как последовательность аппроксимационных и детализирующих коэффициентов имеет конечную длину, то возникает проблема вычисления коэффициентов, находящихся на границах последовательности. Для корректной обработки граничных элементов применяют различные типы расширения используемой последовательности.

Расширение нулями является самым простым типом расширения, однако имеет существенный недостаток. На границах расширяемой последовательности может образоваться резкий перепад значений ее элементов, что приводит к негативным эффектам при лифтинг вейвлет-преобразовании.

Периодическое расширение предусматривает повторение граничных элементов. Недостатком данного расширения является то, что в точках расширения так же, как и при расширении нулями, может образоваться резкий перепад значений элементов.

Более оптимальным способом расширения используемой последовательности является симметричное расширение. В зависимости от того, дублируется или нет первый и последний элемент последовательности, симметричное расширение подразделяется на четное и нечетное. В последовательности, полученной симметричным расширением, отсутствуют резкие перепады значений элементов в точках расширения.

При расширении используемая последовательность увеличивается на определенное число элементов справа и/или слева. Пусть обрабатываемая последовательность размером T расширяется на S элементов справа и слева. Тогда размер T_S расширенной последовательности станет равным

$$T_S = T + 2S\,. \tag{1.19}$$

Будем считать, что в расширенной последовательности индексы элементов принимают следующие значения:

$$n \in [-S, T+S)\,. \tag{1.20}$$

Запишем выражения для различных типов расширения:

– расширение нулевыми значениями:

$$\bar{x}(n) = \begin{cases} x(n), & n \in [0, T), \\ 0, & n \in [-S, 0), \\ 0, & n \in [T, T+S), \end{cases} \tag{1.21}$$

где $\bar{x}(n)-$ элементы расширенной последовательности;

– периодическое расширение:

$$\overline{x}(n)=\begin{cases}x(n), & n\in[0,T),\\ x(n+S), & n\in[-S,0),\\ x(n-S), & n\in[T,T+S);\end{cases} \tag{1.22}$$

– четное симметрическое:

$$\overline{x}(n)=\begin{cases}x(n), & n\in[0,T),\\ x(|n|-1), & n\in[-S,0),\\ x(2T-n-1), & n\in[T,T+S);\end{cases} \tag{1.23}$$

– нечетное симметрическое:

$$\overline{x}(n)=\begin{cases}x(n), & n\in[0,T),\\ x(|n|), & n\in[-S,0),\\ x(2T-n-2), & n\in[T,T+S).\end{cases} \tag{1.24}$$

Если последовательность расширяется только вправо или влево, то размер T_S расширенной последовательности будет равным

$$T_S=T+S\,. \tag{1.25}$$

В расширенной с одной стороны последовательности индексы элементов принимают следующие значения:

– для расширенной слева последовательности

$$n\in[-S,T)\,; \tag{1.26}$$

– для расширенной справа последовательности

$$n\in[0,T+S)\,. \tag{1.27}$$

В формулах для расширения справа будет отсутствовать второе выражение, а для расширения слева – третье.

Последующая обработка расширенной последовательности начинается с нулевого элемента и продолжается до $(T-1)$ включительно. После обработки элементы, добавленные к последовательности после расширения, отбрасываются.

Получение исходной последовательности из расширенной описывается следующим выражением:

$$x(n)=\overline{x}(n),\quad n\in[0,T)\,. \tag{1.28}$$

Данное выражение справедливо для всех типов расширения.

1.2.4 Двумерное лифтинг вейвлет-преобразование

В качестве исходных данных для двумерного лифтинг вейвлет-преобразования (ДЛВП) используются полутоновые изображения размером $M \times N$, где $M = N$. Изображение можно представлять в виде матрицы целых чисел:

$$U = (x(m,n))_{M \times N}, \tag{1.29}$$

где m и n – номер строки и столбца матрицы соответственно.

Для того чтобы применить одномерное ЛВП к матрице, используются различные схемы. Наиболее часто используемая схема заключается в преобразовании матрицы изображения в два этапа: сначала по строкам, а затем по столбцам или наоборот.

При использовании данной схемы на первом шаге преобразования ко всем строкам (столбцам) матрицы изображения применяется одномерное ЛВП (рисунок 1.6, а). При этом одну половину исходной матрицы займут низкочастотные аппроксимационные коэффициенты, а другую половину – высокочастотные детализирующие коэффициенты. Далее полученная матрица преобразуется аналогичным образом по столбцам (строкам) (рисунок 1.6, б). Результирующая матрица состоит из четырех подобластей: LL, LH, HL, HH. Будем считать, что в результате одного шага ДЛВП образуются одна область матрицы с аппроксимационными коэффициентами (LL) и область матрицы с детализирующими коэффициентами, состоящая из трех подобластей (HL, LH, HH). Область аппроксимационных коэффициентов представляет собой уменьшенную копию исходного изображения.

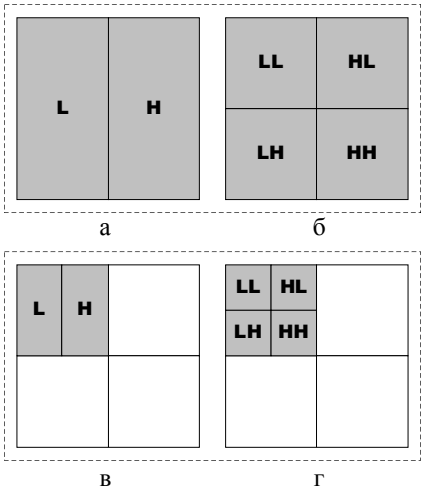


Рисунок 1.6 – Формирование матрицы вейвлет-коэффициентов исходного изображения

На следующем шаге ДЛВП преобразуется область аппроксимационных коэффициентов, полученная на первом шаге (рисунок 1.6, в, г). Таким образом, с каждым шагом преобразования увеличивается число детализирующих и уменьшается количество аппроксимационных коэффициентов. Возможное количество шагов преобразования зависит от размера исходной матрицы изображения, от вейвлет-функции, выбранной для одномерного лифтинг вейвлет-преобразования, требуемой вычислительной сложности алгоритма и т. д.

Алгоритм обратного ДЛВП зеркально симметричен алгоритму прямого ДЛВП. Вначале осуществляется постолбцовая обработка матрицы вейвлет-коэффициентов одномерным обратным ЛВП, а затем – построчная. Количество уровней разложения для обратного ДЛВП должно соответствовать количеству уровней разложения, осуществленных при прямом алгоритме ДВЛП. Перед переходом на следующий шаг изменяется область обработки матрицы – увеличивается длина строк и высота столбцов в два раза, а площадь обработки – в четыре.

1.3 Компрессия коэффициентов вейвлет-матрицы изображения

Прогрессивное сжатие заключается в последовательной передаче информации от кодера к декодеру в порядке убывания ее значимости. Вначале передается наиболее важная часть данных, по которым можно восстановить грубую версию вейвлет-матрицы изображения. Последующие данные дополняют данные, принятые ранее, и позволяют восстановить вейвлет-матрицу изображения более точно.

Для вейвлет-матрицы наиболее значимыми являются коэффициенты с наибольшей амплитудой. Следовательно, для прогрессивной передачи вейвлет-матрицы может быть использована передача по битовым плоскостям в последовательности от наиболее значимой к наименее значимой битовой плоскости.

Каждый коэффициент матрицы $c(m,n)$ является либо значимым, либо незначимым для заданной битовой плоскости k и менее значимых битовых плоскостей. Коэффициент $c(m,n)$ является значимым для битовой плоскости k и менее значимых битовых плоскостей $k-1, k-2, \ldots, 0$, если выполняется условие

$$\left\lfloor \log_2 |c(m,n)| \right\rfloor \geq k, \quad k \in [0, K-1], \tag{1.30}$$

где K – количество битовых плоскостей вейвлет-матрицы.

Условие поиска коэффициентов для битовой плоскости k можно представить в виде функции

$$Sn(c(m,n)) = \begin{cases} 1, \textit{ если } 2^k \leq |c(m,n)| < 2^{n+1}; \\ 0 \ \textit{ иначе.} \end{cases} \tag{1.31}$$

Сжатие может быть достигнуто за счет передачи только координат и знаков значимых элементов для заданной битовой плоскости и передачи значений текущей битовой плоскости для элементов, найденных при анализе более значимых битовых плоскостей.

Обобщенный алгоритм прогрессивного сжатия.

1 Шаг инициализации.

Инициализация списков и переменных $k = K - 1$.

2 Шаг поиска значимых коэффициентов.

Поиск и вывод координат (m,n) и знаков $sign(c(m,n))$ для коэффициентов, которые удовлетворяют условию $Sn(c(m,n)) = 1$. Если найденные для предыдущей битовой плоскости значимые элементы маркируются и не участвуют в текущем шаге поиска, то выражение (1.30) может быть использовано вместо (1.31) в качестве условия поиска. При этом для поиска значимых элементов используются значения самой битовой плоскости.

3 Шаг уточнения.

Вывод значений текущей битовой плоскости для коэффициентов, которые удовлетворяют условию:

$$|c(m,n)| \geq 2^{k+1}, \tag{1.32}$$

в том же порядке, в котором передавались их координаты.

4 Шаг квантования.

Если $k = 0$, то алгоритм завершен, иначе уменьшить k на единицу и перейти на шаг 2.

Предполагается что декодеру известны размеры вейвлет-матрицы, а также количество уровней разложения и количество битовых плоскостей.

Алгоритм поиска значимых элементов основан на представлении вейвлет-матрицы набором множеств определенного вида и рекурсивным делением значимых множеств до тех пор, пока не будут определены координаты отдельных значимых коэффициентов.

Выражения (1.30) и (1.31) могут быть применены к множеству коэффициентов M путем замены $|c(m,n)|$ на $\max_{(m,n)\in M}(|c(m,n)|)$.

Форма множества должна учитывать свойства вейвлет-матрицы и обеспечивать объединение максимального количества незначимых элементов в минимальном количестве множеств.

Явная передача координат элементов вейвлет-матрицы является неэффективной. Вместо явной передачи координат можно отслеживать путь, который проходит алгоритм поиска значимых коэффициентов. Путь выполнения алгоритма определяется ветвлением в точках принятия решения о значимости или незначимости коэффициента или множества коэффициентов вейвлет-матрицы. Передача информации о координатах значимых элементов заключается в обеспечении идентичности пути поиска, который проходят кодер и декодер. Таким образом, передача координат коэффициентов заменяется передачей бинарных решений алгоритма поиска.

1.3.1 Компрессия на основе учета межсубполосной избыточности

Одним из подходов к представлению вейвлет-матрицы в виде рекурсивно делимых множеств является формирование древовидных структур. Этот способ учитывает корреляцию между коэффициентами, находящимися в разных частотных поддиапазонах и представляющих один пространственный участок изображения. Алгоритм SPIHT (Set Partitioning In Hierarchical Trees) использует древовидные структуры, показанные на рисунке 1.7.

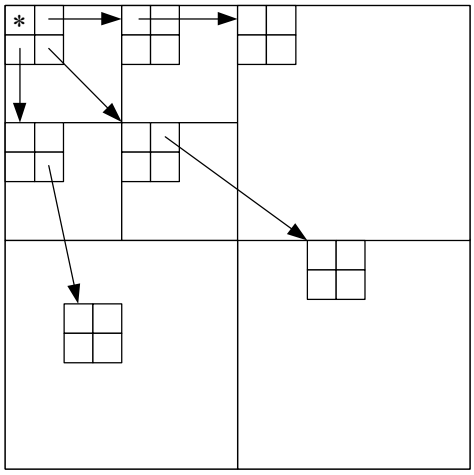


Рисунок 1.7 – Представление вейвлет-матрицы в виде пространственно-ориентированных деревьев

Такие структуры называются пространственно-ориентированными деревьями. Каждый узел дерева соответствует коэффициенту матрицы и определяется его координатами. Прямые потомки узла соответствуют коэффициентам матрицы, расположенным в том же пространственном направлении, но в следующем, более высокочастотном поддиапазоне.

Дерево определяется таким образом, что каждый узел либо не имеет прямых потомков, либо имеет четыре прямых потомка, являющихся группой 2×2 смежных коэффициентов. На рисунке 1.7 стрелки указывают направление от родительского узла к его четырем прямым потомкам. Элементы, находящиеся в самой низкочастотной области матрицы, называются корнями и также образуют группы элементов 2×2. Однако в каждой из групп один из элементов (на рисунке 1.7 отмечен звездочкой) не имеет потомков.

Для кодирования вейвлет-матрицы алгоритм использует следующие множества:

- $O(m,n)$ – множество координат всех прямых потомков узла (m,n);

- $D(m,n)$ – множество координат всех потомков узла (m,n);

- H – множество координат корней всех пространственно-ориентированных деревьев (узлы в низкочастотной области вейвлет-матрицы);

- $L(m,n)$ – множество всех потомков узла (m,n), за исключением прямых потомков.

Например, множество $O(m,n)$ для всех частотных поддиапазонов, за исключением самого низкочастотного и самого высокочастотного, определяется как

$$O(m,n) = \{(2m,2n),(2m,2n+1),(2m+1,2n),(2m+1,2n+1)\}. \quad (1.33)$$

Части пространственно-ориентированных деревьев используются как объекты деления в алгоритме поиска значимых элементов. Деление пространственно-ориентированных деревьев осуществляется по следующим правилам:

1 Начальное разбиение образуется множеством $\{(m,n)\}$ и множествами $D(m,n)$ для всех $(m,n) \in H$.

2 Если $Sn(D(m,n)) = 1$, то это множество делится на $L(m,n)$ и четыре отдельных коэффициента с координатами $(x,y) \in O(m,n)$.

3 Если $Sn(L(m,n)) = 1$, то оно делится на четыре множества $D(x,y)$ с $(x,y) \in O(m,n)$.

Для хранения множеств и отдельных коэффициентов алгоритм использует упорядоченные списки, так как важен порядок, в котором множества и отдельные коэффициенты проверяются на значимость.

Алгоритм работает с тремя списками:

- список незначимых множеств (list of insignificant sets, LIS);
- список незначимых коэффициентов (list of insignificant pixels, LIP);
- список значимых коэффициентов (list of significant pixels, LSP).

Во всех списках элементы представляются координатами (m,n). Для списков LIS и LSP координаты определяют коэффициенты вейвлет-матрицы. Для списка LIS координаты определяют множества $D(m,n)$ или $L(m,n)$. Для различия между элементами списка LIS будем считать, что элемент списка типа A представляет множество $D(m,n)$, а типа B – множество $L(m,n)$ (рисунок 1.8).

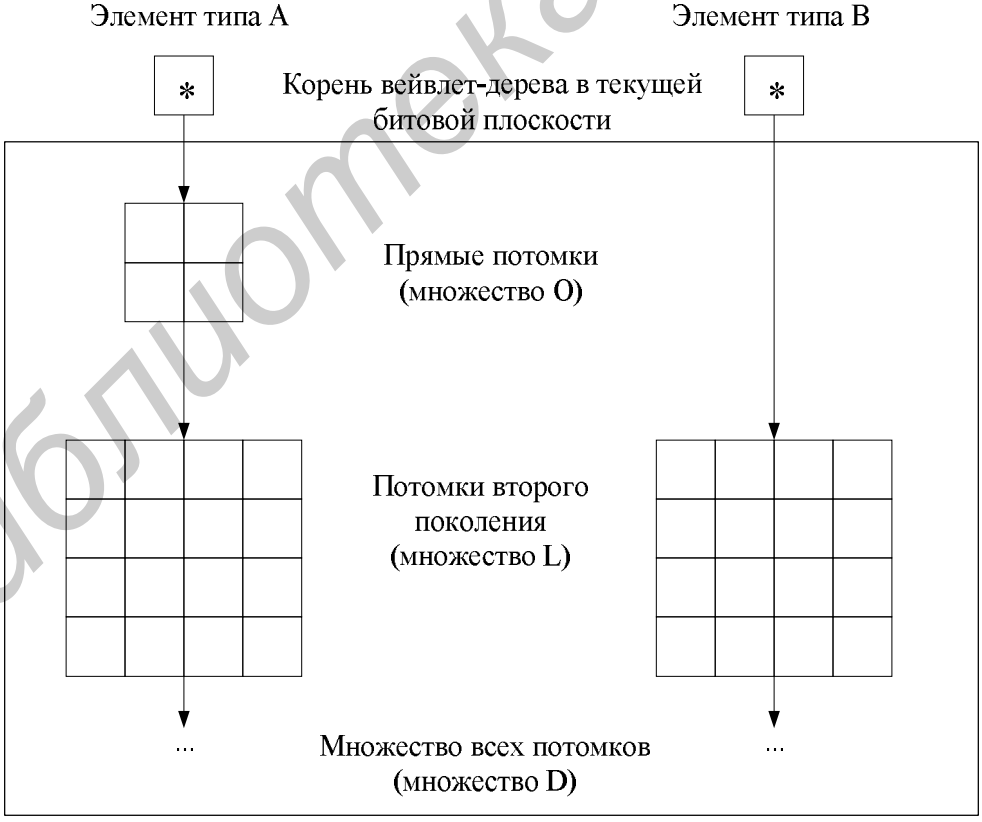


Рисунок 1.8 – Элементы списка LIS

Алгоритм поиска значимых коэффициентов проверяет на значимость элементы списка LIP (незначимые коэффициенты для предыдущей битовой плоскости). Элементы, которые определены как значимые, перемещаются в список LSP. Подобным образом проверяются на значимость элементы списка LIS. Если элемент списка LIS определен как значимый, то он удаляется из списка и делится далее. Образованные после деления подмножества добавляются в конец списка LIS, если имеют более одного элемента. Если множество образуется единичным коэффициентом, то он добавляется в конец списка LSP или LIP в зависимости от его значимости.

Список LSP содержит координаты коэффициентов, которые должны быть обработаны на шаге уточнения.

Алгоритм можно представить в виде следующего псевдокода:

1 Шаг инициализации.

1.1 $k = K$.

1.2 $LSP = \varnothing$.

1.3 $LIP = H$.

1.4 Добавить в LIS как элементы типа A те $(m,n) \in H$, которые имеют потомков.

2 Шаг поиска.

2.1 Для каждого элемента (m,n) из списка LIP:

2.1.1 Вывести $Sn(c(m,n))$.

2.1.2 Если $Sn(c(m,n)) = 1$, то переместить элемент (m,n) из LIP в LSP и вывести его знак $sign(c(m,n))$.

2.2 Для каждого элемента (m,n) из списка LIS:

2.2.1 Если элемент типа A:

2.2.1.1 Вывести $Sn(D(m,n))$.

2.2.1.2 Если $Sn(D(m,n)) = 1$, то:

2.2.1.2.1 Для каждого $(x,y) \in O(m,n)$:

2.2.1.2.1.1 Вывести $Sn(c(x,y))$.

2.2.1.2.1.2 Если $Sn(c(x,y)) = 1$, то добавить (x,y) в LSP и вывести его знак $sign(c(x,y))$.

2.2.1.2.1.3 Если $Sn(c(x,y)) = 0$, то добавить (x,y) в конец LIP.

2.2.1.2.2 Если $L(m,n) \neq \varnothing$, то переместить (m,n) в конец списка LIS как элемент типа B и перейти на шаг 2.2.2, иначе – удалить элемент (m,n) из LIS.

2.2.2 Если элемент типа B:

2.2.2.1 Вывести $Sn(L(m,n))$.

2.2.2.2 Если $Sn(L(m,n)) = 1$, то:

2.2.2.2.1 Добавить каждый $(x,y) \in O(m,n)$ в конец списка LIS как элемент типа A.

2.2.2.2.2 Удалить (m,n) из списка LIS.

3 Шаг уточнения.

3.1 Для каждого элемента (m,n) из LSP, за исключением добавленных на последнем шаге поиска (для текущего k), вывести k-й бит $|c(m,n)|$.

4 Шаг квантования.

4.1 Если $k > 0$, то:

4.1.1 $k = k - 1$.

4.1.2 Перейти на шаг 2.

4.2 Иначе – конец алгоритма.

Важной особенностью алгоритма является то, что элементы, добавленные в конец списка LIS на шаге поиска, проверяются до окончания этого шага.

Выходной поток кодера содержит биты, определяющие путь ветвления алгоритма (биты значимости), биты знаков коэффициентов и биты уточнения.

Декодер работает синхронно с кодером, восстанавливая путь кодера по битам принятия решения о значимости. Таким образом, алгоритм декодера может быть получен на основе алгоритма кодера путем замены операций вывода на операции чтения битов из входного потока. При этом алгоритм декодера формирует идентичные списки LIS, LSP и LIP.

Вместо проверки коэффициентов на значимость декодер обновляет их значения в восстанавливаемой матрице. Если при обработке битовой плоскости k-элемент перемещается в LSP, то это означает, что выполняется условие

$$2^k \le |c(m,n)| < 2^{k+1}, \tag{1.34}$$

т. е. k-й бит является старшим значащим битом коэффициента.

Декодер, используя эту информацию, а также информацию о знаке и биты уточнения, может точно восстановить значения коэффициентов вейвлет-матрицы.

1.3.2 Компрессия на основе учета внутриполосной избыточности

Идея алгоритма SPECK (Set Partitioned Embedded bloCK coder) заключается в использовании кластеризации энергии внутри субполос вейвлет-матрицы.

Алгоритм SPECK использует в качестве основного объекта деления блоки квадратной формы, состоящие из смежных элементов. Эти блоки обозначаются как множества типа S и могут быть различных размеров.

Размер множества определяется количеством его элементов:

$$size(S) \equiv |S|. \tag{1.35}$$

Другим типом множества, с которым работает алгоритм SPECK, является множество I. Множество I формируется из вейвлет-матрицы путем удаления

некоторого количества низкочастотных поддиапазонов. Размер множества I зависит от количества удаленных частотных поддиапазонов (рисунок 1.9).

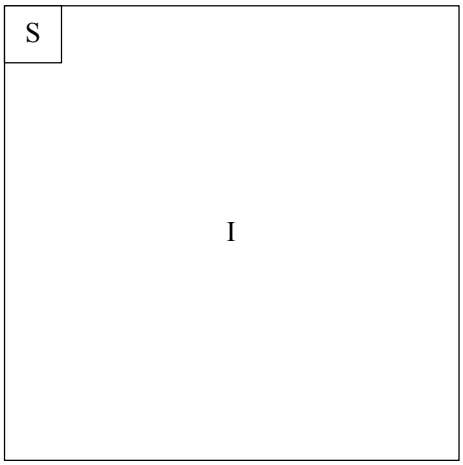


Рисунок 1.9 – Разбиение вейвлет-матрицы на множество S и множество I

Алгоритм работает с двумя упорядоченными списками:

- список незначимых множеств (List of insignificant sets, LIS);
- список значимых коэффициентов (List of significant pixels, LSP).

Элементами списка LIS являются множества типа S. Множество S определяется его позицией (m,n) и размером $|S|$. Позицию множества могут определять координаты его верхнего левого угла. Если $|S|=1$, то такое множество представляет отдельный элемент вейвлет-матрицы. Элементами LSP являются отдельные коэффициенты, представленные координатами (m,n).

Алгоритм SPECK описывается следующим псевдокодом:

1 Шаг инициализации.

1.1 Разбить вейвлет-матрицу на два множества $S = LL$ и $I = C - LL$ (см. рисунок 1.9).

1.2 $k = K$.

1.3 Добавить S в LIS; $LSP = \varnothing$.

2 Шаг поиска.

2.1 Для каждого можества S из LIS в порядке возрастания $|S|$ выполнить процедуру ProcessS (S).

2.2 Выполнить процедуру ProcessI().

3 Шаг уточнения.

3.1 Для каждого (m,n) из LSP, за исключением элементов, добавленных на текущем шаге поиска вывести k-й бит $|c\ (m,n)|$.

4 Шаг квантования.

4.1 Если $k > 0$, то

4.1.1 $k = k - 1$.

4.1.2 Перейти на шаг 2.

4.2 Иначе – конец алгоритма.

Процедура ProcessS(S).

1 Вывести $Sn(S)$.

2 Если $Sn(S)=1$, то:

2.1 Если $|S|=1$ и S имеет координаты (m,n), то вывести знак $sign(c(m,n))$ и добавить (m,n) в LSP.

2.2 Иначе выполнить процедуру CodeS(S).

2.3 Если $S \in LIS$, удалить S из LIS.

3 Иначе:

3.1 Если $S \notin LIS$, добавить S к LIS.

Процедура CodeS(S).

1 Деление множества S на четыре подмножества $O(S)$ (рисунок 1.10, а).

2 Для каждого из четырех $O(S)$:

2.1 Вывести $Sn(O(S))$.

2.2 Если $Sn(O(S))=1$, то:

2.2.1 Если $|O(S)|=1$ и $O(S)$ имеет координаты (m,n), то:

2.2.1.1 Вывести знак $sign(c(m,n))$.

2.2.1.2 Добавить (m,n) к LSP.

2.2.2 Иначе – выполнить процедуру $CodeS(O(S))$.

2.3 Иначе – добавить $O(S)$ к LIS.

Процедура ProcessI().

1 Если I – пустое множество, то выход из процедуры.

2 Вывести $Sn(I)$.

3 Если $Sn(I)=1$, то выполнить процедуру CodeI().

Процедура CodeI().

1 Разделить множество I на три множества S и множество I меньшего размера (рисунок 1.10, б).

2 Для каждого из трех множеств S выполнить процедуру ProcessS(S).

3 Выполнить процедуру ProcessI().

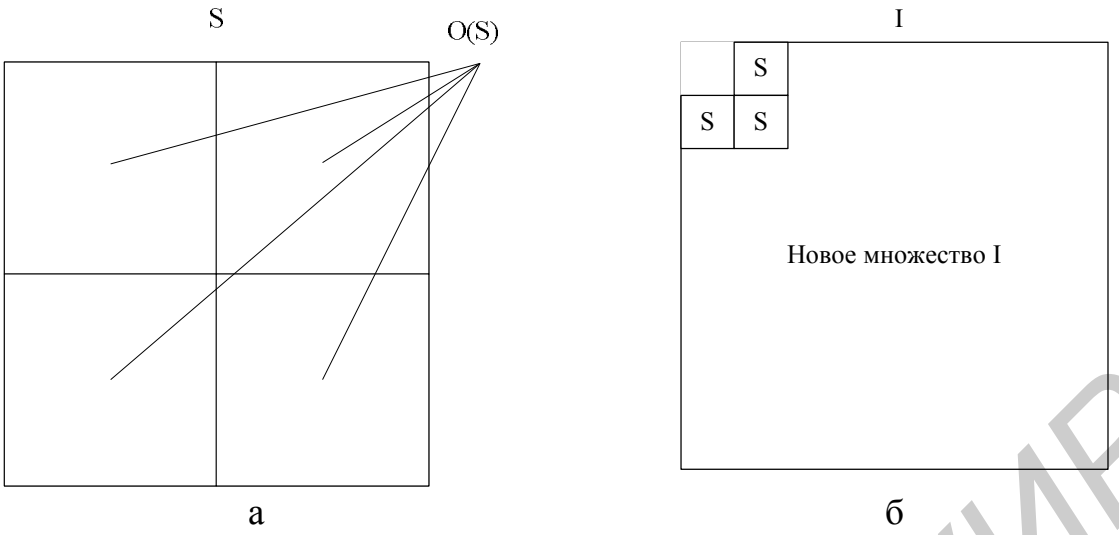


Рисунок 1.10 – Разбиение множества S (а) и множества I (б) на подмножества

Деление множества S

Каждое множество S из LIS проверяется на значимость для определенного номера битовой плоскости. Если оно незначимо, то остается в LIS. Если множество значимо, то делится на подмножества одинакового размера, составляющего ¼ от размера родительского блока. В последующей процедуре CodeS() каждый из четырех потомков $O(S)$ проверяется на значимость для того же k, и если является значимым, то делится таким же образом еще на четыре подмножества и т. д. Если потомок незначим, то он добавляется в LIS. Таким образом, множества S делятся рекурсивно в функциях ProcessS() и CodeS(), пока не будут найдены все значимые элементы вейвлет-матрицы для текущей битовой плоскости. Все незначимые множества, образовавшиеся в результате поиска, добавляются в LIS и затем проверяются на значимость для меньшего значения k.

Деление множества I

Когда для заданного k проверены все множества S, проверяется на значимость множество I. Если множество I значимо, то оно делится по схеме октавного деления на 3 множества типа S и множество типа I меньшего размера (см. рисунок 1.10, б).

Так как список LIS обрабатывается в порядке увеличения размера блока, то более мелкие множества, добавленные на текущей итерации алгоритма, повторно не обрабатываются для текущего значения k.

Декодер строится по тем же принципам, что и декодер алгоритма SPIHT.

При реализации алгоритма SPECK существует возможность оптимизировать некоторые его части. Список LIS содержит нуль-блоки различного размера, которые добавляются в произвольном порядке. Однако в соответствии с алгоритмом элементы типа S требуется обрабатывать в порядке возрастания их размера. Таким образом, требуется дополнительная операция предварительной сортировки списка LIS. Этого можно избежать, если использовать несколько списков LIS (LIS 1, LIS 2, LIS 4, и т. д.), которые хранят нуль-блоки одинакового размера. Количество таких списков определяется наибольшим размером элемента типа S. В этом случае достаточно добавлять элементы типа S в список LIS для соответствующего размера, а на шаге поиска обрабатывать списки LIS последовательно в порядке возрастания хранимых элементов.

На шаге обновления требуется вывести значения текущей битовой плоскости для всех элементов списка LSP за исключением тех, которые были добавлены на шаге поиска для текущей битовой плоскости. Для выполнения этого условия можно использовать два отдельных списка LSP-C (current) и LSP-P (previous). На шаге поиска элементы добавляются в список LSP-C. На шаге обновления используется список LSP-P. Элементы списка LSP-C перемещаются в список LSP-P после шага обновления

Рассмотрим работу алгоритма SPECK для двух старших битовых плоскостей на примере вейвлет-матрицы размером 8×8 элементов, имеющей 3 уровня декомпозиции и 6 битовых плоскостей (рисунок 1.11).

	0	1	2	3	4	5	6	7
0	63	-34	49	10	7	13	-12	7
1	-31	23	14	-13	3	4	6	-1
2	15	14	3	-12	5	-7	3	9
3	-9	-7	-14	8	4	-2	3	2
4	-5	9	-1	47	4	6	-2	2
5	3	0	-3	2	3	-2	0	4
6	2	-3	6	-4	3	6	3	6
7	5	11	5	6	0	3	-4	4

Рисунок 1.11 – Вейвлет-матрица 8×8

На рисунке 1.11 границы субполос вейвлет-матрицы отмечены жирными линиями.

Работу алгоритма рассмотрим по шагам. Нумерация шагов продолжается при обработке последующих битовых плоскостей.

1 Шаг инициализации.

LIS 1 = {[0,0])};
LIS 2 = {};
LIS 4 = {};
LSP-C = {};
LSP-P = {};
$I = [1,1]$;
$k = 5$.

2 Шаг поиска для 5-й битовой плоскости.

5-я битовая плоскость вейвлет-матрицы представлена на рисунке 1.12.

Изменения состояния списков и выходных данных на шаге поиска представлены в таблице 1.6. Новые элементы списков помечены как подчеркнутые. Удаленные элементы списка зачеркнуты.

	0	1	2	3	4	5	6	7
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

Рисунок 1.12 – 5-я битовая плоскость вейвлет-матрицы

Таблица 1.6 – Поиск значимых элементов для 5-й битовой плоскости

Действие	Состояние списков LIS, LSP и множества I					Вывод
	LIS 4	LIS 2	LIS 1	LSP-C	I	
Исходное состояние			[0,0]		[1,1]	
ProcessS(LIS 1[0,0])			~~[0,0]~~	[0,0]	[1,1]	$Sn(LIS1[0,0]) = 1$ $sign([0,0]) = 0$
ProcessI()	без изменений					$Sn(I) = 1$
CodeI()				[0,0]	~~[1,1]~~ [2,2]	–
ProcessS(LIS 1[0,1])				[0,0] [0,1]	[2,2]	$Sn(LIS1[0,1]) = 1$ $sign([0,1]) = 1$

Продолжение таблицы 1.6

Действие	Состояние списков LIS, LSP и множества I					Вывод
	LIS 4	LIS 2	LIS 1	LSP-C	I	
ProcessS(LIS 1[1,1])			[1,1]	[0,0] [0,1]	[2,2]	$Sn(LIS1[1,1]) = 0$
ProcessS(LIS 1[1,0])			[1,1] [1,0]	[0,0] [0,1]	[2,2]	$Sn(LIS1[1,0]) = 0$
ProcessI()	без изменений					$Sn(I) = 1$
CodeI() пункты 1-2			[1,1] [1,0]	[0,0] [0,1]	~~[2,2]~~ [4,4]	–
ProcessS(LIS 2[0,2]) пункты 1-2	без изменений					$Sn(LIS2[0,2]) = 1$
CodeS (LIS 2[0,2]) пункт 2 для $O_1(LIS2[0,2]) = LIS1[0,2]$			[1,1] [1,0]	[0,0] [0,1] [0,2]	[4,4]	$Sn(O_1(LIS2[0,2])) = 1$ $sign([0,2]) = 0$
пункт 2 для $O_2(LIS2[0,2]) = LIS1[0,3]$			[1,1] [1,0] [0,3]	[0,0] [0,1] [0,2]	[4,4]	$Sn(O_2(LIS2[0,2])) = 0$
пункт 2 для $O_3(LIS2[0,2]) = LIS1[1,2]$			[1,1] [1,0] [0,3] [1,2]	[0,0] [0,1] [0,2]	[4,4]	$Sn(O_3(LIS2[0,2])) = 0$
пункт 2 для $O_4(LIS2[0,2]) = LIS1[1,3]$			[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]	[4,4]	$Sn(O_4(LIS2[0,2])) = 0$
ProcessS(LIS 2[2,2])		[2,2]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]	[4,4]	$Sn(LIS2[2,2]) = 0$
ProcessS(LIS 2[2,0])		[2,2] [2,0]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]	[4,4]	$Sn(LIS2[2,0]) = 0$
ProcessI()	без изменений					$Sn(I) = 1$
CodeI()		[2,2] [2,0]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]	~~[4,4]~~	–
ProcessS(LIS 4[0,4])	[1,4]	[2,2] [2,0]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]		$Sn(LIS4[0,4]) = 0$

Продолжение таблицы 1.6

Действие	Состояние списков LIS, LSP и множества I					Вывод
	LIS 4	LIS 2	LIS 1	LSP-C	I	
ProcessS(LIS 4[4,4])	[1,4] [4,4]	[2,2] [2,0]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]		$Sn(LIS4[4,4]) = 0$
ProcessS(LIS 4[4,0]) пункты 1-2	без изменений					$Sn(LIS4[4,0]) = 1$
CodeS (LIS 4[4,0]) пункт 2 для O_1 (LIS 4[4,0]) = LIS2[4,0]	[1,4] [4,4]	[2,2] [2,0] [4,0]	[1,1] [1,0] [0,3] [1,2] [1,3]	[0,0] [0,1] [0,2]		$Sn(O_1(LIS4[4,0])) = 0$
CodeS (LIS 4[4,0]) пункт 2 для O_2 (LIS 4[4,0]) = LIS2[4,2]	без изменений					$Sn(O_2(LIS4[4,0])) = 1$
CodeS (LIS 2[2,4]) пункт 2 для O_1 (LIS 2[4,2]) = LIS1[4,2]	[1,4] [4,4]	[2,2] [2,0] [4,0]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2]	[0,0] [0,1] [0,2]		$Sn(O_1(LIS2[4,2])) = 0$
пункт 2 для O_2 (LIS 2[4,2]) = LIS1[4,3]	[1,4] [4,4]	[2,2] [2,0] [4,0]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2]	[0,0] [0,1] [0,2] [4,3]		$Sn(O_2(LIS2[4,2])) = 1$ $sign([4,3]) = 0$
пункт 2 для O_3 (LIS 2[4,2]) = LIS1[5,2]	[1,4] [4,4]	[2,2] [2,0] [4,0]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2] [5,2]	[0,0] [0,1] [0,2] [4,3]		$Sn(O_3(LIS2[4,2])) = 0$
пункт 2 для O_4 (LIS 2[4,2]) = LIS1[5,3]	[1,4] [4,4]	[2,2] [2,0] [4,0]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]	[0,0] [0,1] [0,2] [4,3]		$Sn(O_4(LIS2[4,2])) = 0$

Продолжение таблицы 1.6

Действие	Состояние списков LIS, LSP и множества I					Вывод
	LIS 4	LIS 2	LIS 1	LSP-C	I	
CodeS (LIS 4[4,0]) пункт 2 для O_3 (LIS 4[4,0]) = LIS1[6,0]	[1,4] [4,4]	[2,2] [2,0] [4,0] [6,0]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]	[0,0] [0,1] [0,2] [4,3]		$Sn(O_3(LIS\,4[4,0])) = 0$
пункт 2 для O_4 (LIS 4[4,0]) = LIS1[6,4]	[1,4] [4,4]	[2,2] [2,0] [4,0] [6,0] [6,4]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]	[0,0] [0,1] [0,2] [4,3]		$Sn(O_4(LIS\,4[4,0])) = 0$

Отметим на рисунке 1.13 найденные значимые элементы (заштрихованные) и образовавшиеся нуль-блоки (обведены жирными линиями).

	0	1	2	3	4	5	6	7
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

Рисунок 1.13 – Найденные значимые элементы и образовавшиеся нуль-блоки после шага поиска для 5-й битовой плоскости

3 Шаг уточнения. Так как обрабатывается старшая битовая плоскость, шаг уточнения не требуется.

LSP T = {[0,0]; [0,1]; [0,2]; [4,3]}

4 Шаг квантования.

$k = 5$.

Далее переходим на второй шаг алгоритма для 4-й битовой плоскости.

5 Шаг поиска для 4-й битовой плоскости.

4-я битовая плоскость вейвлет-матрицы представлена на рисунке 1.14. Действия по поиску значимых элементов представлены в таблице 1.7.

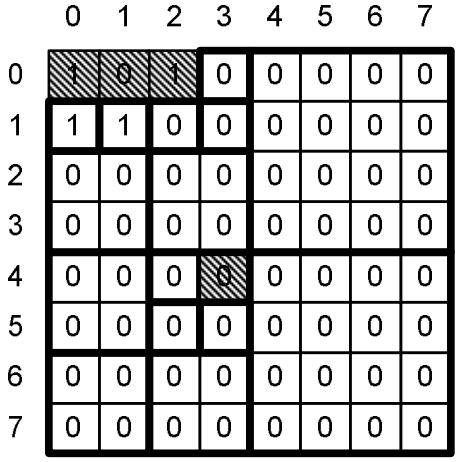


Рисунок 1.14 – 4-я битовая плоскость вейвлет-матрицы

Таблица 1.7 – Поиск значимых элементов для 4-й битовой плоскости

Действие	Состояние списков LIS, LSP и множества I					Вывод
	LIS 4	LIS 2	LIS 1	LSP C	I	
Исходное состояние	[1,4] [4,4]	[2,2] [2,0] [4,0] [6,0] [6,4]	[1,1] [1,0] [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]			
ProcessS(LIS 1[1,1])	[1,4] [4,4]	[2,2] [2,0] [4,0] [6,0] [6,4]	~~[1,1]~~ [1,0] [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]	[1,1]		$Sn(LIS1[1,1]) = 1$ $sign([1,1]) = 0$
ProcessS(LIS 1[1,0])	[1,4] [4,4]	[2,2] [2,0] [4,0] [6,0] [6,4]	~~[1,0]~~ [0,3] [1,2] [1,3] [4,2] [5,2] [5,3]	[1,1] [1,0]		$Sn(LIS1[1,0]) = 1$ $sign([1,0]) = 1$

Продолжение таблицы 1.7

Действие	Состояние списков LIS, LSP и множества *I*					Вывод
	LIS 4	LIS 2	LIS 1	LSP C	*I*	
ProcessS(LIS 1[0,3])	Без изменений					$Sn(LIS1[0,3]) = 0$
ProcessS(LIS 1[1,2])	Без изменений					$Sn(LIS1[1,2]) = 0$
ProcessS(LIS 1[1,3])	Без изменений					$Sn(LIS1[1,3]) = 0$
ProcessS(LIS 1[4,2])	Без изменений					$Sn(LIS1[4,2]) = 0$
ProcessS(LIS 1[5,2])	Без изменений					$Sn(LIS1[5,2]) = 0$
ProcessS(LIS 1[5,3])	Без изменений					$Sn(LIS1[5,3]) = 0$
ProcessS(LIS 2[2,2])	Без изменений					$Sn(LIS2[2,2]) = 0$
ProcessS(LIS 2[2,0])	Без изменений					$Sn(LIS2[2,0]) = 0$
ProcessS(LIS 2[4,0])	Без изменений					$Sn(LIS2[4,0]) = 0$
ProcessS(LIS 2[6,0])	Без изменений					$Sn(LIS6[2,0]) = 0$
ProcessS(LIS 2[6,4])	Без изменений					$Sn(LIS2[6,4]) = 0$
ProcessS(LIS 4[1,4])	Без изменений					$Sn(LIS4[1,4]) = 0$
ProcessS(LIS 4[4,4])	Без изменений					$Sn(LIS4[4,4]) = 0$

Отметим на рисунке 1.15 найденные значимые элементы и образовавшиеся нуль-блоки.

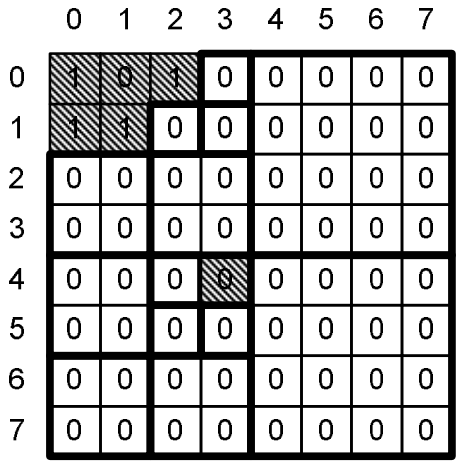


Рисунок 1.15 – Найденные значимые элементы и образовавшиеся нуль-блоки после шага поиска для 5-й битовой плоскости

6 Шаг уточнения.

На данном шаге выводим значения текущей битовой плоскости для элементов списка LSP-P. После вывода значений битовой плоскости в список LSP-P добавляются все элементы списка LSP-C. Список LSP-C становится пустым.

Работа алгоритма продолжается, пока не будут обработаны все битовые плоскости. В этом случае достигается сжатие вейвлет-матрицы без потерь.

1.3.3 Оценка эффективности алгоритма сжатия полутонового изображения

Для оценки эффективности кодека используются среднеквадратичная ошибка *MSE* и пиковое отношением сигнал-шум *PSNR* при определенной степени сжатия *BR*. Значения *MSE, PSNR* и *BR* определяются в соответствии с выражениями:

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}\left(x_i - \tilde{x}_i\right)^2, \tag{1.36}$$

$$PSNR = 10\log_{10}\left(\frac{\left(2^{BD}-1\right)^2}{MSE}\right), \tag{1.37}$$

$$BR = BD/CR, \tag{1.38}$$

где $x_i, \tilde{x}_i$ – исходное и восстановленное значения *i*-го пиксела изображения;

n – общее число пикселей изображения;

BD – битовая глубина (bit depth – число бит на пиксель (bpp) для исходного изображения);

$CR = n \cdot BD/V$ – коэффициент сжатия (Compression Ratio);

V – объем сжатых данных в битах.

2 ЛАБОРАТОРНОЕ ЗАДАНИЕ

В папке с программным обеспечением для лабораторной работы находится файл «*task.pdf*». Для выполнения лабораторного задания следуйте указаниям, приведенным в данном файле.

3 СОДЕРЖАНИЕ ОТЧЕТА

1 Титульный лист.

2 Цель работы.

3 Блок-схема обобщенного алгоритма кодера.

3 Блок-схема прямого и обратного лифтинг вейвлет-преобразования.

4 Блок-схемы кодера и декодера одного из алгоритмов компрессии вейвлет-матрицы.

5 Полученные численные результаты.

6 Выводы.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1 Схематично изобразите гистограмму полутонового изображения. Сравните ее с гистограммой изображения в вейвлет-области, представленной на рисунке 1.2.

2 За счет чего лифтинг вейвлет-преобразование является более быстрым по сравнению с преобразованием на основе банка фильтров?

3 Чем определяется максимально возможное количество уровней разложения вейвлет-матрицы?

4 Какой из типов расширения последовательности при лифтинг вейвлет-преобразовании является наиболее оптимальным?

5 В чем заключается суть прогрессивной передачи информации?

6 За счет чего происходит компрессия вейвлет-матрицы?

7 Различны ли структуры битового потока алгоритмов SPIHT и SPECK?

8 Можно ли избавиться от сортировки списка LIS в алгоритме SPECK? Если да, то каким образом?

9 Для чего в алгоритме SPECK используется множество I? Можно ли модифицировать алгоритм для работы только с множествами типа S?

10 Можно ли оптимизировать передачу информации о значимости в алгоритмах SPIHT и SPECK? Если да, то каким образом?

ЛИТЕРАТУРА

1 Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

2 Joshi, R. L. Comparison of multiple compression cycle performance for JPEG and JPEG 2000 / R. L. Joshi, M. Rabbani, M. Lepley // Proc. of the SPIE, San Diego, CA, USA, July/August 2000. – Vol. 4115. – P. 492–501.

3 Said, A. A New, fast, and efficient codec based on set partitioning in hierarchal trees / A. Said, W. A. Pearlman // IEEE Trans. on Circuits and Systems for Video Technology. – 1996. – Vol. 6. – P. 243–250.

4 Islam, A. Set partitioned sub-block coding (speck) / A. Islam, W. A. Pearlman // ISO/IEC/JTC1/SC29, WG1. – 1998. – №873 – P. 312–326.

5 ICER on Mars: Wavelet-based image compression for the Mars exploration rovers // IND Technology: Science News. – 2002. – Vol. 15. – P. 15–19.

6 Борискевич, А. А. Метод масштабируемого вложенного кодирования изображений на основе иерархической кластеризации вейвлет-структур / А. А. Борискевич, В. Ю. Цветков // Доклады НАН Беларуси. – 2009. – Т. 53, №3. – С. 38–48.

7 Борискевич, А. А. Оценка влияния модификации вейвлет-коэффициентов на сжатие медиаданных / А. А. Борискевич, В. Ю. Цветков // Доклады БГУИР, 2006. – №4 (16). – С. 17–24.

Св. план 2011, поз. 103

Учебное издание

Борискевич Анатолий Антонович
Цветков Виктор Юрьевич
Полещук Павел Леонидович
Борискевич Илья Анатольевич

АЛГОРИТМЫ СЖАТИЯ ПОЛУТОНОВЫХ ИЗОБРАЖЕНИЙ

МЕТОДИЧЕСКОЕ ПОСОБИЕ
по курсу
«Цифровая обработка речи и изображений»
для студентов специальности
«Сети телекоммуникаций»
всех форм обучения

Редактор *Е. Н. Батурчик*
Корректор *Е. И. Герман*
Компьютерная правка, оригинал-макет *А. А. Лысеня*

Подписано в печать 20.06.2013. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. . Уч.-изд. л. 1,9. Тираж 50 экз. Заказ 323.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6