Кодирование изображений с последующим возможным оптимальным декодированием

А. В. ШОКУРОВ
Московский государственный университет им. М. В. Ломоносова

УДК 512.643.8+519.6

Ключевые слова: кодирование изображений, сжатие изображений, многомасштабное пространственное представление, вейвлет-преобразование, постепенная передача.

Аннотация

На практике часто необходимо распаковывать не всё изображение, а только некоторую его часть. Подавляющее большинство методов при запросе требуемого фрагмента, в частности небольшого, вынужденно распаковывают изображение целиком, тратя лишние ресурсы, вследствие чего эти алгоритмы ограничены в применении, так как на используемой системе может оказаться недостаточное количество ресурса-памяти для декодирования изображения целиком.

В данной статье предлагается новый метод кодирования SS-PIHT, позволяющий распаковывать только необходимую часть изображения. При этом память, потребляемая алгоритмом, сравнима по порядку с памятью, необходимой под фрагмент изображения, а не под целое изображение. Помимо этого, предлагаемый метод SS-SPIHT позволяет извлекать фрагменты при различных масштабах. Этот факт позволяет интерактивно анализировать громадные изображения, с разрешением по каждому из измерений, достигающим нескольких тысяч, а то и сотен тысяч точек, на устройстве с малым количеством памяти, например на наладоннике.

Abstract

A. V. Shokurov, Image coding with afterwards possible optimal decoding, Fundamentalnaya i prikladnaya matematika, vol. 13 (2007), no. 5, pp. 225—255.

Instead of decompressing the whole image in practice it is often necessary to decompress a certain part of it. Most methods upon the request of certain image fragment, in particular a small one, have to at first decompress the whole image, thereby wasting excess memory resources. Therefore, these methods have a limited use, since there might not be enough memory resources to decompress the whole image on the hardware being used. The new coding method SS-SPIHT presented in the paper permits the decompression of only the needed image fragment. Herewith, the amount of memory used by the algorithm is comparable on the order to the amount of memory used by the image fragment, and not the image in whole. Aside from this, these fragments can be extracted at various scales by the proposed method SS-SPIHT. Interactive analysis of huge images is possible due to this feature, i.e., viewing images the resolution of which in any dimension reaches several thousand or even hundred thousand points on a low memory device, for instance, on a palm-held.

Фундаментальная и прикладная математика, 2007, том 13, № 5, с. 225—255.

Введение

Традиционно основной целью сжатия изображений было максимально компактное хранение соответствующих данных при сохранении заданного качества изображения. При этом при декодировании для восстановления изображения анализировались все упакованные данные. При загрузке изображений по сети, в том числе через Интернет, такой подход оказался неудобным, так как для просмотра изображения приходилось ждать, когда все данные, относящиеся к изображению, будут перекачены. В связи с этим остро встала проблема частичного восстановления изображения по частичным данным, например по начальным. В идеале хотелось бы уже при приёме первых байтов иметь представление о пересылаемом изображении, а при дальнейшем приёме данных изображение бы уточнялось и обновлялось. Этот метод получил название прогрессивной передачи изображений. Он может быть использован, например, при сжатии по стандарту JPEG [3].

С развитием техники стало актуальным передавать изображение необходимого, а не произвольного разрешения, так как не все устройства могут отобразить изображение исходного разрешения. В связи с этим появился иерархический метод хранения изображений, в котором по изображению строится иерархия изображений, каждое последующее изображение вдвое меньше предыдущего. Размер наименьшего изображения выбирается исходя из возможных запросов. При этом для передачи изображения необходимого масштаба сначала передаётся изображение самого низкого разрешения, а потом добавочные «изображения» для постепенного восстановления изображения необходимого разрешения. Данный метод также может быть использован при сжатии по стандарту JPEG.

Вскоре понадобились методы, позволяющие сжимать изображения таким образом, чтобы впоследствии можно было извлекать не только изображение необходимого масштаба, но и произвольную часть масштабированного изображения. Извлечение должно выполняться без дополнительных расходов на декомпрессию всего изображения или даже на декомпрессию изображения требуемого масштаба. Это стало возможным благодаря алгоритму EBCOT [8], который и был впоследствии использован в JPEG2000 [4].

Многие другие методы не позволяли кодировать изображение таким способом, чтобы потом можно было с лёгкостью вырезать произвольные фрагменты изображения. Более того, большинство этих методов проигрывали по степени сжатию EBCOT. SPIHT [7] — один из методов, позволяющий сжимать изображения на равных с EBCOT. Основной недостаток SPIHT и, возможно, одна из причин, по которой он не стал основой для JPEG2000, — это то, что алгоритм не позволяет передать только те данные, которые относятся к требуемому масштабу изображения, тем более относящиеся к определённому фрагменту масштабированного изображения. Первая проблема была успешно решена в FS-SPIHT [2]. Вторая задача была частично решена в работе [10], а именно стало возможным извлекать фрагменты только из исходного изображения, а не из изображения произвольного масштаба.

В данной работе предлагается метод, развивающий идеи как алгоритма FS-SPIHT, так и работы [10], позволяющий передавать не только изображение необходимого масштаба, но и фрагменты изображений необходимого масштаба.

О предлагаемом алгоритме

Алгоритм, о котором идёт речь в данной статье, основывается на некоторых математических понятиях. В частности, для его построения понадобятся вейвлет-преобразования. Вейвлет-преобразования позволяют описать изменение числовых характеристик изображения и имеют локальный характер. Благодаря этому для восстановления фрагмента изображения нам необходимы не все числовые характеристики изображения, а только те, которые относятся к окрестности требуемого фрагмента. Кроме того, такое преобразование позволяет нам получить изображение при различных масштабах, т. е. мы не только можем вырезать фрагмент исходного изображения, но и этот же фрагмент при различных масштабах.

Организация статьи

Статья организована следующим образом. В разделе 1 излагается математическое обоснование данного алгоритма. Речь пойдёт о вейвлет-преобразованиях. В частности, будет показано, как именно выбираются вейвлет-коэффициенты преобразованного изображения, необходимые для восстановления требуемого фрагмента изображения. В разделе 2 речь пойдёт о самом алгоритме SS-SPIHT. Будет показано, как изображение кодируется и как декодировать произвольный фрагмент. В разделе 3 анализируется работа алгоритма. В заключение показано преимущество предлагаемого метода SS-SPIHT в сравнении с уже существующими.

1. Вейвлеты

При кодировании изображения мы преследуем несколько целей. Во-первых, нам нужно уметь извлекать фрагменты исходного изображения, не пользуясь всеми кодированными данными. Для этого зависимость между исходными и преобразованными числовыми характеристиками должна быть локальна, т. е. любой фрагмент исходного изображения восстанавливается по некоторому фрагменту преобразованного изображения. Во-вторых, предлагаемый метод должен позволять извлекать не только фрагмент исходного изображения, но и этот фрагмент в различных масштабах. Это, в свою очередь, означает, что метод кодирования основан на иерархическом представлении изображений.

Этим двум условиям удовлетворяет, например, вейвлет-преобразование. Возможны и другие преобразования, но на самом деле они все будут в той или иной форме вейвлет-преобразованиями [5]. Помимо указанных свойств, вейвлет-пре-

образования просты в вычислениях, что позволяет использовать их на практике. Например, они используются в JPEG2000.

В данном разделе будет рассказано об эффективном использовании вейвлетов. На этом базируется предлагаемый метод кодирования изображений SS-SPIHT, описанный во втором разделе.

1.1. Оптимизация частичного линейного преобразования

Рассмотрим линейное преобразование из $\mathbb{R}^n$ в $\mathbb{R}^m$. Пусть $v \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ — векторы и $u = Av$ — линейное преобразование с матрицей A координатных пространств $\mathbb{R}^n$ и $\mathbb{R}^m$.

Обычно в линейной алгебре весь вектор v преобразовывается в вектор u, т. е. по компонентам вектора v вычисляются все компоненты вектора u. В общем случае тут нечего оптимизировать, так как даже для вычисления одной компоненты вектора u могут понадобиться все компоненты вектора v.

Допустим, что матрица имеет «хороший» вид и нам нужно вычислить не весь вектор u, а только его часть. Зная, какие именно компоненты u надо вычислить, можно сократить и вектор v. Под оптимизацией линейного преобразования мы понимаем нахождение минимальной части вектора v, по которой можно вычислить необходимую часть вектора u.

В этом разделе рассматривается вначале общий случай линейного преобразования, а потом конкретное линейное преобразование, используемое на практике, а именно обратное биортогональное вейвлет-преобразование $5/3$.

Речь пойдёт об обратном вейвлет-преобразовании, так как оптимизация нам понадобится при обратном к кодированию процессе. А именно, при декодировании по запросу пользователя будет вычислено, какие именно кодированные данные надо передавать. Этот процесс будет изложен более подробно в конце этого раздела.

1.2. Линейные преобразования

Как и выше, $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$ — векторы и $u = Av$ — линейное преобразование. Для удобства изложения нумерация компонент векторов и матриц начинается с 0. Тогда линейное преобразование можно записать в виде

$$\begin{pmatrix} u_0 \\ \dots \\ u_i \\ \dots \\ u_{m-1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & \dots & a_{0,n-1} \\ & \dots & \\ a_{i,0} & a_{i,j} & a_{i,n-1} \\ & \dots & \\ a_{m-1,0} & \dots & a_{m-1,n-1} \end{pmatrix} \begin{pmatrix} v_0 \\ \dots \\ v_j \\ \dots \\ v_{n-1} \end{pmatrix},$$

где компонента u_i преобразованного вектора определяется равенством

$$u_i = \sum_{k=0}^{n-1} a_{i,k} v_k = \sum_{\substack{k=0 \\ a_{i,k} \neq 0}}^{n-1} a_{i,k} v_k.$$

Для вычисления компоненты u_i нужны только те компоненты v_k вектора v, которые умножаются на ненулевые компоненты $a_{i,k}$ матрицы A. Обозначим это множество индексов через

$$\mathrm{SupRows}(A, \{i\}) = \{k \mid a_{i,k} \neq 0\}.$$

Если помимо компоненты u_{i_1} нужны и компоненты $u_{i_2}, \ldots, u_{i_l}$, то надо взять объединение всех $\mathrm{SupRows}(A, \{i_p\})$, а именно

$$\mathrm{SupRows}(A, \{i_1, \ldots, i_l\}) = \bigcup_{p=1}^{l} \mathrm{SupRows}(A, \{i_p\}).$$

Композиции линейных преобразований соответствует произведение матриц: $u = BAv$, при этом

$$\mathrm{SupRows}(BA, \{i_1, \ldots, i_l\}) = \mathrm{SupRows}\big(A, \mathrm{SupRows}(B, \{i_1, \ldots, i_l\})\big). \qquad (1)$$

1.3. Случай свёртки

По вектору $v \in \mathbb{R}^n$ построим *симметрично-периодический* вектор $v_{\{.\}}$ значимой длины n, определив его компоненты для произвольного целого числа i следующим образом:

$$v_{\{i\}} = \begin{cases} v_i, & i \in [0, n-1], \\ v_{-i}, & -i \in [0, n-1], \\ v_{\{i-k\times \mathrm{len}\}}, & i \in [1-n-k\times \mathrm{len}, n-1-k\times \mathrm{len}], \end{cases}$$

где $\mathrm{len} = 2n-1$ — его период. Свёртка вектора $v_{\{.\}}$ с ядром $h \in l^1[\mathbb{Z}]$ есть вектор $u \in \mathbb{R}^n$ с компонентами

$$u_p = (h \star v)_p = \sum_{n=-\infty}^{+\infty} h_{p-n} v_{\{n\}}. \qquad (2)$$

Свёртка линейна по v и может быть задана матрицей A, при этом преобразование (2) записывается в виде $u = Av$.

В данном разделе предполагается, что ядро h сосредоточено в окрестности индекса 0 радиуса меньше n. Тогда существуют такие целые числа $l, r \in \mathbb{Z}$, $l \leqslant 0$, $r \geqslant 0$, $\|l\|, \|r\| < n$, что при $i < l$ или $i > r$ выполняется равенство $h_i = 0$. Иначе считаем, что $\mathrm{SupRows}(A, \{i\}) = \{0, \ldots, n-1\}$.

Найдём явный вид этой матрицы.

Рассмотрим первый случай, когда h равняется нулю при отрицательных индексах, т. е. $l = 0$ и $r \geqslant 0$. Из формулы свёртки (2) следует, что при $r \leqslant p \leqslant n-1$ свёртка эквивалентна свёртке с обычным вектором v. В случае $0 \leqslant p < r$ свёртка имеет особенность в окрестности индекса 0, а именно отрицательные индексы отражаются относительно 0 на положительные. При нулевом индексе $p = 0$ всё ядро отражается, а при увеличении индекса p количество отрицательных индексов уменьшается. Этот процесс будет происходить до тех пор, пока всё

ядро не окажется с положительными индексами, а это будет при $p = r$. Отсюда получаем, что матрица A^+ свёртки имеет вид

$$\begin{pmatrix} h_0 & h_1 & h_2 & \dots & \dots & h_r & 0 & \dots & 0 \\ h_1 & h_0+h_2 & h_3 & \dots & h_r & 0 & \dots & \dots & 0 \\ & & & \dots & \dots & & & & \\ h_r & h_{r-1} & \dots & \dots & h_1 & h_0 & 0 & \dots & 0 \\ 0 & h_r & h_{r-1} & \dots & h_1 & h_0 & 0 & \dots & 0 \\ & & & \dots & & & & & \\ 0 & \dots & 0 & h_r & h_{r-1} & \dots & h_1 & h_0 & 0 \\ 0 & \dots & \dots & 0 & h_r & h_{r-1} & \dots & h_1 & h_0 \end{pmatrix}.$$

Первый ненулевой элемент в строке i и последний ненулевой элемент в строке j соответственно равны

$$S(A^+, i) = \begin{cases} 0, & i \leqslant r, \\ i - r, & i > r, \end{cases}$$

$$E(A^+, j) = \begin{cases} \max(r-j, j), & j \leqslant r, \\ j, & j > r \end{cases} = \begin{cases} r-j, & 2j \leqslant r, \\ j, & 2j > r. \end{cases}$$

Случай, когда h равно нулю для положительных коэффициентов, выводится аналогичным образом с учётом того, что особенность теперь сосредоточена в окрестности индекса $n-1$ и, кроме того, индекс n отражается на индекс $n-1$. Матрица A^- и соответствующие функции $S(A^-, i)$ и $E(A^-, j)$ имеют вид

$$\begin{pmatrix} h_0 & h_{-1} & h_{-2} & \dots & h_l & 0 & \dots & \dots & 0 \\ 0 & h_0 & h_{-1} & h_{-2} & \dots & h_l & 0 & \dots & 0 \\ & & & \dots & \dots & & & & \\ 0 & \dots & 0 & h_0 & h_{-1} & \dots & h_{l+2} & h_l & 0 \\ 0 & \dots & \dots & 0 & h_0 & h_{-2} & \dots & h_{l+1} & h_l \\ & & & \dots & & & & & \\ 0 & \dots & 0 & h_l & h_{l+1} & \dots & h_4 & h_0+h_3 & h_1+h_2 \\ 0 & \dots & \dots & 0 & h_l & h_{l+2} & \dots & h_2 & h_0+h_1 \end{pmatrix},$$

$$S(A^-, i) = \begin{cases} i, & i \leqslant n - (-l), \\ \min(2n - (1+l+i), i), & i > n - l \end{cases} =$$

$$= \begin{cases} i, & 2i \leqslant 2n - (1-l), \\ 2n - (1-l+i), & 2i > 2n - (1-l), \end{cases}$$

$$E(A^-, j) = \begin{cases} j - l, & j < n + l, \\ n - 1, & j \geqslant n + l. \end{cases}$$

Обозначим $\overrightarrow{i \dots j} = \{i, i+1, \dots, j-1, j\}$. Тогда очевидно, что

$$S\left(A^+, \overrightarrow{i \dots j}\right) = S(A^+, i), \quad E\left(A^-, \overrightarrow{i \dots j}\right) = E(A^-, j).$$

Для остальных функций, учитывая тип экстремума, нетрудно вывести

$$S\Big(A^-, \overrightarrow{i\ldots j}\Big) = \min\{S(A^-, i), S(A^-, j)\},$$
$$E\Big(A^+, \overrightarrow{i\ldots j}\Big) = \max\{E(A^+, i), E(A^+, j)\}.$$

В итоге получаем формулу для $\mathrm{SupRows}\Big(A, \overrightarrow{i\ldots j}\Big)$:

$$\mathrm{SupRows}\Big(A, \overrightarrow{i\ldots j}\Big) = \{i_{\mathrm{start}}, i_{\mathrm{start}+1}, \ldots, i_{\mathrm{end}-1}, i_{\mathrm{end}}\}, \tag{3}$$

где

$$\mathrm{start} = \min\Big(S\Big(A^-, \overrightarrow{i\ldots j}\Big), S\Big(A^+, \overrightarrow{i\ldots j}\Big)\Big),$$
$$\mathrm{end} = \max\Big(E\Big(A^-, \overrightarrow{i\ldots j}\Big), E\Big(A^+, \overrightarrow{i\ldots j}\Big)\Big).$$

1.4. Прямое вейвлет-преобразование

Прямое вейвлет-преобразование задаётся двумя фильтрами $h, g \in l^1[\mathbb{Z}]$ [5]. Пусть $u \in \mathbb{R}^{2n}$. Введём обозначение $\bar{h}_p = h_{-p}$. Результатом прямого вейвлет-преобразования являются два вектора $a, d \in \mathbb{R}^n$ вдвое меньшего размера с компонентами, определяемыми равенствами

$$a_p = \sum_{n=-\infty}^{+\infty} h_{n-2p} u_{\{n\}} = \sum_{n=-\infty}^{+\infty} \bar{h}_{2p-n} u_{\{n\}} = (\bar{h} \star u)_{2p}, \tag{4}$$

$$d_p = \sum_{n=-\infty}^{+\infty} g_{n-2p} u_{\{n\}} = \sum_{n=-\infty}^{+\infty} \bar{g}_{2p-n} u_{\{n\}} = (\bar{g} \star u)_{2p}, \tag{5}$$

где вектор a соответствует низкочастотному поддиапазону, а вектор d соответствует высокочастотному поддиапазону вектора u.

Под N-кратным вейвлет-преобразованием понимается N-кратное применение вейвлет-преобразования. Но оно применяется каждый раз не к обоим векторам a и d, а только к вектору a. В результате на каждом шаге вычисляется очередной вектор d, а вектор a каждый раз дробится.

Введём обозначения, позволяющие описать это математически. Пусть вектор $u^i = a^i$ разбивается на два вектора a^{i+1} и d^{i+1} по описанным выше формулам (4) и (5). Тогда результатом N-кратного вейвлет преобразования будет $D_N = \{a^N, d^N, d^{N-1}, \ldots, d^1\}$. Заметим, что для того чтобы все эти вектора вычислялись, вводится ограничение на первоначальный размер вектора u, а именно он должен достаточное количество раз делиться на 2.

С математической точки зрения любое вейвлет-преобразование можно обратить. Для этого по фильтрам h и g строятся фильтры $\tilde{h}$ и $\tilde{g}$:

$$\tilde{h}_p = (-1)^{-p} g_{1-p}, \quad \tilde{g}_p = (-1)^{1-p} h_{1-p}.$$

Тогда вектор u восстанавливается по формуле

$$u_p = \sum_{n=-\infty}^{+\infty} \tilde{h}_{p-2n} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} \tilde{g}_{p-2n} d_{\{n\}}. \tag{6}$$

Легко понять, что и N-кратное вейвлет-преобразование всегда обратимо.

Заметим, что хотя с математической точки зрения все вейвлет-преобразования обратимы, на практике встречаются несколько подходов к приближённому вычислению вейвлет-преобразования, некоторые из которых дают необратимые преобразования.

Во-первых, векторы a и d можно вычислять прямо по формулам (4) и (5). Но так как результатом вейвлет-преобразования являются действительные числа, на компьютере компоненты этих векторов надо тем или иным способом округлять или квантовать. В результате этого действия мы теряем точность значения первоначальных действительных чисел, поэтому при применении формулы обратного вейвлет-преобразования мы не получим в точности исходный вектор u. Вместо этого мы получим некоторое приближение к этому вектору. Поэтому, когда во втором разделе мы будем говорить о постепенном восстановлении изображения, при применении данного подхода к вычислению вейвлет-преобразований исходное изображение в точности никогда не будет получено, оно всегда будет немного отличаться от исходного. Ошибка различия $\mathrm{err}(C - \tilde{C})$, о которой пойдёт речь во втором разделе, будет стремиться не к 0, а к неизбежной минимальной ошибке.

Во-вторых, вейвлет-преобразования можно вычислять, используя «хитрый» метод, основанный на подъёмных схемах (за подробностями направляем читателя к [5]). Этот метод позволяет всегда восстановить в точности исходный вектор u. Однако надо иметь в виду, что это достигается вычислением приближённого вейвлет-преобразования. Возможно восстановление исходного изображения в точности, поэтому ошибка различия $\mathrm{err}(C - \tilde{C})$ действительно стремится к нулю и обращается в нуль.

Также заметим, что обычно два вектора a и d объединяют в один, а именно $u^1 = (a^\top, d^\top)^\top$. Тогда по вектору u^0 строится вектор $u^1 \in \mathbb{R}^{2n}$ того же размера и аналогично по вектору u^1 восстанавливается вектор u^0. Подобным образом записывается и результат действия N-кратного вейвлет-преобразования.

1.5. Обратное вейвлет-преобразование

Как было указано выше, обратное вейвлет-преобразование задаётся двумя фильтрами $\tilde{h}, \tilde{g} \in l^1[\mathbb{Z}]$ [5]. Так как прямое вейвлет-преобразование в этом подразделе встречаться не будет, для упрощения обозначений позволим себе эти векторы обозначать просто h и g соответственно.

Пусть $a, d \in \mathbb{R}^n$ — векторы и вектор $u \in \mathbb{R}^{2n}$ — их обратное вейвлет-преобразование. Тогда компоненты u, как было указано выше (см. (6)), задаются следующим образом:

$$u_p = \sum_{n=-\infty}^{+\infty} h_{p-2n} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} g_{p-2n} d_{\{n\}}. \tag{7}$$

Введём обозначения $u_i^{\text{even}} = u_{2i}$, $u_i^{\text{odd}} = u_{2i+1}$. Тогда

$$u_p^{\text{even}} = u_{2p} = \sum_{n=-\infty}^{+\infty} h_{2p-2n} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} g_{2p-2n} d_{\{n\}} =$$
$$= \sum_{n=-\infty}^{+\infty} h_{p-n}^{\text{even}} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} g_{p-n}^{\text{even}} d_{\{n\}} = (h^{\text{even}} \star a)_p + (g^{\text{even}} \star d)_p.$$

Аналогично

$$u_p^{\text{odd}} = u_{2p+1} = \sum_{n=-\infty}^{+\infty} h_{2p-2n+1} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} g_{2p-2n+1} d_{\{n\}} =$$
$$= \sum_{n=-\infty}^{+\infty} h_{p-n}^{\text{odd}} a_{\{n\}} + \sum_{n=-\infty}^{+\infty} g_{p-n}^{\text{odd}} d_{\{n\}} = (h^{\text{odd}} \star a)_p + (g^{\text{odd}} \star d)_p.$$

Таким образом, преобразование (7) сводится к сумме свёрток. Поэтому применимы результаты, полученные в предыдущем разделе.

Несмотря на то, что имеется алгоритм вычисления, в общем случае формула $\text{SupRows}\left(A, \overrightarrow{i \ldots j}\right)$ громоздка. Однако при конкретных r, l можно записать компактную формулу явно в случае произвольных отвечающих им фильтрах h и g. На практике [4] часто встречается следующий случай: биортогональное вейвлет-преобразование $5/3$. Обратное преобразование задаётся фильтрами h и g по таблице:

i	-1	0	$+1$	$+2$	$+3$
h_i	0,500	1,000	0,500	0	0
g_i	$-0{,}125$	$-0{,}250$	0,750	$-0{,}250$	$-0{,}125$

В этом случае для h^{even}, h^{odd}, g^{even}, g^{odd} получаем следующие значения инвариантов r, l:

	h^{even}	h^{odd}	g^{even}	g^{odd}
l	0	-1	0	-1
r	0	0	1	1

(8)

Поэтому мы в настоящей работе ограничиваемся следующими тремя случаями: $l = -1$, $r = 0$; $l = 0$, $r = 1$; $l = 0$, $r = 0$.

Если $l = -1$, $r = 0$, то непосредственным вычислением получаем, что $S(A^-, i) = i$ и $E(A^-, j) = j + 1$. Аналогично при $l = 0$, $r = 1$ получаем,

что $S(A^+, i) = i - 1$ и

$$E(A^+, j) = \begin{cases} 1, & j = 0, \\ j, & j > 1. \end{cases}$$

При $l = 0$, $r = 0$ получаем, что $S(A^+, i) = i$ и $E(A^+, j) = j$.

Применив эти формулы к ядрам h_{even}, h_{odd}, g_{even}, g_{odd}, получаем следующий результат.

Предложение 1. *Пусть h и g — такие фильтры, что соответствующие им векторы h_{even}, h_{odd}, g_{even}, g_{odd} имеют инварианты l, r, как в таблице* (8)*. Тогда*

$$S\left(h^{\mathrm{even}}, \overrightarrow{i \dots j}\right) = i, \quad E\left(h^{\mathrm{even}}, \overrightarrow{i \dots j}\right) = j,$$
$$S\left(h^{\mathrm{odd}}, \overrightarrow{i \dots j}\right) = i, \quad E\left(h^{\mathrm{odd}}, \overrightarrow{i \dots j}\right) = j + 1,$$
$$S\left(g^{\mathrm{even}}, \overrightarrow{i \dots j}\right) = i - 1, \quad S\left(g^{\mathrm{odd}}, \overrightarrow{i \dots j}\right) = i - 1, \quad E\left(g^{\mathrm{odd}}, \overrightarrow{i \dots j}\right) = j + 1,$$
$$E\left(g^{\mathrm{even}}, \overrightarrow{0 \dots 0}\right) = 1, \quad E\left(g^{\mathrm{even}}, \overrightarrow{i \dots j}\right) = j.$$

Следствие 1. *В условиях предложения* 1 *выполнены следующие равенства*:

$$S\left(h, \overrightarrow{i \dots j}\right) = \left\lfloor \frac{i}{2} \right\rfloor, \quad E\left(h, \overrightarrow{i \dots j}\right) = \left\lfloor \frac{j+1}{2} \right\rfloor,$$
$$S\left(g, \overrightarrow{i \dots j}\right) = \left\lfloor \frac{i}{2} \right\rfloor - 1, \quad E\left(g, \overrightarrow{0 \dots 0}\right) = 1, \quad E\left(g, \overrightarrow{i \dots j}\right) = \left\lfloor \frac{j+1}{2} \right\rfloor.$$

Теперь, используя формулы (3), можно вычислить $\mathrm{SupRows}(h, I^{2n})$ и $\mathrm{SupRows}(g, I^{2n})$.

1.6. Произвольное вейвлет-преобразование

Алгоритм кодирования изображений SS-SPIHT, приведённый в этой статье, применим не только к указанному биортогональному вейвлет-преобразованию, но и к другим вейвлет-преобразованиям. Поэтому в дальнейшем будем считать, что у нас произвольное вейвлет-преобразование, удовлетворяющее ограничениям, указанным далее при кодировании.

Будем пользоваться тем фактом, что для произвольного интервала индексов $I^{2n} = \overrightarrow{i \dots j}$ вектора $u \in \mathbb{R}^{2n}$ существуют функции, с помощью которых вычисляются достаточные интервалы индексов $I_a^n = \overrightarrow{i_a \dots j_a}$ и $I_d^n = \overrightarrow{i_d \dots j_d}$ для векторов a и d соответственно, а именно $i_a = S(h, I^{2n})$, $j_a = E(h, I^{2n})$, $i_d = S(g, I^{2n})$, $j_d = E(g, I^{2n})$. (В предыдущем разделе был описан способ получения этих формул на примере конкретного вейвлет-преобразования.)

Введём следующие обозначения:

$$I_a^n := \mathrm{SupRows}(h, I^{2n}), \quad I_d^n := \mathrm{SupRows}(g, I^{2n}). \tag{9}$$

1.7. Композиция обратных вейвлет-преобразований

Теперь рассмотрим композицию вейвлет-преобразований. Для удобства пусть $n = q2^k$, где $q \in \mathbb{N}$ — произвольное натуральное число. Вейвлет-преобразование W_p кратности p, как было указано ранее, раскладывает вектор $u = u^0 = a^0 \in \mathbb{R}^n = \mathbb{R}^{q2^k}$ на множество векторов $D_p = \{a^p, d^p, d^{p-1}, \ldots, d^1\}$. Заметим, что $a^s, d^s \in \mathbb{R}^{q2^{k-s}}$. Вектор a^0 восстанавливается последовательным применением, начиная с $l = p$, следующей формулы:

$$a_p^{l-1} = \sum_{n=-\infty}^{+\infty} h_{p-2n} a_{\{n\}}^l + \sum_{n=-\infty}^{+\infty} g_{p-2n} d_{\{n\}}^l. \tag{10}$$

Найдём формулу для $R_p(I^n) := \mathrm{SupRows}(W_p^{-1}, I^n)$. Заметим, что предыдущие формулы давали множество индексов компонент векторов a и d. Для удобства изложения последующего материала введём общую нумерацию компонент векторов. А именно, пусть вектор u^p содержит последовательно все векторы множества D_p:

$$u^{p\top} = \left(a^{p\top}, d^{p\top}, d^{p-1\top}, \ldots, d^{1\top}\right), \quad u^p \in \mathbb{R}^n.$$

Непосредственно из предыдущего обозначения и формул (9) следует, что для необходимого множества индексов I^n вектора u^0 достаточным множеством индексов $R_1(I^n)$ вектора u^1 будет $R_1(I^n) = I_a^{n/2} \cup \{I_d^{n/2} + n/2\}$, где сложение числа $n/2$ с множеством индексов $I_d^{n/2}$ обозначает прибавление этого числа к каждому числу данного множества. Из определения также следует, что $R_0(I^n) = I^n$.

Заметим, что предыдущие формулы (9) могли давать компоненты, выходящие за пределы индексации вектора, а именно компоненты больше максимального индекса или меньше 0. Так как использование таких компонент в расширенной версии вектора может привести к включению лишних компонент, перед добавлением числа $n/2$ к множеству все такие компоненты удаляются.

Пользуясь формулой (1), получаем следующий результат.

Предложение 2. *В условиях, сформулированных выше, множество достаточных компонент $R_p(I^n)$ вектора $u^p \in \mathbb{R}^n$, по которым можно восстановить, используя формулу* (10)*, необходимые компоненты I^n вектора $u = u^0$, вычисляется по формуле*

$$R_p(I^n) = R_{p-1}(I_a^{n/2}) \cup \left\{I_d^{n/2} + \frac{n}{2}\right\}.$$

1.8. Изображения

Прежде чем вводить определение вейвлет-преобразования изображения, определим, что такое изображение. С математической точки зрения для начала вводится понятие непрерывного изображения. Это непрерывная функция на бесконечной двумерной плоскости, значение которой в каждой точке задаёт интенсивность. Такое изображение призвано имитировать взгляд на реальный

мир: у наблюдателя, смотрящего на изображение, создаётся впечатление, что он действительно смотрит из окна дома или что перед ним действительно находится девушка.

Изображения, определённые с математической точки зрения, не годятся в практике, так как в компьютерах невозможно представлять бесконечные объекты, по крайней мере непосредственно. Поэтому накладываются некоторые ограничения на изучаемые изображения, позволяющие сократить необходимый объём памяти. Так, например, вместо бесконечной плоскости используется ограниченная область, например прямоугольник. Вне этого прямоугольника изображение нас не интересует, поэтому его можно не хранить. Но даже в этом случае изображение не поместится в память компьютера, так как количество точек в этом прямоугольнике бесконечно. Поэтому вводится ограничение, касающееся характера изменений в изображении: часто́ты изображения ограничиваются константой. Тогда по теореме Котельникова [6] вместо непрерывного изображения можно использовать дискретное. По дискретному изображению можно восстановить непрерывное, т. е. на этом прямоугольнике вводится сетка и только узлы этой сетки, которые называются пикселями, необходимо сохранять.

Но даже в этом случае такое дискретное изображение нельзя сохранить на компьютере, так как каждое из чисел в узлах является действительным, а компьютер оперирует конечным набором чисел. Поэтому, во-первых, вводится ограничение на максимальную интенсивность и, во-вторых, эти числа квантуются. Ещё стараются, чтобы значения получившихся чисел помещались в стандартные регистры процессора, т. е. в двоичном исчислении не превосходили 32 разрядов, или 32 битов.

1.9. Двумерное вейвлет-преобразование, или вейвлет-преобразование изображений

В данной статье для простоты изложения предполагается, что изображения квадратные, причём ширина является степенью двойки. Тогда изображение можно представить в виде квадратной матрицы I. Значение интенсивности в точке с индексами (i, j) обозначается элементом $p_{i,j}$ матрицы I.

Двумерное вейвлет-преобразование состоит из поочередного одномерного вейвлет-преобразования строк и столбцов этой матрицы. Сначала выполняются одномерные вейвлет-преобразования каждой строки в отдельности, преобразованная строка записывается на прежнее место. Элементы нумеруются способом, указанным в предыдущих разделах. Далее вейвлет-преобразования применяются ко всем столбцам. В результате изображение разбивается на четыре равные части, как показано на рис. 1. На рисунке также показано стандартное обозначение каждого из квадрантов.

Если не оговорено противное, под *N-кратным двумерным вейвлет-преобразованием* понимается применение N раз двухмерного вейвлет-преобразования, причём очередное двумерное вейвлет-преобразование применяется к младшей

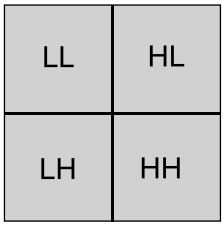

Рис. 1. Однократное применение двумерного вейвлет-преобразования к квадратному изображению. На изображении показаны принятые стандартные обозначения квадрантов преобразованного изображения: LL, LH, HL, HH. Квадрант LL соответствует низкочастотным вейвлет-коэффициентам, HH — высокочастотным вейвлет коэффициентам (буква L означает Low, H — High)

четверти матрицы (в обозначениях рис. 1 этот квадрант обозначается LL). В итоге N-кратное преобразование выглядит так, как показано на рис. 2 (при $N = 3$):

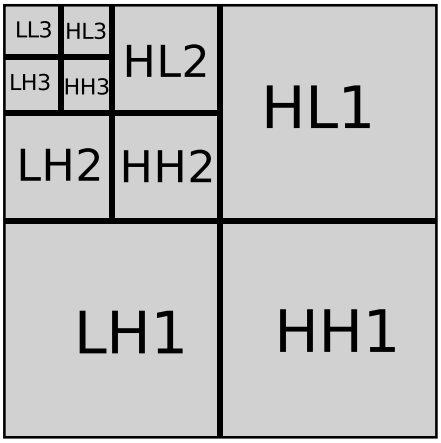

Рис. 2. Трёхкратное применение двумерного вейвлет-преобразования. Показаны принятые стандартные обозначения квадрантов изображения. Квадранты N-кратного двумерного вейвлет-преобразования имеют аналогичное обозначение

Обратное двумерное вейвлет-преобразование рекурсивно восстанавливает младший квадрант. В случае, показанном на рис. 2, для получения (восстановления) нового квадранта LL2. используются квадранты LL3, LH3, HL3 и HH3. Далее для восстановления квадранта LL1 используются квадранты LL2, LH2, HL2, HH2, и т. д. Аналогично выполняется N-кратное обратное вейвлет-преобразование.

Заметим, что указанное преобразование является иерархическим, т. е. если мы при применении обратного вейвлет-преобразования вычислим не все уровни,

а меньшее количество, то в квадранте LLk образуется уменьшенная копия изображения, как показано на рис. 3. В частности, если мы вообще не используем обратное вейвлет-преобразование, то самый младший квадрант тоже является уменьшенной копией изображения.

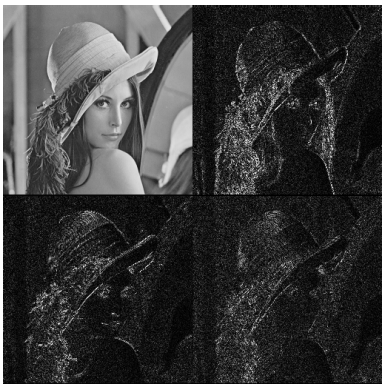

Рис. 3. Однократное применение двумерного вейвлет-преобразования или применение $(N-1)$-кратного обратного вейвлет-преобразования к изображению, полученному N-кратным вейвлет-преобразованием

Благодаря этому свойству обратное вейвлет-преобразование позволяет вырезать фрагменты изображений при различных масштабах. Однако заметим, что, во-первых, доступные масштабы определяются количеством уровней вейвлет-преобразования и, во-вторых, масштабы не произвольны, а отличаются увеличением в два раза.

Теперь осталось определить достаточные области на изображении, которые используются для восстановления фрагментов изображения.

1.10. Достаточная область изображения

По формулам, указанным ранее, определяются достаточные компоненты преобразованного изображения для восстановления необходимого фрагмента изображения. Так как на обработку компонент по отдельности уходит много времени, в практических целях компоненты объединяются в группы. Например, преобразованное изображение разбивается на квадраты фиксированного размера, называемые блоками. Пример разбиения показан на рис. 4 (это разбиение будет использоваться в разделе 2, когда речь пойдёт о кодировании изображений).

В общем случае по указанным компонентам определяются блоки, которые содержат эти компоненты. В дальнейшем обрабатываются эти блоки, а не компоненты, например, на запрос пользователя пересылаются блоки, а не компоненты.

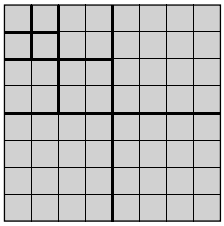

Рис. 4. Трёхкратное применение двумерного вейвлет-преобразования. На изображении показано разбиение на блоки, принятое в этой статье

2. Кодирование изображений

2.1. Введение

В разделе 2.2 напоминается известный метод кодирования изображений SPIHT и вводятся новые понятия, которые будут использоваться в разделе 2.3 при изложении усовершенствованного метода SPIHT — SS-SPIHT (Spatially Scalable SPIHT).

В разделе 1.8 было пояснено понятие изображения. Изображение можно различными способами сохранить на компьютере, Для просмотра изображения данные восстанавливаются с компьютера. Различные способы сохранения изображения различаются методом кодирования исходных данных изображения. Сохранённые данные называются также кодированным потоком.

Под кодированием понимается определённый способ переработки исходных данных, дающий на выходе кодированные данные. В алгоритмах, приведённых ниже, используется команда «вывести . . . » для вывода 0 или 1 в кодированный поток.

2.2. Подразбиение иерархического дерева на части

Напомним известный алгоритм кодирования изображений SPIHT (Set Partitioning In Hierarchical Trees). Целью данного метода является эффективное кодирование числовых характеристик, т. е. вейвлет-коэффициентов, преобразованного изображения. Метод оптимален именно для изображений, т. е. для кодирования случайного набора чисел он не годится.

Оптимальный метод был выявлен при анализе преобразованных изображений. Было замечено, что если вейвлет-коэффициент преобразованного изображения близок к нулю, то и коэффициенты его поддерева близки к нулю. И обратно, если коэффициент значительный, то и соответствующие коэффициенты поддерева значительные. Примеры изображений, на которых это видно, приведены на рис. 3 и 5.

Метод основан на следующих принципах:

- передача коэффициентов в порядке уменьшения их «значимости»;
- побитовая передача коэффициентов;
- эффективная передача упорядочения коэффициентов.

2.2.1. Постепенная передача изображения

Постепенная передача изображения заключается в передаче изображения с постепенным увеличением его качества по мере получения новых данных. Пиксели первоначального приближения к изображению инициализируются нулём. Далее желательно передавать данные в порядке их вклада в итоговое изображение. Опишем это математически.

К матрице I изображения с компонентами $p_{i,j}$ применяется двумерное N-кратное вейвлет-преобразование, описанное в разделе 1.9. В результате этого преобразования получаем матрицу C, размер которой совпадает с матрицей исходного изображения I. Элементы $c_{i,j}$ матрицы C являются *коэффициентами вейвлет-преобразования* в координате (i, j) преобразованного изображения. Как было отмечено в предыдущем разделе, для практического применения мы будем считать, что коэффициенты — целые числа, причём представимые в целочисленной арифметике компьютера. Например, в бинарном представлении используется не более 32 битов.

При постепенной передаче изображения первое приближение преобразованного изображения $\tilde{C}$ полагается равным нулю. В ходе работы алгоритма коэффициенты $\tilde{c}_{i,j}$ приближения преобразованного изображения $\tilde{C}$ обновляются. В любой момент алгоритм обновления можно остановить и применить к приближению преобразованного изображения обратное N-кратное вейвлет-преобразование, тем самым получая изображение $\tilde{I}$, являющееся приближением исходного изображения I. По мере получения новых данных изображение $\tilde{I}$ приближается к исходному изображению I. В качестве меры различия между изображениями используется квадратическая разница

$$\operatorname{err}(I - \tilde{I}) = \sum_{x=0,\ y=0}^{x=N-1,\ y=N-1} (p_{x,y} - \tilde{p}_{x,y})^2.$$

Разность между исходным и приближённым изображением стремится к нулю (см. замечание в разделе 1.4) по мере получения новых данных.

Но для нас важно не просто стремление к нулю, а определённый характер этого стремления. Нас интересует максимально быстрое стремление к нулю, поэтому очевидно, что лучше начинать передачу с той информации, которая даст наибольшее снижение ошибки между исходным и приближённым изображением.

Информация передаётся побитно, поэтому нужно, чтобы каждый последующий бит «стоил» не больше чем предыдущий, т. е. ошибка различия $\operatorname{err}(I - \tilde{I})$

снижалась бы при получении очередного бита на значение, не большее снижения при получении предыдущего бита. Тем самым при получении уже первых битов ошибка снизится максимально, и пользователь сможет восстановить максимально хорошее приближение исходного изображения.

Алгоритм оперирует с преобразованными вейвлет-коэффициентами $c_{i,j}$, а не с исходными значениями яркости $p_{i,j}$. Евклидова норма инвариантна относительно ортогональных преобразований и сохраняет ошибку различия, поэтому ошибку различия можно применять прямо к преобразованным коэффициентам, если вейвлет-преобразование ортогонально. Вейвлет-преобразование, которое мы рассматриваем, почти ортогонально, поэтому к нему также применимы эти рассуждения. Тем самым мы получаем функцию ошибки

$$\operatorname{err}(C - \tilde{C}) = \sum_{i=0,\ j=0}^{i=N-1,\ j=N-1} (c_{i,j} - \tilde{c}_{i,j})^2,$$

которую необходимо уменьшать в процессе передачи изображения.

Из предыдущей формулы непосредственно следует, что передача наибольшего коэффициента с индексами (i, j) даст наибольший вклад в уменьшение ошибки, ошибка различия уменьшится на $|c_{i,j}|^2$. Но вместо оговорённого одного бита в данном случае передаются все биты в представлении коэффициента $c_{i,j}$. Чтобы избежать этого, представим каждый из преобразованных коэффициентов $c_{i,j}$ побитно. Тогда вместо передачи всего коэффициента сразу передадим только старший бит, а также старшие биты остальных (каких именно, будет указано ниже) коэффициентов, а уже потом оставшиеся биты коэффициентов.

Таким образом, прогрессивная передача изображений основана на следующих двух принципах:

- коэффициенты вейвлет-преобразования $c_{i,j}$ должны быть упорядочены по убыванию;
- при передаче используется битовое представление коэффициентов, т. е. вначале передаются старшие биты, а при следующих проходах — младшие.

2.2.2. Передача коэффициентов вейвлет-преобразования

Из двух принципов передачи изображения, указанных в предыдущем разделе, следует, что для передачи коэффициентов преобразованного изображения эти коэффициенты надо вначале упорядочить по убыванию, а точнее по количеству бит, необходимых для представления коэффициента. Имеется функция

$$\mu\colon [0 \ldots N^2 - 1] \to [0 \ldots N - 1] \times [0 \ldots N - 1],$$

задающая порядок преобразованных коэффициентов, т. е. выполнено неравенство

$$\lfloor \log(c_{\mu(i)}) \rfloor \geqslant \lfloor \log(c_{\mu(i+1)}) \rfloor$$

при $i = 0 \ldots N^2 - 2$.

Допустим, что, помимо порядка, декодеру передаётся и количество коэффициентов ν_n с одинаковым старшим битом n (позже мы откажемся от этих предположений). Число ν_n можно задать явно:

$$\nu_n = \#\{c_{i,j} \mid 2^n \leqslant |c_{i,j}| < 2^{n+1}\}.$$

Заметим, что в этом случае передача старших битов не нужна, так как они восстанавливаются по упорядочению $\mu()$ и значениям ν_n. Вместо старших битов коэффициентов $c_{i,j}$ передаются знаки этих коэффициентов (например, вместо $-$ передаётся 0, а вместо $+ — 1$). При дальнейших проходах передаются остальные биты коэффициентов. Так как вейвлет-преобразование единообразно преобразует каждый из коэффициентов, то порядок передачи коэффициентов в каждом из проходов не важен. На практике обычно мы также выбираем построчный проход изображения, откуда получаем побитовую передачу коэффициентов преобразованного изображения.

Ниже приведён получившийся первоначальный алгоритм кодирования изображения. Этот алгоритм может быть остановлен в любой момент, он даёт при этом определённое приближение к исходному изображению. Тогда, останавливая алгоритм в определённый момент, можно получить нужное приближение к исходному изображению.

Алгоритм 1: первоначальный алгоритм кодирования изображения

вход : преобразованное изображение
выход: поток закодированного изображения
вывести $n = \left\lfloor \log_2 \left(\max\limits_{(i,j)} \{|c_{i,j}|\} \right) \right\rfloor$
начиная с $i = n$ **до** 0 **выполнить**
  вывести ν_i
  `// сортировочный проход`
  начиная с $k = 0$ **до** $N^2 - 1$ **выполнить**
    если $2^n \leqslant |c_{\mu(k)}| < 2^{n+1}$ **то**
      вывести координаты $\mu(k)$
      вывести знак вейвлет-коэффициента $c_{\mu(k)}$
  `// уточняющий проход`
  начиная с $k = 0$ **до** $N^2 - 1$ **выполнить**
    если $2^{n+1} \leqslant \left|c_{\mu(k)}\right|$ **то**
      вывести n-й бит вейвлет-коэффициента $\left|c_{\mu(k)}\right|$

2.2.3. Алгоритм сортировки

Одно из преимуществ описываемого алгоритма заключается в том, что информация о порядке явно не передаётся. Она восстанавливается на основании того, что работа любого алгоритма определяется результатами сравнений в точках разветвления программы. Если алгоритмы сортировки коэффициентов у ко-

дера и декодера будут одинаковые, то алгоритм декодера может повторить работу алгоритма сортировки кодера, если ему будут переданы результаты сравнения в точках разветвления.

Оказывается, что не надо сортировать все коэффициенты. Достаточно просто отделять те вейвлет-коэффициенты $c_{i,j}$, которые имеют одинаковый старший бит n, где n меняется от максимального до 0. В связи с этим вводится следующее определение: для данного натурального числа n коэффициент $c_{i,j}$ считается *значимым*, если $|c_{i,j}| \geqslant 2^n$, иначе он считается *незначимым*.

Сортировочный алгоритм разбивает множество вейвлет-коэффициентов $\{c_{i,j}\}$ на подмножества τ_m и выполняет проверку значимости каждого из них:

$$\max_{(i,j)\in\tau_m} (|c_{i,j}|) \geqslant 2^n.$$

Если проверка даёт ложь, то это значит, что все элементы множества τ_m незначимы. Если же проверка даёт истину, то хотя бы какой-то элемент множества τ_m значим. При этом значимое множество τ_m разбивается на подмножества $\tau_{m,p}$ по заранее оговорённому правилу, и эта проверка выполняется для каждого из этих новых подмножеств. Такое разбиение выполняется до тех пор, пока не будет проверена значимость всех одноэлементных значимых множеств и, следовательно, не будут известны все значимые вейвлет-коэффициенты.

Чтобы уменьшить количество сравнений при сортировке и, значит, количество передаваемой информации, используется предполагаемое упорядочение в иерархическом дереве, заданное частотной пирамидой. Создаваемый алгоритм сортировки должен использовать такое правило при разбиении на подмножества, при котором незначимые подмножества содержали бы большое количество точек, а значимые подмножества были бы одноэлементными. Способ, указанный ниже, использует специфику разложения элементов изображения в иерархическое дерево.

Чтобы внести определённость между значимостью множества τ и передаваемой информацией, введём функцию значимости множества:

$$S_n(\tau) = \begin{cases} 1, & \max\limits_{(i,j)\in\tau} |c_{i,j}| \geqslant 2^n, \\ 0 & \text{иначе.} \end{cases}$$

Для простоты в случае одноэлементных множеств $\{(i,j)\}$ вместо $S_n(\{(i,j)\})$ пишем $S_n(i,j)$.

2.2.4. Построение пространственно ориентированного дерева

Обычно основная энергия изображения находится в низких частотах. Было также обнаружено сходство между различными частотными поддиапазонами, В частности, области с маленькими вейвлет-коэффициентами в низких частотах схожи с соответствующими областями в высоких частотах. Абсолютная величина вейвлет-коэффициентов упорядочивается лучше, если двигаться

вдоль пирамиды в определённом пространственном направлении. Для наглядности приводим преобразованное изображение на рис. 5.

Рис. 5. Пример трёхкратного вейвлет преобразования. Самый верхний левый квадрант является уменьшенной копией исходного изображения. Областям постоянства соответствуют тёмные области, а резким переходам — светлые границы. Разломы в кирпичной кладке дома проявили себя как набор горизонтальных линий в нижних квадрантах и набор кусочков вертикальных линий в правых квадрантах

Древовидная структура, называемая *пространственно ориентируемым деревом*, естественным образом определяет пространственные отношения на иерархических деревьях. На рис. 6 показано, как пространственно ориентированное дерево задаётся на вейвлет-пирамиде. Каждый узел дерева — это вейвлет-коэффициент, который определяется индексами (i, j). Его непосредственные потомки (дети) соответствуют вейвлет-коэффициентам той же ориентации, но в следующем уровне пирамиды. Дерево определено таким образом, что каждый узел имеет либо четыре ребёнка, либо ни одного. На рис. 6 стрелками показана ориентация от родителя к четырём детям. Вейвлет-коэффициенты на самом верхнем уровне пирамиды устроены несколько иначе: в каждой группе один из коэффициентов не имеет потомков. Этот факт тоже отражён на рисунке: вейвлет-коэффициент, не имеющий потомков, обозначен звёздочкой.

Определим следующие множества координат:

- $O(i, j)$ — множество координат непосредственных потомков (детей) узла (i, j);
- $D(i, j)$ — множество координат всех потомков узла (i, j);
- $L(i, j)$ — множество координат всех потомков детей узла (i, j), т. е. $L(i, j) = D(i, j) - O(i, j)$;

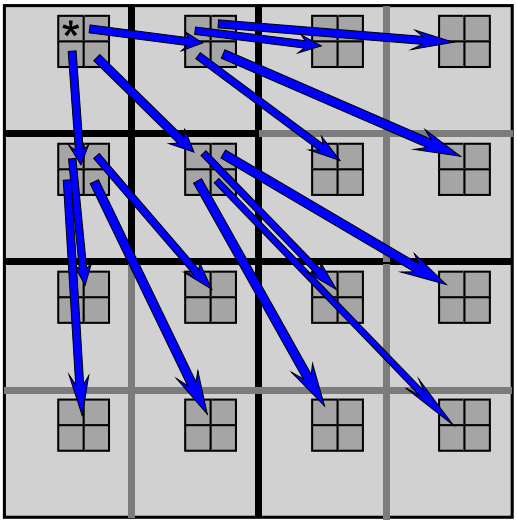

Рис. 6. Пример пространственно ориентируемого дерева в случае трёхкратного вейвлет-преобразования

- H — множество координат всех числовых характеристик $c_{i,j}$ из самого низкочастотного диапазона B_{N+1}.

Например, для всех вейвлет-коэффициентов, кроме принадлежащих самому высокому B_1 и самому низкому B_{N+1} частотным диапазонам, имеем

$$O(i,j) = \{(2i, 2j),\ (2i, 2j+1),\ (2i+1, 2j),\ (2i+1, 2j+1)\}.$$

Указанные множества используются для задания правил разбиения множества вейвлет-коэффициентов при проверке на значимость. Эти правила таковы:

- первоначальное разбиение состоит из множеств $\{(i,j)\}$ и $D(i,j)$, таких что $(i,j) \in H$;
- если множество $D(i,j)$ значимо, то оно разбивается на множество $L(i,j)$ и на четыре одноэлементных множества $(k,l) \in O(i,j)$;
- если множество $L(i,j)$ значимо, то оно разбивается на четыре множества $D(k,l)$, где $(k,l) \in O(i,j)$.

2.2.5. Алгоритм кодирования

Так как порядок, в котором подмножества проверяются на значимость, важен, то в алгоритме используются три типа упорядоченных списков:

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

В каждом из списков элементами являются индексы (i, j). В списках LIP и LSP индексы (i, j) соответствуют вейвлет-коэффициентам $c_{i,j}$ преобразованного изображения, а в LSP — множеству $D(i, j)$ либо множеству $L(i, j)$ вейвлет-коэффициентов преобразованного изображения. Чтобы отличить два последних случая, введём тип множества. Элемент (i, j) типа $\mathcal{A}$ соответствует множеству $D(i, j)$, а элемент (i, j) типа $\mathcal{B}$ — множеству $L(i, j)$.

Замечание. В алгоритме SS-SPIHT вместо глобальных списков LIS, LIP, LSP используются локальные списки $\mathrm{LIS}_{r,k}$, $\mathrm{LIP}_{r,k}$, $\mathrm{LSP}_{r,k}$.

При сортировочном проходе те вейвлет-коэффициенты в списке LIP, которые были незначимы в предыдущем проходе, проверяются на значимость и, если они оказываются значимыми, передвигаются в список LSP. Аналогичным образом множества в списке LIS проверяются на значимость и, если обнаруживается значимое множество, то оно удаляется из списка и дробится по указанным ранее правилам. Вновь полученные множества с более чем одним элементом добавляются в список LIS, тогда как одноэлементные множества добавляются в конец списка LIP или LSP в зависимости от значимости вейвлет-коэффициента. В списке LSP содержатся вейвлет-коэффициенты, у которых обновляются остальные биты.

Алгоритм 2: алгоритм кодирования изображения SPIHT

вход : преобразованное изображение
выход: поток кодированного изображения
инициализация алгоритма кодирования изображения
повторять
    сортировка списка LIP
    сортировка списка LIS
    уточнение пикселей из списка LSP
    $n = n - 1$
пока $n \geqslant 0$

Алгоритм 3: инициализация алгоритма кодирования изображения

вход : преобразованное изображение
выход: заголовок потока
вывести $n = \left\lfloor \log_2 \left(\max\limits_{(i,j)} \{|c_{i,j}|\} \right) \right\rfloor$
инициализировать все списки: $\mathrm{LSP} = \varnothing$, $\mathrm{LIP} = \varnothing$, $\mathrm{LIS} = \varnothing$
для всех *элементов* (i, j) *множества* H **выполнить**
    добавить элемент (i, j) в список LIP
    если *у элемента* (i, j) *имеются наследники* **то**
        добавить элемент (i, j) в список LIS как элемент типа $\mathcal{A}$

Алгоритм 4: сортировка списка LIP

вход : список LIP
выход: часть потока кодированного изображения
для всех *элементов* (i,j) *множества* LIP **выполнить**
    вывести $S_n(i,j)$
    если $S_n(i,j)$ *равняется* 1 **то**
        переместить элемент (i,j) в список LSP
        вывести знак элемента $c_{i,j}$

Алгоритм 5: сортировка списка LIS

вход : список LIS
выход: поток кодированного изображения
для всех *элементов* (i,j) *множества* LIS **выполнить**
    если *элемент* (i,j) *типа* $\mathcal{A}$ **то**
        вывести $S_n(D(i,j))$
        если $S_n(D(i,j))$ *равняется* 1 **то**
            для всех *элементов* (k,l) *множества* $O(i,j)$ **выполнить**
                вывести $S_n(k,l)$
                если $S_n(k,l)$ *равняется* 1 **то**
                    переместить элемент (k,l) в список LSP
                    вывести знак элемента $c_{k,l}$
                иначе
                    переместить элемент (k,l) в список LIP
                если $L(i,j) \neq \varnothing$ **то**
                    переместить элемент (i,j) в список LIS как тип $\mathcal{B}$
                иначе
                    удалить элемент (i,j) из списка LIS
    иначе если *элемент* (i,j) *типа* $\mathcal{B}$ **то**
        вывести $S_n(L(i,j))$
        если $S_n(L(i,j))$ *равняется* 1 **то**
            добавить все $(k,l) \in O(i,j)$ в список LIS как элементы типа $\mathcal{A}$
            удалить элемент (i,j) из списка LIS

Алгоритм 6: уточнение пикселей из списка LSP

вход : список LSP
выход: поток кодированного изображения
для всех *элементов* (i,j) *множества* LSP*, добавленных на предыдущих проходах* **выполнить**
    вывести n-й бит числа $|c_{i,j}|$

Заметим, что элементы, добавляемые в конец списка LIS, обрабатываются до того, как этот проход заканчивается. Этот алгоритм позволяет точно оценивать

ошибку, так как информация передаётся в каждом бите. Кодер может быть остановлен в любой нужный момент.

Все точки разветвления алгоритма зависят от значимости, которая может быть вычислена только в случае, когда известны все вейвлет-коэффициенты $c_{i,j}$. Результат проверки в точках разветвления передаётся декодеру. Для получения алгоритма декодирования, который повторяет ход кодера, надо заменить все посылки на приёмы. Заметим, что всегда, когда декодер принимает данные, списки LIS, LIS и LSP совпадают с такими списками у кодера, поэтому мы действительно восстанавливаем исходное изображение. Отсюда также следует, что сложность кодера и декодера одинаковы.

2.3. Spatially Scalable SPIHT (SS-SPIHT)

Пусть I — исходное изображение. К нему применяется N-кратное вейвлет-преобразование. Обозначим преобразованное изображение через C, а его коэффициенты — через $c_{i,j}$. Эти коэффициенты, как было показано в разделе 1.10, можно группировать по вкладу в частотную характеристику изображения. А именно, рассмотрим следующее стандартное разбиение:

$$C = \bigsqcup_{k=1}^{N+1} B_k,$$

где

$$B_k = \begin{cases} \{\mathrm{LL}_{k-1}\}, & k = N+1, \\ \{\mathrm{LH}_{k-1},\, \mathrm{HL}_{k-1},\, \mathrm{HH}_{k-1}\}, & 1 \leqslant k \leqslant N. \end{cases}$$

Рассматривая не все B_k, а только такие, что $k \geqslant r$, получим коэффициенты преобразованного масштабированного изображения уровня r. Обозначим его через R_r:

$$R_r = \{B_{N+1},\, B_N, \ldots, B_{r+1},\, B_r\}.$$

Пусть I_r — соответствующее ему изображение. Тогда $I = I_1$.

Коэффициенты R_1 передаются по битовым плоскостям. Введём число $n_{\max}$, которое равно максимальному битовому слою, необходимому для корректного представления R_1.

Заполним каждое из изображений I_r блоками фиксированного размера, выровняв их по координате $(0, 0)$. (Размер блока должен быть степенью двойки, причём больше восьми. Это свойство используется в функциях добавления элемента в список нужного блока.) Пронумеруем блоки построчно, слева направо, начиная с верхней строки. Обозначим количество блоков в изображении I_r переменной $\mathrm{num\_blocks}_r$. Для каждого из этих блоков вводятся такие же списки, как и для всего изображения в алгоритме SPIHT, описанном выше:

- $\mathrm{LIP}_{r,k}$ — список незначащих пикселей (i, j), принадлежащих блоку номер k в изображении I_r;

- $\mathrm{LSP}_{r,k}$ — список значащих пикселей (i,j), принадлежащих блоку номер k в изображении I_r;
- $\mathrm{LIS}_{r,k}$ — список множеств незначащих пикселей (i,j), принадлежащих блоку номер k в изображении I_r.

Списки $\mathrm{LIP}_{r,k}$ и $\mathrm{LSP}_{r,k}$ содержат индексы вейвлет-коэффициентов блока, которому соответствуют эти списки. Список $\mathrm{LIS}_{r,k}$ устроен сложнее. Множества этого списка задаются координатой корневого узла (i,j) и типом $\mathcal{A}$ или $\mathcal{B}$. Следует иметь в виду, что корни (i,j) множеств данного списка не обязаны принадлежать тому блоку, которому соответствуют эти списки. Вместо этого списку данного блока принадлежат те корни, чьи потомки (т. е. $(k,l) \in O(i,j)$ или $(k,l) \in O\big(O(i,j)\big)$ соответственно для типов $\mathcal{A}$ и $\mathcal{B}$) принадлежат этому блоку. Этот факт используется в конструкции функции, добавляющей элемент (i,j) в список $\mathrm{LIS}_{r,k}$.

Разбиение на блоки изображения I_k индуцирует разбиение B_k на блоки. Пример такого разбиения показан на рис. 4. При $k \leqslant N$ каждому блоку в I_k соответствуют три блока вдвое меньшего размера в B_k.

Алгоритм работы организован следующим образом. Как и в исходном алгоритме SPIHT, производится проход по всем битовым плоскостям от $n_{\max}$ до 0. При проходе очередной битовой плоскости алгоритм обрабатывает три списка для каждого из блоков, в отличие от SPIHT, где эти списки глобальны. Блоки обходятся в естественном порядке, а именно сначала обходятся блоки, соответствующие изображению I_{N+1}, потом I_N и так далее до I_1. В слое каждого масштаба блоки обходятся в порядке возрастания их номеров.

Алгоритм 7: алгоритм кодирования изображения SS-SPIHT

вход : преобразованное изображение
выход: поток кодированного изображения
инициализация алгоритма кодирования изображения
повторять
  начиная с $r = N+1$ **до** *1* **выполнить**
    начиная с $k = 1$ **до** $\mathrm{num\_blocks}_r$ **выполнить**
      сортировка списка $\mathrm{LIP}_{r,k}$
      сортировка списка $\mathrm{LIS}_{r,k}$
      уточнение пикселей из списка $\mathrm{LSP}_{r,k}$
  $n = n - 1$
пока $n \geqslant 0$

Сами алгоритмы сортировки списков не изменились, уточнению подвергся способ добавления элементов в списки. В новом варианте элементы добавляются в нужный список нужного блока. Если разбиение преобразованного изображения на блоки не производится, то элементы по-прежнему добавляются в глобальные списки, и тогда это изменение алгоритма не добавляют ничего нового к алгоритму SPIHT.

Опишем функции добавления элементов в списки. Способ применения этих функций в приведённых ранее алгоритмах очевиден.

Функции, добавляющие элементы (i,j) в списки $\mathrm{LIP}_{r,k}$ и $\mathrm{LSP}_{r,k}$, устроены просто. Они добавляют элемент (i,j) в список того блока, который содержит индексы (i,j) этого элемента. Основное отличие заключается в функции добавления элемента (i,j) в список $\mathrm{LIS}_{r,k}$. Она добавляет элемент (i,j) в список $\mathrm{LIS}_{r,k}$ того блока, который содержит детей (или внуков) данного элемента (i,j). В случае разбиения типа $\mathcal{A}$, как уже отмечалось выше, множество будет разбито на множество $L(i,j)$ и на четыре одноэлементных множества непосредственных потомков $(k,l) \in O(i,j)$. Поэтому элемент (i,j) будет помещён в список $\mathrm{LIS}_{r,k}$ блока, содержащего индексы $(k,l) \in O(i,j)$. Такой блок единственен (здесь используются ограничения, накладываемые на размер блоков). В случае же разбиения типа $\mathcal{B}$ множество будет разбито на четыре множества $D(k,l)$, где $(k,l) \in O(i,j)$. Поэтому в данном случае элемент (i,j) будет добавлен в список $\mathrm{LIS}_{r,k}$ блока, содержащего индексы $(k,l) \in O\big(O(i,j)\big)$. Такой блок тоже единственен.

Алгоритм 8: добавление элемента (i,j) в список LIP

вход : элемент (i,j)
выход: элемент (i,j) добавлен в нужный список $\mathrm{LIP}_{r,k}$
поиск блока с индексами (r,k), содержащего вейвлет-индекс (i,j)
добавить индекс (i,j) к списку $\mathrm{LIP}_{r,k}$

Алгоритм 9: добавление элемента (i,j) в список LSP

вход : элемент (i,j)
выход: элемент (i,j) добавлен в нужный список $\mathrm{LSP}_{r,k}$
поиск блока с индексами (r,k), содержащего вейвлет-индекс (i,j)
добавить индекс (i,j) к списку $\mathrm{LSP}_{r,k}$

Алгоритм 10: добавление элемента (i,j) в список LIS

вход : элемент (i,j)
выход: элемент (i,j) добавлен в нужный список $\mathrm{LIS}_{r,k}$
если *элемент* (i,j) *типа* $\mathcal{A}$ **то**
    поиск блока с индексами (r,k), содержащего множество
    вейвлет-индексов $O(i,j)$
    добавить индекс (i,j) к списку $\mathrm{LIS}_{r,k}$
иначе если *элемент* (i,j) *типа* $\mathcal{B}$ **то**
    поиск блока с индексами (r,k), содержащего множество
    вейвлет-индексов $O\big(O(i,j)\big)$
    добавить индекс (i,j) к списку $\mathrm{LIS}_{r,k}$

2.4. Организация потока данных

Так как предлагаемый метод кодирования предназначен для кодирования изображений с последующим извлечением данных, удовлетворяющих требуемому запросу, необходимо наложить определённые ограничения на структуру получаемых данных. Для этого, например, можно хранить кодированные данные как последовательность пакетов, содержащих заголовок и данные. Обычно заголовок содержит длину пакета. При формировании ответа на запрос пользователя из потока извлекаются только те пакеты, которые необходимы пользователю. Тем самым передаются не все, а только часть пакетов.

Способ организации потока данных, рассматриваемый в этой статье, следующий. Сначала необходимо сгруппировать выводимые данные в текущей битовой плоскости, относящиеся к определённому блоку (r, k). Перед выполнением шагов сортировки в поток пишется заголовок, содержащий номер блока k и номер изображения r. Кроме этого, в заголовке следует оставить место под длину последующих данных. Данное поле заполняется после выполнения всех сортировок. Разумеется, в самом начале потока должен находиться заголовок, характеризующий всё изображение. Он содержит информацию о разрешении изображения, количестве и типе вейвлет-преобразования, размере блоков.

В дальнейшем обработчик потока может из кодированного потока выбирать необходимые для удовлетворения запроса пакеты. Необходимые пакеты вычисляются применением выкладок данных из первого раздела статьи. Заметим, что заголовки пакетов необходимы только для выделения из кодированного потока необходимых данных. Для этого используется длина пакета, которая хранится в заголовке пакета. Поэтому заголовки пакетов можно не передавать клиенту, так как клиент знает, какие именно пакеты ему будут переданы, а длина пакета ему не нужна. Подчеркнём, что обработчику (т. е. серверу, передающему данные клиенту) для удовлетворения запроса клиента не надо декодировать изображение, ему достаточно декодировать только заголовки пакетов.

3. Анализ работы алгоритма

Эффективность предлагаемого алгоритма SS-SPIHT показана в разделе 3.1. В разделе 3.2 сравнивается его степень сжатия с широко известным алгоритмом JPEG2000.

3.1. Анализ эффективности кодирования изображения

Как уже было отмечено, предлагаемому алгоритму не надо декодировать всё изображение для извлечения нужного фрагмента. На примере изображения, приведённого на рис. 7, сравним объём памяти, необходимый предлагаемому алгоритму, при различных размерах блока. Заметим, что при отсутствии размера блока алгоритм полностью совпадает с первоначальным алгоритмом SPIHT.

Рис. 7. Изображение pentagon взято из известной базы [11]. Оно имеет 1024 пикселя в ширину и в высоту, 8 бит на пиксель

В таблице указан объём передаваемой информации, необходимый для восстановления фрагмента изображения, приведённого на рис. 8. Объём указан в процентах от памяти, необходимой для хранения фрагмента без сжатия, т. е. 100 % соответствуют ширине фрагмента, умноженной на высоту фрагмента. Из таблицы видно неоспоримое преимущество предлагаемого алгоритма.

Размер блока	Степень сжатия
8×8	100 %
16×16	109 %
32×32	140 %
64×64	209 %
нет	2792 %

Для примера приведём приближение к фрагменту при фиксированном объёме передаваемой информации, но при различных размерах блока. Результат сравнения показан на рис. 9.

3.2. Анализ степени сжатия

Помимо того, что указанный метод позволяет извлекать фрагменты изображения необходимого масштаба, он ещё и сжимает данные. В приведённой ниже таблице сравнивается степени сжатия предлагаемого метода SS-SPIHT и

Рис. 8. Пример фрагмента, вырезанного из изображения, приведённого на рис. 7, уменьшенного в два раза. Координаты левого верхнего угла фрагмента равны 140, 170, размер фрагмента — 162 на 162 пикселя

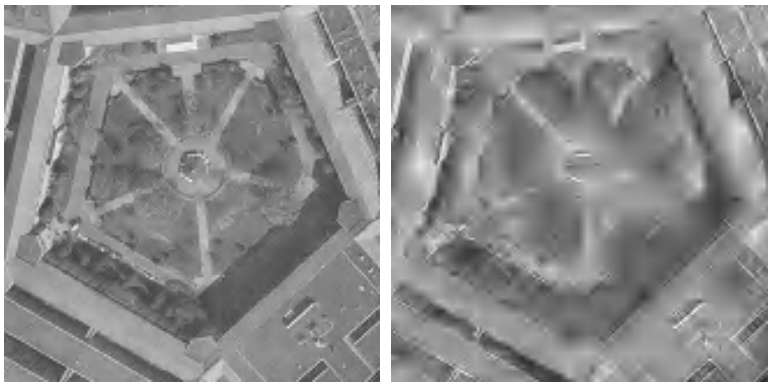

Рис. 9. Сравнение фрагментов, полученных новым и первоначальным алгоритмом. Объём передаваемой информации равнялся 1,875 % от объёма целого изображения. На рисунке видно неоспоримое преимущество предлагаемого алгоритма. Значения PSNR соответственно равны 32,03 dB и 22,11 dB

метода JPEG2000 на широко известных изображениях lena и barb. Из таблицы видно существенное преимущество предлагаемого алгоритма при размерах блока меньше 64 на 64, причём чем меньше размер блока, тем существеннее преимущество.

Размер блока	lena		barb	
	SS-SPIHT	JPEG-2000	SS-SPIHT	JPEG-2000
8×8	78,3 %	120,3 %	83,8 %	126,2 %
16×16	62,4 %	73,8 %	67,5 %	78,3 %
32×32	58,5 %	60,0 %	63,5 %	64,3 %
64×64	57,5 %	55,7 %	62,5 %	60,0 %

4. Заключение

В данной работе предложен новый алгоритм кодирования изображений SS-SPIHT, который, помимо сжатия изображения, наделяет выходной поток данных важными свойствами. Выходной поток данных организован таким образом, что для восстановления фрагмента масштабированного изображения можно использовать не весь поток данных, а только его часть, что существенно снижает объём передаваемых данных.

Преимущества предложенного алгоритма подтверждены на практике. В предыдущем разделе было показано, что уже при объёме $1,875$ % предлагаемый алгоритм SS-SPIHT формирует фрагмент изображения удовлетворительного качества, тогда как стандартный алгоритм формирует изображение неудовлетворительного качества. При этом достигается сжатие исходного изображения в 72 %, тогда как немодифицированный SPIHT даёт 70 %, т. е. на накладные расходы уходит всего 2 % — совсем немного.

Дальнейшая работа предполагает совершенствование алгоритма SS-SPIHT, в частности разработку способа более компактного хранения заголовков, более эффективное в смысле SNR разбиение на пакеты и, в конечном счёте, внедрение SPIHT в стандарт JPEG2000. Также предполагается исследовать вопрос об использовании контекстной информации для повышения степени сжатия изображений и степень влияния на этот процесс разбиения на блоки.

Литература

[1] Шокуров А. В., Михалёв А. В. Оптимальное использование вейвлет-компонент // Успехи мат. наук. — 2007. — Т. 62, № 4. — С. 171—172.

[2] Danyali H., Mertins A. Fully spatial and SNR scalable, SPIHT-based image coding for transmission over heterogeneous networks // J. Telecommunications Information Technol. — 2003. — Vol. 2. — P. 92—98.

[3] ISO/IEC 10918-1 ITU Recommendation T. 81: Information Technology — Digital Compression and of Continuous-Tone Still Images — Requirements and Guidelines.

[4] ISO/IEC fdc-15444-1. JPEG2000 Image Coding System.

[5] Mallat S. A Wavelet Tour of Signal Processing. — Academic Press, 1999.

[6] Pratt W. K. Digital Image Processing. — Wiley, 2001.

[7] Said A., Pearlman W. A. A new fast and efficient image codec based on set partitioning in hierarchical trees // IEEE Trans. Circuits Systems Video Technol. — 1996. — Vol. 6. — P. 243—250.

[8] Taubman D. High performance scalable image compression with EBCOT // IEEE Trans. Image Processing. — 2000. — Vol. 9, no. 7. — P. 1151—1170.

[9] Taubman D., Ordentlich E., Weinberger M., Seroussi G. Embedded block coding in JPEG2000 // Signal Processing — Image Communication. — 2002. — Vol. 17, no. 1. — P. 49—72.

[10] Wheeler F. W., Pearlman W. A. Combined spatial and subband block coding of images // Proc. Int. Conf. on Image Processing, ICIP'00. Vol. 3. — P. 861—864.

[11] The USC-SIPI Image Database: Aerials: 3.2.25 (Pentagon). — http://sipi.usc.edu/database/database.cgi?volume=aerials&image=37#top.