Статья-победитель конкурса на лучшую статью по сжатию
www.compression.ru

УДК 519.677

Прогрессивная передача изображений через Интернет

В этой статье приводится практическая реализация нескольких вариантов вейвлетного преобразования Баттерворта, а так же описывается эффективный метод прогрессивной передачи изображений по сети Интернет ■

Progressive image transmission over the Internet
This article gives the practical implementation of the Butterworth wavelet transform and describes an efficient method for progressive image transmission over the Internet ■

А.В. Симаков
(студент 4-го курса
Сыктывкарского
государственного университета)
Научный руководитель
д. ф.-м.н., проф. А.Б. Певный

Введение

В этой статье приведена практическая реализация нескольких вариантов вейвлетного преобразования Баттерворта [1, 2], а так же описывается эффективный способ прогрессивной передачи изображений по сети. Безусловно, наибольший интерес представляет передача изображений по низкоскоростным каналам связи, в частности по сети Интернет.

Процесс «вейвлетного» сжатия можно условно разбить на два шага: дискретное вейвлетное преобразование и кодирование. Чтобы все преобразования были в равных условиях, необходимо использовать для них одинаковый алгоритм кодирования. В этой работе был использован алгоритм кодирования SPIHT [6, 7].

Результаты численных экспериментов приводятся в заключительной части статьи. Они показывают, какой из вариантов вейвлетного преобразования Баттерворта работает наиболее эффективно. Для сравнения так же приводятся результаты, полученные с помощью известного вейвлетного преобразования Добеши 9/7, используемого в новом стандарте сжатия изображений JPEG2000.

1. Преобразование Баттерворта

1.1 Фильтр Баттерворта

Фильтр Баттерворта F_r сигналу $x = \{x(k)\}_{k=-\infty}^{\infty}$ ставит в соответствие сигнал $y = \{y(l)\}_{l=-\infty}^{\infty}$. Действие F_r описывается с помощью z-преобразований

$$X(z) = \sum_{k=-\infty}^{\infty} x(k) z^{-k} \quad \text{и} \quad Y(z) = \sum_{l=-\infty}^{\infty} y(l) z^{-l} .$$

Считаем, что z пробегает единичную окружность $|z| = 1$, это обычно обеспечивает сходимость рядов. Связь между X и Y устанавливается равенством

$$Y(z) = F_r(z)X(z).$$

Общий вид функций F_r приведен в [1], например, при четном значении *r* будем иметь

$$F_r(z) = \sum_{k=1}^{r} C_{2r}^{2k-1} z^k \left(\sum_{k=0}^{r} C_{2r}^{2k} z^k \right)^{-1},$$

а при нечетном

$$F_r(z) = z\sum_{k=0}^{r} C_{2r}^{2k} z^k \left(\sum_{k=1}^{r} C_{2r}^{2k-1} z^k \right)^{-1}.$$

Рассмотрим случай $r = 2$. Получаем следующую формулу

$$F_2(z) = \frac{4z + 4z^2}{1 + 6z + z^2}.$$

Для реализации при помощи элементарных рекурсивных фильтров разложим $F_2(z)$ в сумму простейших дробей

$$F_2(z) = \frac{4\alpha}{1+\alpha}\left(\frac{1}{1+\alpha z^{-1}} + \frac{z}{1+\alpha z} \right),\ \alpha = 3 - 2\sqrt{2} \approx 0.172.$$

В таком случае

$$Y(z) = F_2(z)X(z) = \frac{4\alpha}{1+\alpha}\left(\frac{X(z)}{1+\alpha z^{-1}} + \frac{X(z)z}{1+\alpha z} \right).$$

Введем, сигналы y_1 и y_2 с *z*-преобразованиями

$$Y_1(z) = \frac{X(z)}{(1+\alpha z^{-1})},\ Y_2(z) = \frac{X(z)z}{(1+\alpha z)}.$$

Для сигналов y_1 и y_2 справедливы равенства

$$y_1(l) + \alpha y_1(l-1) = x(l),\ y_2(l) + \alpha y_2(l+1) = x(l+1).$$

Отсюда получаем

$$y_1(l) = x(l) - \alpha y_1(l-1)\ \ \forall l \in Z, \tag{1.1}$$

$$y_2(l) = x(l+1) - \alpha y_2(l+1)\ \ \forall l \in Z. \tag{1.2}$$

Окончательно получаем отфильтрованный сигнал $y = \{y(l)\}_{l=-\infty}^{\infty}$ по формуле

$$y(l) = \frac{4\alpha}{1+\alpha}\left(y_1(l) + y_2(l)\right). \tag{1.3}$$

Рассмотрим теперь случай $r = 3$:

$$F_3(z) = \frac{1}{6}\left(-\frac{8}{1+\gamma z} - \frac{8}{9}\frac{z^{-1}}{1+\gamma z^{-1}} + z + \frac{35}{3} \right),\ \gamma = \frac{1}{3}.$$

Следовательно

$$Y(z) = F_3(z)X(z) = \frac{1}{6}\left(-\frac{8X(z)}{1+\gamma z} - \frac{\frac{8}{9}z^{-1}X(z)}{1+\gamma z^{-1}} + (z + \frac{35}{3})X(z) \right).$$

Введем, как и в предыдущем случае, сигналы y_1 и y_2 с *z*-преобразованиями

$$Y_1(z) = \frac{X(z)}{1+\gamma z},\ Y_2(z) = \frac{z^{-1}X(z)}{1+\gamma z^{-1}}.$$

Для сигналов y_1 и y_2 справедливы равенства

$$y_1(l) + \gamma y_1(l+1) = x(l),\ y_2(l) + \gamma y_2(l-1) = x(l-1).$$

Откуда заключаем

$$y_1(l) = x(l) - \gamma y_1(l+1)\ \ \forall l \in Z, \tag{1.4}$$

$$y_2(l) = x(l-1) - \gamma y_2(l-1) \quad \forall l \in Z\,. \tag{1.5}$$

Окончательно получаем отфильтрованный сигнал $y = \{y(l)\}_{l=-\infty}^{\infty}$ по формуле

$$y(l) = \frac{1}{6}\left(-8y_1(l) - \frac{8}{9}y_2(l) + x(l+1) + \frac{35}{3}x(l)\right). \tag{1.6}$$

Стоит отметить, что оператор F_r нужно адаптировать для применения к сигналам конечной длины. Эта адаптация не носит строго математического характера, а является набором практических рецептов.

1.2 Фильтр Баттерворта для конечных сигналов

Во всех предыдущих рассуждениях предполагали бесконечность сигнала. На практике же приходится иметь дело лишь с сигналами конечной длины. Ясно, что конечные сигналы представляют собой обыкновенные массивы чисел. В дальнейшем, там, где это уместно, вместо термина «сигнал» будем употреблять термин «массив».

Итак, далее будем рассматривать только конечные сигналы. Пусть $x = \{x(k)\}_{k=0}^{N-1}$ исходный сигнал длины N, $y = \{y(l)\}_{l=0}^{N-1}$ его отфильтрованная версия, а $y_1 = \{y_1(l)\}_{l=0}^{N-1}$ и $y_2 = \{y_2(l)\}_{l=0}^{N-1}$ некоторые промежуточные сигналы. При работе с конечным сигналом возникает одна проблема: необходимо корректно оценить его крайние элементы. Рассмотрим, к примеру, формулу (1.1). Видно, что если $l = 0$, то формула не имеет смысла, так как элемент $y_1(-1)$ не определен. Рассмотрим теперь формулу (1.2). Если $l = N-1$, то эта формула также теряет смысл, так как элементы $x(N)$ и $y_2(N)$ не определены.

Эта проблема решается путем продолжения обрабатываемого сигнала влево и вправо зеркальным образом. Будем считать, что

$$\begin{aligned}
&x(-1) = x(0),\\
&x(-2) = x(1),\\
&x(-3) = x(2),\\
&\ldots\\
&x(-d) = x(d-1).
\end{aligned}$$

Вправо сигнал продолжается аналогичным образом

$$\begin{aligned}
&x(N) = x(N-1),\\
&x(N+1) = x(N-2),\\
&x(N+2) = x(N-3),\\
&x(N+3) = x(N-4),\\
&\ldots\\
&x(N+d) = x(N-d-1).
\end{aligned}$$

Вернемся теперь к формуле (1.1) и попробуем оценить значение $y_1(0)$:

$$y_1(0) = x(0) - \alpha y_1(-1),$$
$$y_1(-1) = x(-1) - \alpha y_1(-2),$$
$$y_1(-2) = x(-2) - \alpha y_1(-3),$$
$$y_1(-3) = x(-3) - \alpha y_1(-4),$$
$$...$$
$$y_1(-d) = x(-d) - \alpha y_1(-d-1).$$

Выберем некоторое $d \le N$, положим $y_1(-d-1) = 0$ и подставим все полученные формулы в выражение для $y_1(0)$:

$$y_1(0) = x(0) - \alpha(x(-1) - \alpha(x(-2) - \alpha(x(-3) - \alpha(...) - \alpha(x(-d) - \alpha \cdot 0)...)))).$$

Учитывая выражение для $x(-k)$, получаем следующее выражение для $y_1(0)$

$$y_1(0) = x(0) - \alpha(x(0) - \alpha(x(1) - \alpha(x(2) - \alpha(...) - \alpha(x(d-1) - \alpha \cdot 0)...)))) =$$
$$= x(0) - \alpha x(0) + \alpha^2 x(1) - \alpha^3 x(2) + ... + (-\alpha)^d x(d-1),$$

что равносильно

$$y_1(0) = x(0) + \sum_{i=1}^{d} (-\alpha)^i x(i-1). \qquad (1.7)$$

Величина d называется глубиной инициализации и определяется экспериментально. Видно, что с ростом d величина $(-\alpha)^{d+1}$ будет очень быстро убывать, например, если $d = 8$, то $(-\alpha)^9 \approx 0.0000001$. Исходя из этих соображений, в программной реализации была использована глубина инициализации $d = 8$.

Пользуясь аналогичными соображениями, получим оценку значения $y_2(N-1)$ в формуле (1.2):

$$y_2(N-1) = x(N-1) + \sum_{i=1}^{d} (-\alpha)^i x(N-i). \qquad (1.8)$$

Таким же способом получаем оценки для $y_1(N-1)$ и $y_2(0)$ в формулах (1.4) и (1.5) соответственно,

$$y_1(N-1) = x(N-1) + \sum_{i=1}^{d} (-\gamma)^i x(N-i), \qquad (1.9)$$

$$y_2(0) = x(0) + \sum_{i=1}^{d} (-\gamma)^i x(i-1). \qquad (1.10)$$

Итак, подведем итоги. Действие фильтра F_2 для конечного сигнала сводится к следующим пяти шагам:

1. Вычисляем значение $y_1(0)$ по формуле (1.7).
2. Для всех $l \in 1 : N-1$ вычисляем $y_1(l)$ по формуле (1.1).
3. Вычисляем значение $y_2(N-1)$ по формуле (1.8).
4. Для всех $l \in N-2 : 0$ вычисляем $y_2(l)$ по формуле (1.2).
5. Для всех $l \in 0 : N-1$ вычисляем $y(l)$ по формуле (1.3).

Аналогичная последовательность действий выполняется и для фильтра F_3.

1. Вычисляем значение $y_1(N-1)$ по формуле (1.9).
2. Для всех $l \in N-2 : 0$ вычисляем $y_1(l)$ по формуле (1.4).
3. Вычисляем значение $y_2(0)$ по формуле (1.10).
4. Для всех $l \in 1 : N-1$ вычисляем $y_2(l)$ по формуле (1.5).
5. Для всех $l \in 0 : N-1$ вычисляем $y(l)$ по формуле (1.6) положив $x(N) = x(N-1)$.

1.3 Вейвлетное преобразование Баттерворта

Опишем теперь вейвлетное преобразование Баттерворта для конечного дискретного сигнала $x = \{x(k)\}_{k=0}^{N-1}$, где N – четное число. Наиболее эффективным способом реализации вейвлетных преобразований вообще является, так называемый, лифтинговый алгоритм (Lifting algorithm). Принцип его работы достаточно прост и укладывается всего в четыре шага.

Начнем с алгоритма декомпозиции сигнала. Иногда его называют анализом или просто вейвлетным преобразованием. Затем дадим алгоритм реконструкции сигнала. Его также называют синтезом, восстановлением или обратным вейвлетным преобразованием.

Декомпозиция сигнала

Шаг 1: Расщепление

Расщепим массив $x = \{x(k)\}_{k=0}^{N-1}$ на четную (**e**ven) и нечетную (o**d**d) части следующим образом

$$e_1 = \{e_1(k) = x(2k)\}_{k=0}^{N/2-1},\ d_1 = \{d_1(k) = x(2k+1)\}_{k=0}^{N/2-1}.$$

Шаг 2: Предсказание

Четный массив e_1 будет использован для предсказания элементов нечетного массива d_1. Чем точнее будет предсказание, тем эффективнее будет работать преобразование. Введем теперь массив $\widetilde{d}_1 = \{\widetilde{d}_1(k)\}_{k=0}^{N/2-1}$ и определим его как разность между d_1 и предсказанием

$$\widetilde{d}_1 = d_1 - F_r e_1.$$

Если предиктор работает хорошо, то элементы $\widetilde{d}_1$ должны быть близки к нулю.

Шаг 3: Обновление

Введем теперь массив $\widetilde{e}_1 = \{\widetilde{e}_1(k)\}_{k=0}^{N/2-1}$ и определим его следующим образом

$$\widetilde{e}_1 = e_1 + \Phi_r \widetilde{d}_1,$$

где действие оператора Φ_r объясним ниже. По существу, $\widetilde{e}_1$ - это сглаженная версия сигнала x.

Шаг 4: Нормализация

$$\widetilde{e}_1 = \widetilde{e}_1 \cdot \sqrt{2},\ \widetilde{d}_1 = \widetilde{d}_1 / \sqrt{2}.$$

Набор из двух массивов $\{\widetilde{e}_1, \widetilde{d}_1\}$ называется одноуровневым вейвлетным преобразованием Баттерворта сигнала x. Реконструкция сигнала x из $\{\widetilde{e}_1, \widetilde{d}_1\}$ осуществляется в обратном порядке.

Реконструкция сигнала

Шаг 1:

$$\widetilde{e}_1 = \widetilde{e}_1 / \sqrt{2},\ \widetilde{d}_1 = \widetilde{d}_1 \cdot \sqrt{2}.$$

Шаг 2:

$$e_1 = \widetilde{e}_1 - \Phi_r \widetilde{d}_1.$$

Шаг 3:

$$d_1 = \widetilde{d}_1 + F_r e_1.$$

Шаг 4:

$$x(2k) = e_1(k),\ x(2k+1) = d_1(k),\ k \in 0 : N/2 - 1.$$

Таким образом, был полностью реконструирован (восстановлен) исходный сигнал x. Как можно было заметить, в лифтинговом алгоритме появился новый оператор Φ_r. Он отличается от $\frac{1}{2}F_r$ лишь отставанием на один отсчет, т.е.

$$(\Phi_r\widetilde{d}_1)(k) = \frac{1}{2}(F_r\widetilde{d}_1)(k-1),\ k \in 1:N-1,$$
$$(\Phi_r\widetilde{d}_1)(0) = \frac{1}{2}(F_r\widetilde{d}_1)(0).$$

По сути, программная реализация $\Phi_r\widetilde{d}_1$ сводится к вычислению $\frac{1}{2}F_r\widetilde{d}_1$ и сдвигу получившегося сигнала (массива) на один отсчет вправо, а $(\Phi_r\widetilde{d}_1)(0)$ полагается равным $\frac{1}{2}(F_r\widetilde{d}_1)(0)$.

Как уже было упомянуто выше, набор $\{\widetilde{e}_1, \widetilde{d}_1\}$ называется одноуровневым вейвлетным преобразованием. Применив лифтинговый алгоритм к сигналу $\widetilde{e}_1$, можно получить набор $\{\widetilde{e}_2, \widetilde{d}_2, \widetilde{d}_1\}$, где $\widetilde{e}_2 = \{\widetilde{e}_2(k)\}_{k=0}^{N/4-1}$ и $\widetilde{d}_2 = \{\widetilde{d}_2(k)\}_{k=0}^{N/4-1}$ суть вейвлетное разложение сигнала $\widetilde{e}_1$. Набор $\{\widetilde{e}_2, \widetilde{d}_2, \widetilde{d}_1\}$ называется двухуровневым вейвлетным преобразованием Баттерворта сигнала x. Ясно, что, продолжая в таком же духе, можно получить практически любое количество уровней разложения сигнала. Единственным ограничением является длина исходного сигнала $x = \{x(k)\}_{k=0}^{N-1}$. Если требуется выполнить t-уровневое вейвлетное преобразование, необходимо чтобы N делилось нацело на 2^t.

1.4 Гибридная схема

Как можно было заметить из п. 1.3, на этапе предсказания и на этапе обновления использовались фильтры одного и того же порядка r. На самом деле вовсе не обязательно поступать именно так. Более того, как показывает практика [2], использование разных фильтров на этапе предсказания и этапе обновления часто дает лучшие результаты. Таким образом, комбинируя фильтры, получаем четыре различных вейвлетных преобразования.

Обозначение	Предсказание	Обновление
$F_2 + \Phi_2$	F_2	Φ_2
$F_2 + \Phi_3$	F_2	Φ_3
$F_3 + \Phi_2$	F_3	Φ_2
$F_3 + \Phi_3$	F_3	Φ_3

1.5 Вейвлетное преобразование Баттерворта для изображений

Изображение является конечным двумерным дискретным сигналом. Такой сигнал представляет собой просто двумерный массив чисел (т.е. матрицу). Итак, будем рассматривать изображение $X = \{X(k,l)\}$ размера $M \times N$ точек, где $k \in 0:M-1$, $l \in 0:N-1$, N – ширина изображения (количество столбцов), M – высота изображения (количество строк). На числа N и M, как и в предыдущем случае, наложим условие четности. В этом разделе будет показано, каким образом можно применить вейвлетное преобразование Баттерворта к двумерному сигналу X.

Двумерное вейвлетное преобразование сводится к применению одномерного, сначала ко всем строкам, а затем ко всем столбцам изображения X. Полученная в результате матрица $Y = \{Y(k,l)\}$, где $k \in 0:M-1$, а $l \in 0:N-1$ называется двумерным вейвлетным преобразованием сигнала X. Реконструкция (восстановление) изображения производится в обратном порядке. Сначала одномерный алгоритм реконструкции применяется ко всем столбцам, а затем ко всем строкам матрицы Y.

Введем несколько обозначений. Пусть $X[i,*]$ - i-я строка матрицы X, $X[*,j]$ - j-й столбец матрицы X. Символом $\cup$ будем обозначать слияние двух массивов.

Алгоритм двумерной декомпозиции сигнала

- Для каждой строки $X[i,*]$ изображения X, где $i \in 0:M-1$.
  1. Считываем строчку $X[i,*]$ из изображения X.
  2. Применяем к $X[i,*]$ одномерное вейвлетное преобразование Баттерворта, описанное в п. 1.3. В результате этого преобразования получаем два массива: $\widetilde{e}_1$ и $\widetilde{d}_1$ - вейвлетное разложение одномерного сигнала $X[i,*]$.
  3. Вместо строчки $X[i,*]$ изображения X записываем массив $\widetilde{e}_1 \cup \widetilde{d}_1$.
- Для каждого столбца $X[*,j]$ изображения X, где $j \in 0:N-1$.
  4. Считываем столбец $X[*,j]$ из изображения X.
  5. Применяем к $X[*,j]$ одномерное вейвлетное преобразование Баттерворта. В результате получаем два массива $\widetilde{e}_1$ и $\widetilde{d}_1$.
  6. Вместо столбца $X[*,j]$ записываем массив $\widetilde{e}_1 \cup \widetilde{d}_1$.

Алгоритм двумерной реконструкции сигнала

Алгоритм двумерной реконструкции аналогичен алгоритму декомпозиции за одним исключением. Алгоритм реконструкции сначала применяется к столбцам, а затем уже к строкам.

Итак, теперь известно как организовано одноуровневое двумерное вейвлетное преобразование Баттерворта. Покажем как работает его многоуровневая версия. Для этого рассмотрим процесс преобразования строк и столбцов исходного изображения более подробно.

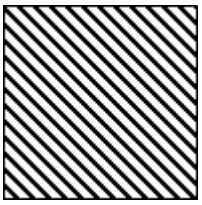

Рисунок 1

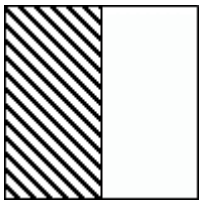

Рисунок 2

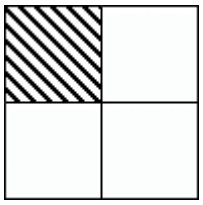

Рисунок 3

На рисунке 1 заштриховано оригинальное изображение X. Обратите внимание, что на первом шаге лифтинг-алгоритма (см. п. 1.3) происходит прореживание исходного сигнала (в массив e_1 записывается каждый второй элемент). Применим одномерный алгоритм декомпозиции ко всем строкам. Результат изображен на рисунке 2. Заштрихованная область рисунка 2 – это сплюснутое в 2 раза по горизонтали исходное изображение. Применим теперь одномерный алгоритм декомпозиции ко всем столбцам. Результат изображен на рисунке 3. Заштрихованная область на рисунке 3 – это сплюснутая в два раза по вертикали заштрихованная область рисунка 2. Выходит, что после применения одноуровневого двумерного вейвлетного преобразования к исходному изображению X получаются четыре области (sub-bands). При этом заштрихованная на

рисунке 3 область практически (нужно конечно учитывать нормализацию и действие оператора Φ_r) совпадает с исходным изображением, уменьшенным в 2 раза.

Заштрихованную область рисунка 3 можно рассматривать как изображение размера $\frac{M}{2}\times\frac{N}{2}$ точек. Следовательно, к нему так же можно применить одноуровневое двумерное вейвлетное преобразование, описанное в этом разделе. В результате можно получить двухуровневое двумерное вейвлетное преобразование исходного изображения X. Оно схематично изображено на рисунке 4.

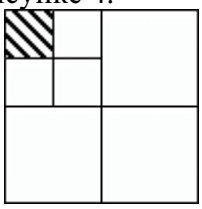

Рисунок 4

Ясно, что можно применять данный алгоритм рекурсивно столько раз, сколько нам нужно. Единственным ограничением является размер исходного изображения. Если требуется применить *t*-уровневое двумерное вейвлетное преобразование Баттерворта, необходимо чтобы *N* и *M* делились нацело на 2^t.

2. Преобразование Добеши 9/7

Вейвлетное преобразование Добеши 9/7 [3-5] является, пожалуй, самым известным из существующих на сегодняшний день вейвлетных преобразований. Более того, оно считается одним из наиболее эффективных. Лишним тому доказательством может являться то, что преобразование Добеши 9/7 было выбрано за основу в новом стандарте для сжатия изображений JPEG2000.

Так как преобразование Добеши 9/7 не является центральной темой этой статьи, приведем лишь итоговые формулы и алгоритмы. Итак, пусть дан исходный сигнал $x = \{x(k)\}_{k=0}^{N-1}$, где N – четное число. Положим

$$\alpha = \text{-}1.58615986717275,$$
$$\beta = \text{-}0.05297864003258,$$
$$\gamma = 0.88293362717904,$$
$$\delta = 0.44350482244527,$$
$$\varepsilon = 1.14960430535816.$$

Декомпозиция сигнала

1. $e_0(l) = x(2l),\ l \in 0 : N/2 - 1.$
2. $d_0(l) = x(2l+1),\ l \in 0 : N/2 - 1.$
3. $d_1(l) = d_0(l) + \alpha(e_0(l) + e_0(l+1)),\ l \in 0 : N/2 - 1.$
4. $e_1(l) = e_0(l) + \beta(d_1(l) + d_1(l-1)),\ l \in 0 : N/2 - 1.$
5. $d_2(l) = d_1(l) + \gamma(e_1(l) + e_1(l+1)),\ l \in 0 : N/2 - 1.$
6. $e_2(l) = e_1(l) + \delta(d_2(l) + d_2(l-1)),\ l \in 0 : N/2 - 1.$
7. $\tilde{e}(l) = \varepsilon \cdot e_2(l),\ l \in 0 : N/2 - 1.$
8. $\tilde{d}(l) = -d_2(l)/\varepsilon,\ l \in 0 : N/2 - 1.$

Набор $\{\widetilde{e},\widetilde{d}\}$ называется одноуровневым вейвлетным преобразованием Добеши 9/7 сигнала x. Реконструкция сигнала x из $\{\widetilde{e},\widetilde{d}\}$ осуществляется в обратном порядке.

Реконструкция сигнала

1. $d_2(l) = -\varepsilon \cdot \widetilde{d}(l),\ l \in 0 : N/2 - 1.$
2. $e_2(l) = \widetilde{e}(l)/\varepsilon,\ l \in 0 : N/2 - 1.$
3. $e_1(l) = e_2(l) - \delta(d_2(l) + d_2(l-1)),\ l \in 0 : N/2 - 1.$
4. $d_1(l) = d_2(l) - \gamma(e_1(l) + e_1(l+1)),\ l \in 0 : N/2 - 1.$
5. $e_0(l) = e_1(l) - \beta(d_1(l) + d_1(l-1)),\ l \in 0 : N/2 - 1.$
6. $d_0(l) = d_1(l) - \alpha(e_0(l) + e_0(l+1)),\ l \in 0 : N/2 - 1.$
7. $x(2l+1) = d_0(l),\ l \in 0 : N/2 - 1.$
8. $x(2l) = e_0(l),\ l \in 0 : N/2 - 1.$

Значения крайних элементов сигнала, в отличие от преобразования Баттерворта, оцениваются намного проще. А именно,

$$e_0(N/2) = e_0(N/2-1),$$
$$d_1(-1) = d_1(0),$$
$$e_1(N/2) = e_1(N/2-1),$$
$$d_2(-1) = d_2(0).$$

Все, что было сказано в п. 1.3 и 1.5 про многоуровневое и двумерное вейвлетное преобразование, справедливо для любого вейвлетного преобразования, в том числе и для преобразования Добеши 9/7.

Ко многим «плюсам» преобразования Добеши 9/7 можно отнести еще одно: в отличие от преобразования Баттерворта, условие четности сигнала не является обязательным. Преобразование Добеши, при необходимости, можно применять и к сигналам нечетной длины.

3. Алгоритм кодирования SPIHT

3.1 Введение

Алгоритм SPIHT (Set Partitioning in Hierarchical Trees) [6, 7] применяется на заключительной стадии для кодирования коэффициентов вейвлетного разложения изображения. Для большей эффективности SPIHT комбинируют с арифметическим кодированием [8]. К моменту кодирования коэффициенты должны быть округлены до ближайших целых. Как правило, коэффициенты вейвлетного разложения умещаются в 16 бит.

Как ни странно это звучит, но основная идея SPIHT заключается не в том, чтобы непосредственно сжимать изображение, а в том, чтобы переупорядочить биты вейвлетного разложения специальным образом. Уже давно установлено, что для человеческого восприятия низкочастотные компоненты изображения (т.е. плавные переходы яркости и цвета) несут гораздо больше информации, чем высокочастотные (резкие границы, углы, прямые линии). По сути, на принципе выделения низко- и высокочастотной информации, с последующим подавлением последней, и построены практически все технологии сжатия изображений с потерями. Не исключение и SPIHT.

Используя особенности структуры вейвлетных коэффициентов, SPIHT переупорядочивает их биты. При этом, первые биты будут нести наиболее важную информацию, в то время как последние – лишь незначительные, уточняющие детали. Такое упорядочение данных часто называют прогрессивным. При прогрессивной передаче изображения, декодер, получая очередные порции закодированных данных, может последовательно улучшать и уточнять полученное им изображение. При этом вначале декодером будут прорисованы основные цветовые и яркостные переходы, а уж затем второстепенные контуры и малозаметные детали.

Прогрессивное упорядочение, помимо всего прочего, позволяет точно задавать требуемую степень сжатия. Действительно, можно сохранить лишь требуемое количество первых битов закодированного изображения, а оставшийся «хвост» просто отбросить, так как он несет сравнительно не много информации.

3.2 Прогрессивная передача изображений

Пусть $X = \{X(i,j)\},\ i \in 0:M-1,\ j \in 0:N-1$ исходное изображение размера $M \times N$ точек, где N и M – ширина и высота изображения соответственно. Пусть далее $Y = \{Y(i,j)\},\ i \in 0:M-1,\ j \in 0:N-1$ вейвлетные коэффициенты исходного изображения. Символом $\Omega(\cdot)$ будем обозначать прямое, а символом $\Omega^{-1}(\cdot)$ - обратное вейвлетное преобразование, т.е. $Y = \Omega(X)$ и $X = \Omega^{-1}(Y)$. Кодированию подвергаются элементы массива $Y = \Omega(X)$, округленные до ближайших целых.

При прогрессивной передаче изображения, декодер вначале работы заполняет массив $\hat{Y} = \{\hat{Y}(i,j)\},\ i \in 0:M-1,\ j \in 0:N-1$ нулями и обновляет значения его компонент в соответствии с поступающими данными. После получения очередного точного или приблизительного значения коэффициента, декодер может реконструировать изображение по формуле $\hat{X} = \Omega^{-1}(\hat{Y})$. Ясно, что чем ближе $\hat{Y}$ будет к Y, тем ближе $\hat{X}$ будет к X. Другими словами, качество реконструируемого изображения будет расти в процессе декодирования очередных порций данных. Отсюда и главная идея: выбрать наиболее важную информацию и передать ее в первую очередь. Ясно, что б*о*льшие по абсолютной величине коэффициенты должны передаваться в первую очередь, так как содержат больше информации. Такой же подход применим и для передачи отдельных битов самих коэффициентов. Так как старшие биты несут больше информации, их необходимо передавать первыми. Это, так называемый, метод битовых плоскостей (bit-plane), который часто используется для прогрессивной передачи данных.

Далее представим алгоритм прогрессивной передачи, сочетающий в себе два описанных выше метода: упорядочение коэффициентов по величине и кодирование их битовых плоскостей. Для упрощения изложения будем считать, что координаты коэффициентов передаются явным образом.

3.3 Передача значений коэффициентов

Предположим, что коэффициенты отсортированы по убыванию длины их бинарного представления (при этом ведущий бит не учитывается). Другими словами, задана некоторая перестановка индексов

$$\eta(k):\{0:NM-1\}\to\{0:M-1\}\times\{0:N-1\},$$

такая, что

$$\lfloor\log_2|Y(\eta(k))|\rfloor\geq\lfloor\log_2|Y(\eta(k+1))|\rfloor,\ k=0:NM-2.$$

Знак	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
5	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	—	>	1	1	0	0	0	0	0	0	0	0	0	0	0
3	—	—	—	>	1	1	1	1	0	0	0	0	0	0	0
2	—	—	—	—	—	—	—	>	1	1	1	1	1	1	1
1	—	—	—	—	—	—	—	—	—	—	—	—	—	—	>
0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	>

Таблица 1: Двоичное представление упорядоченных коэффициентов

На таблице 1 схематично изображено множество упорядоченных по величине коэффициентов. Каждый столбец k таблицы 1 содержит биты коэффициента $Y(\eta(k))$. Биты в самой верхней строке – это знаки соответствующих коэффициентов. Строки пронумерованы снизу вверх, где в нижней строке записаны самые младшие биты, а в верхней (под знаком) – самые старшие.

Теперь предположим, что наряду с информацией об упорядочении коэффициентов (отображение $\eta(k)$), декодер получает числа μ_n. Значение μ_n равно количеству коэффициентов $Y(i,j)$, таких, что $2^n\leq|Y(i,j)|<2^{n+1}$. В нашем примере $\mu_5=2$, $\mu_4=2$, $\mu_3=4$ и т.д.

Т.к. все биты в строке содержат одинаковое количество информации, можно определить наиболее эффективный алгоритм их передачи. Необходимо последовательно передавать биты каждой строки, в направлении, указанном стрелками. Начинать нужно с верхней строки. Заметим, что передавать ведущие «0» и первую «1» не нужно, т.к. эту информацию можно восстановить, используя $\eta(k)$ и μ_n.

Описанный выше метод прогрессивной передачи коэффициентов может быть реализован при помощи следующего алгоритма.

Алгоритм прогрессивной передачи значений коэффициентов

1. Найти и передать $n=\lfloor\log_2(\max_{(i,j)}|Y(i,j)|)\rfloor$.
2. Передать μ_n, координаты $\eta(k)$ и знаки коэффициентов $Y(\eta(k))$, для которых $2^n\leq|Y(\eta(k))|<2^{n+1}$.
3. Передать *n*-й значимый бит всех коэффициентов $Y(i,j)$, для которых $|Y(i,j)|\geq2^{n+1}$ (т.е. тех, чьи координаты были переданы на предыдущем проходе).
4. Уменьшить *n* на 1 и идти на шаг 2.

Данный алгоритм можно остановить в любой момент, когда будет достигнута требуемая степень сжатия или возмущение. Обычно, хорошее качество можно получить, передав лишь небольшую часть данных таким способом.

По сути, SPIHT предлагает лишь одно значимое усовершенствование к этому алгоритму. В то время как бо́льшая часть передаваемых битов тратится на передачу $\eta(k)$ в явном виде, SPIHT передает эту информацию в гораздо более компактном, неявном виде. Подробнее об этой неявной передаче см. в статьях [6, 7].

4. Численные эксперименты

В качестве тестового было использовано полноцветное изображение SIMAKOV размера 1024×1280 точек. К нему применялось 8-ми уровневое вейвлетное преобразование. Схожие результаты получались для стандартных изображений LENA, BARBARA и GOLDHILL.

Для оценки качества восстановленных изображений использовалась значение PSNR (Peak Signal-to-Noise Ratio). Для полутоновых изображений используется формула

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{N}\sum_{i,j=0}^{N-1}(x(i,j)-y(i,j))^2}\right),$$

а для полноцветных, формула

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{3N}\sum_{i,j=0}^{N-1}((x_r(i,j)-y_r(i,j))^2+(x_g(i,j)-y_g(i,j))^2+(x_b(i,j)-y_b(i,j))^2)}\right),$$

где N – количество пикселей в изображении, а нижние индексы r,g,b соответствуют красному, зеленому и синему каналам соответственно.

Степень сжатия	$F_2+\Phi_2$	$F_2+\Phi_3$	$F_3+\Phi_2$	$F_3+\Phi_3$	Добеши 9/7
1:2	46.870 dB	**46.893 dB**	46.654 dB	46.856 dB	46.528 dB
1:4	46.087 dB	**46.120 dB**	45.952 dB	46.142 dB	45.968 dB
1:8	44.604 dB	**44.628 dB**	44.462 dB	44.617 dB	44.623 dB
1:16	42.606 dB	**42.641 dB**	42.501 dB	42.628 dB	42.651 dB
1:32	41.213 dB	**41.244 dB**	41.115 dB	41.234 dB	41.363 dB
1:64	40.438 dB	**40.462 dB**	40.363 dB	40.439 dB	40.491 dB
1:128	39.815 dB	**39.839 dB**	39.743 dB	39.812 dB	39.948 dB
1:256	39.385 dB	**39.389 dB**	39.288 dB	39.324 dB	39.422 dB
1:512	**38.441 dB**	38.430 dB	38.274 dB	38.293 dB	38.527 dB
1:1024	**36.632 dB**	36.621 dB	36.347 dB	36.350 dB	36.790 dB

Из приведенной таблицы видно, что преобразование Баттерворта показывает результаты, сравнимые со всемирно известным преобразованием Добеши 9/7, а в ряде случаев даже превосходит его. Наилучшим из семейства преобразований Баттерворта, описанных в этой статье, нужно признать преобразование $F_2+\Phi_3$.

5. Исходный текст

Все описанные в этой статье алгоритмы реализованы в виде программной библиотеки на языке C. Ее исходный текст, полезные утилиты и тестовые изображения можно загрузить из сети Интернет, с сайта автора http://www.entropyware.info

Работа выполнена при поддержке гранта РФФИ N 02-01-00084.

6. Библиография

[1] В. А. Желудев, А. Б. Певный. Вейвлетное преобразование Баттерворта и его реализация при помощи рекурсивных фильтров // Ж. вычисл. мат. и матем. физ. 2002. Т. 42. N 4. С. 571-582.

[2] Amir Z. Averbuch, Valery A. Zheludev. A library of spline based biortogonal wavelet transforms for image compression // Preprint. 2002.

[3] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies. Image Coding Using Wavelet Transform // IEEE Transactions on Image Processing. 1992. V. 1. N 2. P. 205-220.

[4] A. Cohen, I. Daubechies, J. C. Feauveau. Biorthogonal Bases of Compactly Supported Wavelets // Communications on Pure and Applied Mathematics. 1992. V. 45. N 5. P. 485-560.

[5] I. Daubechies, W. Sweldens. Factoring Wavelet Transforms Into Lifting Steps // J. Fourier Anal. Appl. 1998. V. 4. N 3. P. 245-267.

[6] Amir Said, William A. Pearlman. A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees // IEEE Trans. on Circuits and Systems for Video Technology. 1996. V. 6. P. 243-250.

[7] Shapiro, J. M. Embedded image coding using zerotrees of wavelet coefficients // IEEE Transactions on Signal Processing. 1993. V. 41. N 12. P. 3445-3462.

[8] I. H. Witten, R. M. Neal, J. H. Cleary. Arithmetic coding for data compression // CACM. 1987. V. 30. N 6. P. 520-540.

[9] А. П. Петухов. Биортогональные базисы всплесков с рациональными масками и их приложения // Труды СПбМО. 1999. Т. 7. С. 168-193.

[10] D. A. Huffman. A method for the construction of minimum-redundancy codes // Proc. Inst. Radio Engineers. 1952. V. 40. N 9. P. 1098-1101.

[11] А. В. Симаков. Код Хаффмана. http://www.entropyware.info/HuffmanCode/huffcode.html

[12] А. В. Симаков. Сайт, посвященный коду Хаффмана. http://www.webcenter.ru/~xander

[13] В. А. Кирушев. Быстрый алгоритм сжатия изображений // Вестник молодых ученых. Прикладная математика и механика. 1997(1). С. 4-10.

[14] Семинар «Всплески и их приложения». http://www.math.spbu.ru/user/dmp

[15] Ватолин Д., Ратушняк А., Смирнов М. Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с. http://www.compression.ru

[16] В. А. Желудев. Домашняя страница. http://www.cs.tau.ac.il/~zhel