# Вейвлет-сжатие.

# JPEG2000

Стандарт JPEG2000 разработан той же группой экспертов в области фотографии, что и JPEG. Формирование JPEG как международного стандарта было закончено в 1992 году. В 1997 стало ясно, что необходим новый, более гибкий и мощный стандарт, который и был доработан к зиме 2000 года. Основные отличия алгоритма JPEG2000 от JPEG заключаются в следующем:

1. Лучшее качество изображения при сильной степени сжатия.

2. Поддержка кодирования отдельных областей с лучшим качеством. Здесь предполагается, что имеется возможность человеку «на глаз» определить: какие области можно сжать с меньшим качеством, а для каких оставить прежнее. В результате при одинаковом субъективном качестве изображения могут быть достигнуты более высокие степени сжатия.

3. Алгоритм основан на вейвлет-преобразовании. Появилась возможность постепенного проявления изображения при его загрузке по сетям связи.

4. Бит-ориентированное арифметическое кодирование. Арифметическое кодирование предполагалось использовать еще в стандарте JPEG, но тогда оно было защищено патентами. Сейчас срок действия основного патента истек и стало возможным его применение в JPEG2000.

5. Поддержка сжатия без потерь. Благодаря этой опции появилась возможность применения JPEG2000 при сжатии медицинских изображений, полиграфии, для задач распознавания текста и т.п., критичных к потерям информации.

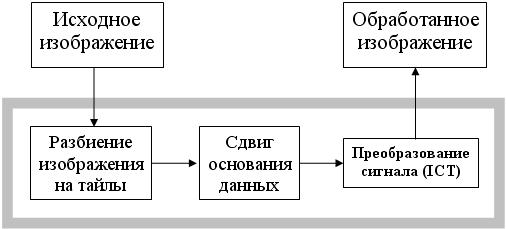
6. Поддержка сжатия однобитных (2-цветных) изображений.

На следующей диаграмме представлены основные шаги работы кодера JPEG2000.



Рассмотрим более подробно каждый из шагов

### Препроцессинг



* В отличие от JPEG кодер JPEG2000 не требует разбиения изображения на малые квадратные блоки, так как используемое в ходе работы алгоритма ДВП (дискретное вейвлетное преобразование) работает на фрагментах любого размера. С другой стороны иногда, в случае, если объем памяти, доступный кодеру для работы, меньше, чем объем памяти, необходимый для кодирования всего изображения, выполняется разбиение изображения на квадратные тайлы, которые кодируются независимо друг от друга. Далее будет рассматриваться кодирование одного тайла.
* В JPEG2000 выполняется центрирование яркости каждой компоненты RGB изображения перед преобразованием в цветовое пространство YUV. Это делается для выравнивания динамического диапазона, что приводит к увеличению степени сжатия. Формулу преобразования можно записать так:

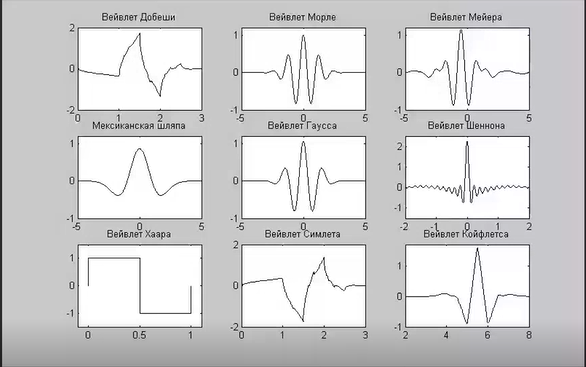
http://sernam.ru/htm/lect_cod/files/cod_17.files/image001.gif, где http://sernam.ru/htm/lect_cod/files/cod_17.files/image002.gif - число бит/пиксел для кодируемого сигнала. Т.е. происходит отображение [0,2p - 1] -> [-2p-1, 2p-1 - 1].

* Если изображение содержит более одной компоненты, то они кодируются по отдельности. В связи с этим, на данном этапе выполняется перевод графического изображения из его компонентного представления в цветоразностное, яркостное представление (ICT — Irreversible Color Transform). В связи с тем, что человеческий глаз более восприимчив к яркостному сигналу, чем к цветовому, это преобразование позволит добиться больших результатов сжатия при меньших визуальных потерях. Далее будет рассматриваться кодирование одной отдельной компоненты.

### ДВП

JPEG2000 использует дискретное вейвлетное преобразование (Discrete Wavelet Transformation) – некий ортогональный базис(функции), по которому можно разложить исходный сигнал, для разбиения изображения на высокочастотные и низкочастотные области. ДВП обрабатывает каждую строку и столбец исходного изображения с помощью частотного фильтра.

Вместо синуса и косинуса (как в ДКП) используются другие функции. Наиболее типичные вайвлеты:



Они могут гладкими или нет(не везде непрерывны). Но это не мешает, так как эти вейвлеты задаются в матричной форме, и матрицы обладают свойством ортогональности.

**Основные св-ва вейвлетов:**

1. **Средний интеграл от –оо до +оо по данной функции= 0,** то есть, если провести линию по оси Х и померить площадь вейвлета над ней и под ней, то площади должны совпасть.
2. **Быстрое затухание при стремлении к бесконечности,** то есть, ее значимая область представляет собой достаточно короткий отрезок.

**Вейвлет Хаара**

Представляет из себя полусумму(верхний пик) и полуразность(нижний пик)

**НЧ = (a+b)/2**

**ВЧ = (a-b)/2**

Пример:

Характерный для видео сигнал исходный**(одномерный):** 255 254 254 255 253 255 252 251 – яркости

Разложение по вейвлету Хаара:

Первый проход:

НЧ = 254.5 254.5 254 251.5

ВЧ = 0.5 0.5 1 0.5

Второй проход (итеративно по первому):

НЧ = 254.5((254.5+254.5) /2) 252.75

ВЧ = 0 1.25

3 проход:

НЧ = 253.625

ВЧ = 0.875

Тогда закодированное изображение без потерь качества можно представить следующим образом:

НЧ3 ВЧ1 ВЧ2 ВЧ3 – исходные 8 чисел также представлены 8 числами.

Обратным ходом можно это восстановить.

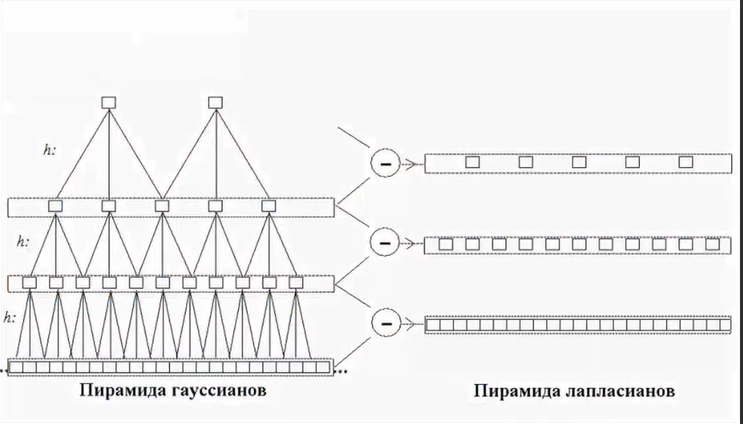
Мы добились того, что низкая частота всего одна, все остальное – высокие частоты, близкие к нулю, которые удобно сжимать, квантовать и тд.

Для двумерного сигнала будет 4 фильтра: полусумма/полуразность всех, а также полусумма/полуразность по вертикали/диагонали.

Графически: исходный сигнал с помощью 4ех фильтров получить 4 более мелких изображения, из которых верхнее левое будет содержать уменьшенную исходную картинку, справа – полусумму по горизонтали, полуразность по вертикали, слева внизу – наоборот, справа – полуразность по диагонали (содержит наименьшее количество значимой информации, поэтому сразу отбрасывается). Далее, берем левый верхний и проделываем то же самое (**ниже есть картинка**).

После окончания этого процесса смотрим, какой квадрат с каким качеством пойдет в сжатый файл путем округления, квантования высоких и низких частот.

Иерархическое представление данного подхода – **пирамида лапласианов и гауссианов.**



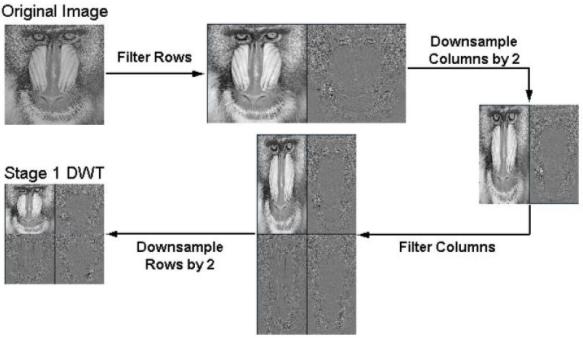
Это была **общая идея вейвлет-сжатия.**

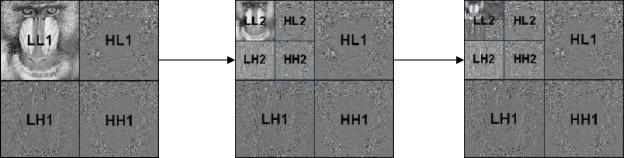
**Хаар в реальности не используется, хоть он и самый быстрый, а используются ортогональные и биортогональные вейвлеты (вейвлет Добеши)**

В связи с тем, что каждый проход с использованием частотного фильтра на выходе увеличивает объем информации в два раза, после обработки размер изображения уменьшается в два раза. После одного этапа ДВП обрабатываемый фрагмент делится на четыре сегмента:

* LL – низкие частоты по строкам и столбцам
* HL – высокие частоты по строкам и низкие по столбцам
* LH – низкие частоты по строкам и высокие по столбцам
* HH – высокие частоты по строкам и столбцам

По стандарту количество этапов может быть от 0 до 32. Для обычного изображения используют от 4-х до 8-ми этапов. На каждом следующем этапе обрабатывается только низкочастотная область (LL), так как в высокочастотных областях обычно не содержится важной информации.

  
  
**Рис. 10.** Схема выполнения одного этапа ДВП.

  
  
**Рис. 11.** Последовательное применение ДВП к изображению.

Для вычисления вейвлет преобразования в стандарте определены два биортогональных вейвлет фильтра – фильтр Добеши(9,7) для сжатия с потерями и фильтр с целочисленными коэффициентами для сжатия без потерь(5,3). В расширениях стандарта разрешается применение любых фильтров, а также не только традиционное разбиение, но и произвольное.

Коэффициенты приведены <http://sernam.ru/cod_17.php>

***Фильтр с целочисленными коэффициентами***

В случае сжатия без потерь прямое ВП в одномерном случае будет иметь вид

http://sernam.ru/htm/lect_cod/files/cod_17.files/image010.gif,

http://sernam.ru/htm/lect_cod/files/cod_17.files/image011.gif,

что соответствует лифтинговой схеме для вычисления низкочастотной и высокочастотной составляющих. Так как преобразование должно быть без потерь, то сделаем так, чтобы выходные значения http://sernam.ru/htm/lect_cod/files/cod_17.files/image012.gif были целочисленными. Это достигается следующими формулами:

http://sernam.ru/htm/lect_cod/files/cod_17.files/image013.gif

#### Преобразование Добеши

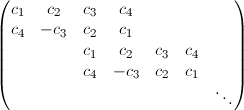
#### Будем брать по четыре значения, смещаясь каждый раз на два. То есть, если исходная последовательность — 1, 2, 3, 4, 5, 6,…, N-1, N, то будем брать четвёрки (1, 2, 3, 4), (3, 4, 5, 6) и т. д. Последняя четвёрка «кусает последовательность за хвост»: (N-1, N, 1, 2).

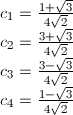
Построим два фильтра: высокочастотный и низкочастотный. Каждую четвёрку будем заменять на два числа. Так как четвёрки перекрываются, то количество значений после преобразования не изменится.

Для удобства подсчёта обратной матрицы потребуем также ортогональности преобразования. Тогда поиск обратной матрицы сведётся к транспонированию.

Пусть значения яркостей в четвёрке равны x, y, z, t. Тогда первый фильтр запишем в виде:  
http://habrastorage.org/storage2/234/86f/ff2/23486fff2b0e3c009671073e7d83d770.png

Четыре коэффициента, образующих вектор-строку матрицы преобразования, пока нам неизвестны.  
  
Чтобы вектор-строка коэффициентов второго фильтра был ортогонален первому, возьмём те же коэффициенты но переставим их и поменяем знаки:  
http://habrastorage.org/storage2/912/c56/4df/912c564dfcfc2774a537338ca8a99a6c.png

Матрица преобразования будет иметь вид.  
  


Требование ортогональности выполняется для первой и второй строк автоматически. Потребуем, чтобы строки 1 и 3 тоже были ортогональны:  
  
http://habrastorage.org/storage2/dec/49c/0c7/dec49c0c7db1844ea1c86387692937b0.png  
  
Векторы должны иметь единичную длину (иначе определитель будет не единичным):  
  
http://habrastorage.org/storage2/1c7/eb5/c04/1c7eb5c04c25eddcbb6a0b0a553d22b7.png  
  
Преобразование должно обнулять цепочку одинаковых значений (например, (1, 1, 1, 1)):  
  
http://habrastorage.org/storage2/eba/c60/ab8/ebac60ab84de7f47f9419c7a093e2c52.png  
  
Преобразование должно обнулять цепочку линейно растущих значений (например, (1, 2, 3, 4)):  
  
http://habrastorage.org/storage2/483/773/83e/48377383ece7e8e838b859071454b4a1.png  
  
Получили 4 уравнения, связывающие коэффициенты. Решая их, получаем:  
  
  
  
Подставив их в матрицу, получаем искомое преобразования. После его применения к фотографиям получим больше нулей и малых коэффициентов, что позволит сжать изображение сильнее.

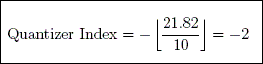
**Дефекты**

Поскольку вейвлет-сжатие – сжатие с потерями, для него характерны аналогичные с JPEG дефекты – потеря мелких деталей, эффект Гиббса (так как раскладываем на функции), дефекты цветового прореживания (в зависимости от кодека).

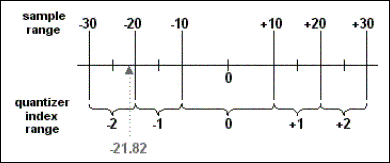
**Квантование**

Для округления коэффициентов ДВП используется постоянный квантователь с мертвой зоной. (рис. 14) Для каждого фрагмента используется постоянное значение шага округления для всех коэффициентов этого фрагмента. Формула вычисления округленных значений представлена на рисунке 12. Здесь *y* - исходное значение коэффициента, *sign(y)* определяет знак коэффициента, а Δb - значение шага округления. *Мертвая зона* квантователя - это интервал диапазоном 2Δb около нуля, она дает большее количество нулей на выходе.

|  |  |
| --- | --- |
| Формула округления коэффициентов |  |
| **Рис. 12.** Формула округления коэффициентов. |  |
| Схема работы постоянного квантователя с мертвой зоной |  |
| **Рис. 14.** Схема работы постоянного квантователя с мертвой зоной. |  |
|  |  |



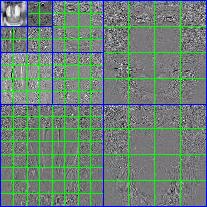
**Рис. 13.** Пример округления коэффициента.



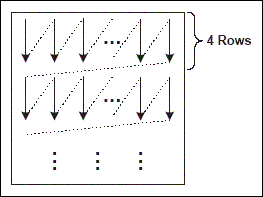
**Рис. 15.** Пример работы квантователя.

### Кодирование

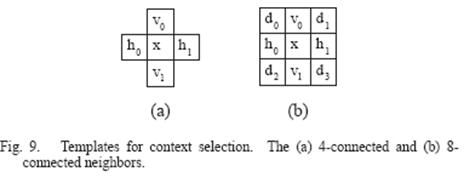
Кодирование полученных округленных коэффициентов выполняется поблочно. По стандарту JPEG2000 непосредственно перед кодированием фрагменты разбиваются на достаточно малые блоки (например, размером 32x32 или 64x64) так, чтобы все блоки одного фрагмента были одинакового размера. Разбиение на блоки выполняется для того, чтобы осуществить более гибкую организацию сжатой информации для повышения помехоустойчивости и так далее.

  
**Рис. 16.** Пример разбиения фрагментов изображения на блоки.

В JPEG2000 каждый блок кодируется по отдельности. Алгоритм кодирования обходит матрицу коэффициентов округления каждого блока полосами, как показано на рисунке 17. Блоки разбиваются на блоки с номинальной высотой 4. Далее полосы сканируются сверху вниз, а колонки в каждой полосе обходятся слева направо.

  
**Рис. 17.** Порядок кодирования блоков.

После разделения коэффициентов в кодовые блоки они независимо кодируются, используя битовый кодер. Для этих целей используется контекстно-зависимый адаптивный бинарный арифметический кодер, известный как MQ-кодер. После разложения каждого блока на битовые плоскости они кодируются независимо. При этом контекст определяется на основе ближайших отсчетов, как показано на рис. 9.



В рамках стандарта JPEG2000 имеется возможность выбирать отдельные области для их лучшего представления, по сравнению с остальным изображением. В этом случае мы можем в разы увеличить степень сжатия за счет изменения качества разных участков изображений.

Проблемой этого подхода является то, что необходимо каким-то образом узнавать расположение наиболее важных для человека участков изображения. Например, таким участком может быть лицо человека на фоне природы. Если при сжатии будет размыто дерево на фоне, то это не так критично как размытия лица.

Работы по автоматическому выделению таких областей активно ведутся. В частности, созданы алгоритмы автоматического выделения лиц на изображениях. Продолжаются исследования методов выделения наиболее значимых контуров и т.д. Однако очевидно, что универсальный алгоритм в ближайшее время создан не будет, поскольку для это требуется критерий визуального восприятия, которого на сегодняшний день отсутствует.

Вместо этого вполне реально применение автоматизированных алгоритмов сжатия, в которых значимые области определяются самим человеком. Данный подход уменьшает количество возможных областей применения такого алгоритма, но позволяет достигать больших степеней сжатия. Такой подход целесообразно применять, если:

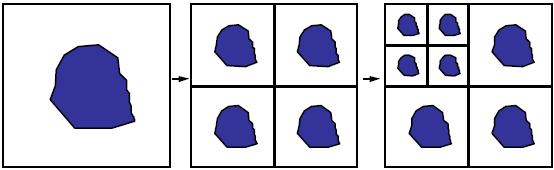
1. Для приложения должна быть критична степень сжатия, причем настолько, что возможен индивидуальный подход к каждому изображению.

2. Изображение сжимается один раз, а разжимается множество раз.

В качестве примеров приложений, удовлетворяющим этим требованиям, можно привести практически все мультимедийные продукты на DVD. Также сюда относятся DVD энциклопедии и игры, где графика занимает до 70% всего объема. При этом технология производства дисков позволяет сжимать каждое изображение индивидуально, максимально повышая степень сжатия.

Интересным примером являются WWW-сервера. Для них тоже, как правило, выполняются оба изложенных выше условия. При этом совершенно не обязательно индивидуально подходить к каждому изображению поскольку по статистике 10% изображений будут запрашиваться 90% раз. То есть для крупных справочных или игровых серверов появляется возможность уменьшать время загрузки изображений и степень загруженности каналов связи адаптивно.

В JPEG2000 используется однобитное изображение-маска, задающее повышение качества в данной области изображения. Поскольку за качество областей у нас отвечают коэффициенты ДВП преобразования во 2, 3 и 4 квадрантах, то маска преобразуется т.о., чтобы указывать на все коэффициенты, соответствующие областям повышения качества:



Эти области обрабатываются далее другими алгоритмами (с меньшими потерями), что и позволяет достичь более высоких показателей сжатия.

**Организация данных**

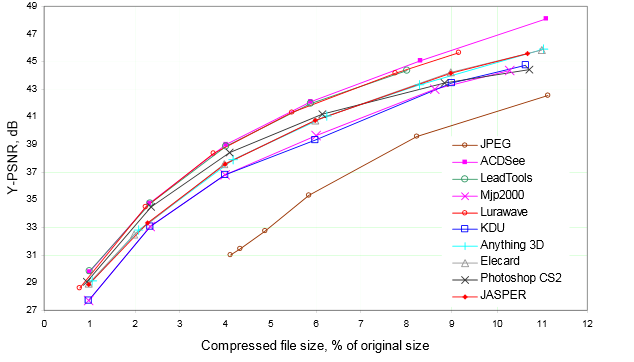
Важным преимуществом рассматриваемого стандарта является возможность доступа к отдельным элементам изображения без полного декодирования его представления. Обеспечивается такая возможность, во-первых, разбиением исходного изображения на непересекающиеся области (тайлы), которые кодируются как отдельные изображения, а во-вторых, представлением кода отдельного тайла в виде частей (слоев), каждая из которых является суммарным кодом коэффициентов, соответствующих некоторой его (тайла) области. Слои в свою очередь делятся на так называемые пакеты, содержащие код блоков коэффициентов на разных уровнях декомпозиции. Для того, чтобы декодировать какую-либо область изображения, достаточно определить, каким тайлам она принадлежит, и какие слои, относящиеся к этим тайлам, содержат код блоков коэффициентов, необходимых для восстановления требуемой области.

**Наглядные данные:**







**Литература**

<http://rain.ifmo.ru/cat/view.php/theory/data-compression/jpeg-2006>

<http://www.vocord.ru/download/articles/jpeg2000.pdf>

<http://sernam.ru/cod_17.php>

<http://habrahabr.ru/post/168517/>

<http://storage.tusur.ru/files/292/%D0%A0%D0%A2%D0%A1-1102_%D0%A1%D0%B0%D0%BC%D0%B0%D1%80%D0%B8%D0%BD.pdf>

http://hall.org.ua/halls/wizzard/books/articles-cg/jpeg2000\_codec\_comparison\_ru.pdf

# JPEG-LS

Формат **JPEG-LS** был основан на формате **LOCO-I** (Low Complexity Lossless Compression for Images). Алгоритм сжатия без потерь LOCO-I, принятый за основу при разработке стандарта JPEG-LS, впервые предусматривал не только lossless, но и near lossless режим (сжатие с ограниченными, задаваемыми пользователем потерями). В отличие от JPEG2000 lossless mode, JPEG-LS получился по-настоящему удачным: при большей эффективности сжатия новый стандарт обеспечивает высокую скорость компрессии/декомпрессии и не слишком требователен к ресурсам компьютера.

Важно понимать, что формат JPEG-LS:

* не является расширением или модификацией метода JPEG;
* не использует ни DCT (ДКП), ни арифметическое кодирование;
* использует слабое квантование только в моде «почти без потерь»

Сжатие данных без потерь состоит из двух отдельных независимых частей: моделирования (modeling) и кодирования (coding). Определим некоторые термины, которые будем активно использовать в дальнейшем:

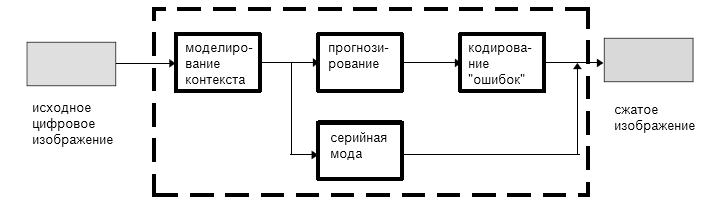
Кодер (Encoder)

«отвечает» за процесс кодирования, а именно: получает на вход исходное изображение в цифровом формате и все необходимые параметры, определенные стандартом, и с помощью специального набора процедур создает набор данных, содержащих сжатое изображение

Декодер (Decoder)

«отвечает» за процесс декодирования и преобразование фрагментов, а именно: получая на вход данные со сжатым изображением и все необходимые параметры, выводит реконструированное изображение

Декодер JPEG-LS мало отличается от кодера, поэтому этот алгоритм сжатия можно назвать почти симметричным. Приведем упрощенную схему, показывающую принципы кодирования:

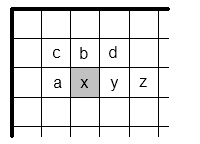
  
**Рис. 21.** Упрощенная схема кодирования без потерь.

Порядок, в котором пиксели подаются на обработку процедурам кодирования, определен так: left-to-right (слева направо) и top-to-bottom (сверху вниз) по компоненте.

Для прогнозирования текущего пикселя x используются пиксели контекста a, b, c, d. В зависимости от контекста кодер выбирает моду: **серийную (run mode)** или **регулярную (regular mode)**. **Серийная мода** выбирается, если y и z скорее всего будут совпадать, **регулярная** – в противном случае. Сделаем тут замечание, связанное с наличием опции **«почти без потерь»**: при включении этой опции серийная мода будет выбрана, если y и z будут почти совпадать в соответствии с параметром допустимого отклонения NEAR.

В случае использования серийной моды мы начинаем просмотр текущей строки с пикселя x и находим наибольшую длину серии пикселей, совпадающих с контекстным пикселем a. Таким образом, в пределах текущей строки мы получаем серию одинаковых пикселей, совпадающих по значению с известным нам пикселем a. Осталось только закодировать длину серии. (Это делается с помощью массива J из 32 элементов). Вы уже могли догадаться, что при включенной опции «почти без потерь» выбирается серия пикселей, близких к a с помощью параметра NEAR.

Теперь рассмотрим наши действия в случае использования регулярной моды. Для вычисления прогноза пикселя x (Px) используются величины пикселей a, b и c. Затем вычисляется так называемая ошибка прогноза (Errval). Ее значение равно разности значения x и Px. Errval корректируется некоторым членом, зависящим от контекста, а затем кодируется с помощью кодов Голомба. Код Голомба зависит от a, b, c, d и Errval этих же пикселей, которые хранятся в специальных массивах A и N. При включении опции «почти без потерь» перед кодированием ошибка прогноза ещё дополнительно квантуется.

  
**Рис. 23.** Контекстные пиксели для пикселя x.

**Коды Голомба**

Код Голомба неотрицательного целого числа http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image001.gif [Golomb 66] может быть эффективным кодом Хаффмана. Этот код зависит от выбора некоторого параметра http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image002.gif. Прежде всего необходимо вычислить две величины http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image003.gif, http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image004.gif (где выражение http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image005.gif обозначает округление http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image006.gif), а затем построить код из двух частей; первая часть - это число http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image007.gif, закодированное с помощью унарного кода (см. стр. 195), а вторая - двоичное выражение для http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image008.gif, состоящее из http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image009.gif бит (для малых остатков) или из http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image010.gif бит (для больших). Если взять http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image011.gif, то три возможных остатка 0, 1 и 2 будут кодироваться как 0, 10 и 11. Выбрав http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image012.gif, получаем 5 остатков от 0 до 4, которые кодируются как 00, 01, 100, 101 и 110. В табл. 3.60 приведены некоторые коды Голомба при http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image011.gif и http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image012.gif.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image001.gif | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image011.gif | 0|0 | 0|10 | 0|11 | 10|0 | 10|10 | 10|11 | 110|0 | 110|10 | 110|11 | 1110|0 |
| http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image012.gif | 0|00 | 0|01 | 0|100 | 0|101 | 10|110 | 10|00 | 10|01 | 10|100 | 10|101 | 110|110 |

Табл. 3.60. Некоторые коды Голомба при http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image011.gif и http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image012.gif.

Предположим, что входной поток данных состоит из целых чисел, причем вероятность числа http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image001.gif равна http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image013.gif. Здесь http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image014.gif - некоторый параметр, http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image015.gif. Можно показать, что коды Голомба будут оптимальными кодами для этого потока данных, если http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image002.gif выбрать из условия

http://www.sernam.ru/archive/arch.php?path=../htm/book_sel/files.book&file=sel_61.files/image016.gif.

Имея такие данные на входе, легко породить наилучшие коды переменной длины, не прибегая к алгоритму Хаффмана.

**Порядок кодирования JPEG-LS**

Обычно, JPEG-LS используется как метод сжатия без потери информации. В этом случае восстановленный файл изображения идентичен исходному файлу. В моде почти без потерь исходный и реконструированный образ могут отличаться. Будем обозначать реконструированный пиксел Rp, а исходный пиксел - p. При кодировании верхней строки контекстные пикселы, и отсутствуют, поэтому их значения считаются нулевыми. Если текущий пиксел находится в начале или конце строки, то пикселы a, c или d не определены. В этом случае для a и d используется реконструированное значение Rb пиксела b (или нуль для верхней строки), а для c используется реконструированное значение a при кодировании первого символа предыдущей строки. Все это означает, что кодер должен выполнить часть работы декодера, реконструируя некоторые пикселы.

Кодер начинает работу со следующих трех шагов:

Вычисление значений градиентов D1 = Rd - Rb, D2 = Rb - Rc, R3 = Rc - Ra.

Если все эти величины равны нулю (или в моде почти без потерь их абсолютные значения не превосходят порога NEAR), то кодер переходит в серийную моду и ищет наибольшую длину серии пикселов, совпадающих с Ra иначе кодер прогнозирует пиксель х. Прогноз вычисляется в виде Px = а + b – с.

После того, как прогноз Рх найден, кодер вычисляет ошибку прогноза Errval в виде разности х – Рх.

В моде почти без потерь ошибка квантуется, и кодер использует это реконструированное значение Rx пикселя х так же, как это будет делать декодер. Основной шаг квантования заключается в следующем:

if (Errval > 0)   
  Errval = (Errval + NEAR) / (2 \* NEAR + 1);  
else   
  Errval = - (Errval - NEAR) / (2 \* NEAR + 1);

Кодирование в серийной моде делается иначе. Напомним, что кодер выбирает эту моду, когда обнаруживает последовательные пиксели x, чьи значения Ix совпадают и равны восстановленной величине Ra контекстного пикселя a. Для опции «почти без потерь» пиксели в серии должны иметь значения Ix, которые удовлетворяют неравенству |Ix - Ra| <= NEAR. Серия не должна выходить за пределы текущей строки. Длина серии кодируется (сам пиксель кодировать не нужно, поскольку он равен Ra), и если конец серии находится раньше конца строки, то после ее закодированной длины будет сразу записан код следующего пикселя (который прерывает серию). Две основные задачи кодера в этой моде состоят

1. в отслеживании серии и кодировании ее длины;
2. в кодировании пикселя, прервавшего серию.

### Заключение

Формат JPEG-LS разрабатывался, прежде всего, для хранения изображений в медицинских целях, то есть для тех случаев, когда важно иметь большое изображение без малейших потерь качества. Как уже говорилось, за основу был взят формат LOCO-I, разработанный в стенах «HP Labs». Затем он был доработан совместными усилиями «HP» и «Mitsubishi». Обе компании разрешили использовать их патенты на этот формат без оплаты лицензии, поэтому JPEG-LS можно встретить и в обычных программах для PC.

**Литература**

<http://rain.ifmo.ru/cat/view.php/theory/data-compression/jpeg-2006>

<http://sernam.ru/book_sel.php?id=61>