УДК 004.627

Э.А. Никонов, магистр,
М.В. Полякова, канд. техн. наук, доц.,
Одес. нац. политехн. ун-т

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМА КОДИРОВАНИЯ ДЛИН СЕРИЙ ДЛЯ СЖАТИЯ ИЗОБРАЖЕНИЙ БЕЗ ПОТЕРЬ

Е.О. Ніконов, М.В. Полякова. **Підвищення ефективності алгоритму кодування довжин серій для стиснення зображень без втрат.** Розроблено алгоритм стиснення без втрат КДС для зображень формату MS Windows® BMP з підвищеним ступенем стиску через його адаптацію до якостей BMP зображень. Проведено розрахунки коефіцієнтів ступеня стиску тестових та реальних BMP зображень запропонованим алгоритмом.

Э.А. Никонов, М.В. Полякова. **Повышение эффективности алгоритма кодирования длин серий для сжатия изображений без потерь.** Разработан алгоритм сжатия без потерь КДС для изображений формата MS Windows® BMP с повышенной степенью сжатия благодаря его адаптации к свойствам BMP изображений. Проведены расчеты коэффициентов степени сжатия тестовых и реальных изображений предложенным алгоритмом.

E.A. Nikonov, M.V. Polyakova. **Enhancing the efficiency of the run length encoding algorithm for lossless image compression.** The run length encoding lossless compression algorithm with enhanced compression rate was developed for images of MS Windows® BMP format by means of its adaptation to the BMP images properties. The compression rate factors were calculated for the test and real images by the proposed algorithm.

Во многих приложениях сжатие изображений без потерь является единственно допустимым способом сокращения объема данных. Одним из таких приложений является архивация медицинских или деловых документов, сжатие которых с потерями обычно запрещено по закону. Другим является обработка спутниковых изображений, где как применение, так и стоимость получения исходных данных делают сжатие с потерями нежелательным. Еще одним направлением является цифровая рентгенология, в которой потеря информации может ухудшить точность диагностики [1].

Известен алгоритм кодирования длин серий пикселей (далее — КДС) для сжатия без потерь изображений формата MS Windows® BMP (далее — формат BMP), одного из встроенных в операционную систему MS Windows®. Формат BMP поддерживает без сжатия изображения с 1, 4, 8, 16, 24 и 32 битами на пиксель, хотя изображения с 16 и 32 битами на пиксель используются редко, и со сжатием без потерь при помощи КДС изображения до 256-ти цветов или градаций серого, с 4 и 8 битами на пиксель. Однако при сжатии, объем памяти для хранения изображения сокращается только при наличии в последнем больших областей пикселей одинакового цвета, что ограничивает область применения указанного алгоритма [2].

Алгоритм КДС простейший среди алгоритмов сжатия без потерь, в частности алгоритмов LZW, Хаффмана, арифметического кодирования и т.д., ориентирован на изображения с небольшим количеством цветов, деловую и научную графику и в среднем сжимает изображения в 2…3 раза [3]. Его трудоемкость линейно зависит от количества пикселей в изображении.

Преимущества алгоритма КДС:

— простота реализации; не требуются сложные вычисления;

— малая ресурсоемкость; не требуется дополнительная память для архивации, а также разархивации изображений;

— высокое быстродействие; работа только с целочисленными данными.

Недостаток алгоритма КДС — низкая степень сжатия изображений, очень малая для фотоизображений с высокой детализацией, возможно даже, что полученное изображение будет занимать больше памяти, чем исходное, т.е произойдет т.н. “отрицательное сжатие” [3].

Таким образом, степень сжатия изображений алгоритмом КДС определяется свойствами изображений, под которыми, в частности, понимается наличие в последовательностях пикселей изображения серий с одинаковыми или повторяющимися значениями цвета. Тогда актуальна задача адаптации этого алгоритма к свойствам различных BMP изображений. Предлагается повысить степень сжатия изображений алгоритмом КДС путем его адаптации к свойствам изображений. Адаптированный алгоритм исследован на тестовых BMP изображениях и применен для сжатия реальных BMP изображений до 256-ти цветов или градаций серого с 4 и 8 битами на пиксель.

При кодировании длин серий данные сохраняются таким образом, что повторяющиеся значения пикселей замещаются числом, равным их количеству. Изображения с 4 битами на пиксель подвергаются сжатию алгоритмом RLE4, построенному на основе КДС, а изображения с 8 битами на пиксель сжимаются алгоритмом RLE8 на основе КДС [2].

Указанные алгоритмы сжатия BMP изображений работают следующим образом. В алгоритме сжатия RLE8 данные разбиваются на 2-байтовые пары. Первый байт — счетчик повторов — равен количеству повторяющихся значений пикселей, а второй — значению пикселя, который повторяется. В сжатом изображении с 8 битами на пиксель байтовая последовательность

$$08_{16}01_{16}$$

разворачивается в последовательность значений пикселей

$$01_{16}01_{16}01_{16}01_{16}01_{16}01_{16}01_{16}01_{16}.$$

Значение счетчика повторов, равное нулю, используется в качестве управляющего кода. Нулевое значение счетчика повторов, за которым следует другое нулевое значение, т.е.

$$00_{16}00_{16},$$

означает переход к следующей строке в изображении, причем, если строка изображения еще не закончена, то она дополняется нулями до конца, а затем осуществляется переход к следующей строке изображения. Нулевое значение счетчика повторов, за которым следует единица, т.е.

$$00_{16}01_{16},$$

означает конец изображения, причем, если изображение еще не закончено, то оно дополняется нулями до конца. Нулевое значение счетчика повторов, за которым следует двойка, т.е.

$$00_{16}02_{16},$$

изменяет текущее положение пикселей в изображении. Следующие за ними два байта означают, соответственно, число столбцов и строк, на которые следует выполнить переход в новое положение. Этот код позволяет пропускать большое число пикселей с нулевыми значениями.

Так, кодовая последовательность

$$04_{16}15_{16}00_{16}00_{16}02_{16}11_{16}02_{16}03_{16}00_{16}01_{16}$$

разворачивается в две строки пикселей изображения

$$15_{16}15_{16}15_{16}15_{16};$$

$$11_{16}11_{16}03_{16}03_{16} \text{ (конец изображения).}$$

Нулевой управляющий код с последующим значением, большим двух, указывает, что далее следуют литеральные байты, т.е. такие, которые должны быть скопированы в результирующее изображение без изменений. Этот код используется для хранения несжатых данных. Литеральные байты следуют непосредственно за счетчиком. Если значение счетчика нечетное, то после литеральных байтов сохраняется один байт-заполнитель [2], содержащий любое число от 0 до 255 (обычно 0), необходимый для дополнения данных до четного числа байт.

Из приведенного описания алгоритма сжатия RLE8 нетрудно заключить, что если изображение не содержит последовательных байтов с одинаковым значением цвета, итоговый размер сжатых данных изображения превысит размер эквивалентных несжатых.

Алгоритм сжатия RLE4 работает аналогично алгоритму сжатия RLE8. Основное различие заключается в том, что при кодировании значений цвета байт данных, который следует за зна-

чением счетчика повторов, содержит два значения пикселей. Значения четырех старших битов используются для кодирования первого и всех последующих нечетных пикселей в серии, в то время как значения четырех младших битов используются для кодирования второго и остальных четных пикселей. Таким образом, алгоритм сжатия RLE4 позволяет кодировать серии одинаковых значений пикселей либо серии двух чередующихся значений пикселей. Например, закодированная пара

$$05_{16}56_{16}$$

разворачивается в последовательность 4-битовых значений пикселей

$$5_{16}6_{16}5_{16}6_{16}5_{16}.$$

Все управляющие коды алгоритма сжатия RLE4 совпадают с управляющими кодами алгоритма сжатия RLE8, за исключением кодирования литеральных значений. Нулевой управляющий код с последующим значением, большим двух, задает число абсолютных значений пикселей, которые располагаются за управляющим кодом. Каждое из абсолютных значений пикселей упаковывается в 4 бита, причем старшие 4 бита располагаются в начале. Если значение счетчика не кратно четырем, то далее следует заполнить данные нулями так, чтобы число абсолютных байтов данных (не пикселей) было четным [2].

При реализации алгоритмов сжатия RLE4 и RLE8 учитывается, что любую последовательность пикселей можно закодировать как минимум двумя способами: как последовательности длин серий либо как литеральные данные. Например, последовательность пикселей

$$01_{16}01_{16}01_{16}02_{16}02_{16}03_{16}$$

можно закодировать как три последовательности длин серий

$$03_{16}01_{16}02_{16}02_{16}01_{16}03_{16};$$

либо как литеральные байты

$$00_{16}06_{16}01_{16}01_{16}01_{16}02_{16}02_{16}03_{16}.$$

Основная идея разработанного адаптивного алгоритма КДС — для каждой исходной последовательности пикселей выбрать лучший с точки зрения степени сжатия способ кодирования. Например, исходную последовательность пикселей

$$01_{16}02_{16}03_{16}04_{16}05_{16}05_{16}05_{16}$$

можно закодировать тремя способами:

— как пять последовательностей длин серий

$$01_{16}01_{16}01_{16}02_{16}01_{16}03_{16}01_{16}04_{16}03_{16}05_{16}\ (10\ \text{байт});$$

— как литеральные данные и одну последовательность длин серий

$$00_{16}04_{16}01_{16}02_{16}03_{16}04_{16}03_{16}05_{16}\ (8\ \text{байт});$$

— только как литеральные данные

$$00_{16}07_{16}01_{16}02_{16}03_{16}04_{16}05_{16}05_{16}05_{16}00_{16}\ (10\ \text{байт}).$$

Из приведенных трех способов второй является лучшим — закодированные данные занимают 8 байт. Однако, если далее будут следовать данные

$$0A_{16}0B_{16}0C_{16}0D_{16}0E_{16},$$

которые целесообразно кодировать как литеральные данные, то ситуация изменится в пользу третьего способа

$$01_{16}01_{16}01_{16}02_{16}01_{16}03_{16}01_{16}04_{16}03_{16}05_{16}00_{16}05_{16}0A_{16}0B_{16}0C_{16}0D_{16}0E_{16}00_{16}\ (18\ \text{байт});$$

$$00_{16}04_{16}01_{16}02_{16}03_{16}04_{16}03_{16}05_{16}00_{16}05_{16}0A_{16}0B_{16}0C_{16}0D_{16}0E_{16}00_{16}\ (16\ \text{байт});$$

$$00_{16}0C_{16}01_{16}02_{16}03_{16}04_{16}05_{16}05_{16}05_{16}0A_{16}0B_{16}0C_{16}0D_{16}0E_{16}\ (14\ \text{байт}).$$

Из приведенного примера видно, что лучший с точки зрения степени сжатия способ кодирования для подпоследовательности данных не всегда является лучшим для всей последовательности данных.

Стоит отметить, что значения данных в исходном несжатом BMP изображении означают не абсолютную величину интенсивности цвета (или градации серого) пикселя, а индекс в палитре цветов BMP изображения. Особенность алгоритма КДС для сжатия изображений состоит в том, что степень сжатия для некоторых изображений может быть существенно повышена за счет изменения порядка цветов в палитре BMP изображения [3]. Точнее, за счет перемещения одного из цветов на первое место в палитре изображения пиксели такого цвета будут иметь нулевое значение. Принцип поиска такого преобладающего цвета следующий. Необходимо подсчитать количество пикселей каждого цвета, встречающихся в конце каждой строки изображения и в конце изображения в целом, а также в достаточно больших одноцветных блоках изображения, пригодных для использования маркера изменения текущего положения пикселей в изображении. Цвет, счетчик пикселей которого принимает наибольшее значение, является преобладающим, его необходимо переместить на первое место в палитре цветов, тогда пиксели с таким цветом будут иметь нулевое значение в исходном несжатом изображении.

Известно также, что строки пикселей в BMP изображении при хранении могут располагаться в зависимости от значения высоты изображения формата BMP: при положительном значении обычно снизу-вверх, при отрицательном — сверху-вниз [2]. Использование маркера конца изображения позволяет, согласно описанию алгоритмов сжатия RLE4 и RLE8, не кодировать последние нулевые пиксели. Таким образом, если в конце верхней строки изображения и/или в конце нижней строки изображения есть пиксели с нулевым значением цвета, то можно выбрать такой порядок хранения строк данных, в котором будет большее количество нулевых пикселей в конце изображения (в верхней или нижней строке). Это позволит не кодировать нулевые пиксели в конце изображения.

Экспериментально установлено, что при сжатии для эффективного представления изображения минимальная длина последовательностей пикселей с одним значением цвета

$$L_1 = 5;$$

с разными значениями цвета для каждой пары соседних пикселей

$$L_2 = 3.$$

Реализация описанных идей позволяет сформулировать алгоритм КДС для сжатия изображений без потерь, адаптированный к свойствам изображений:

— найти преобладающий цвет и переместить его значение на первое место в палитре цветов изображения;

— определить наилучший с точки зрения степени сжатия порядок хранения строк в изображении;

— сжать исходное изображение при помощи адаптивного алгоритма КДС; определить наилучший с точки зрения степени сжатия способ кодирования для каждой исходной последовательности ненулевых пикселей:

— разбить последовательность пикселей изображения на подпоследовательности двух типов: содержащие пиксели только с одним значением цвета и содержащие пиксели с разными значениями цвета для каждой пары соседних пикселей в подпоследовательности;

— если подпоследовательность первого типа имеет длину меньше L_1 пикселей, а также заключена между двумя подпоследовательностями второго типа, то ее следует кодировать как литеральные данные, иначе — как последовательность длин серий повторяющихся пикселей;

— если подпоследовательность второго типа имеет длину меньше L_2 пикселей, ее следует кодировать как последовательности длин серий, иначе — как литеральные данные.

Для каждой исходной последовательности нулевых пикселей:

— пропустить нулевые пиксели и записать маркер конца строки в изображении, если исходная последовательность находится в конце строки изображения;

— пропустить нулевые пиксели и записать маркер конца изображения, если исходная последовательность находится в конце изображения;

— изменить текущее положение пикселей в изображении, если последовательность имеет достаточную длину для использования маркера изменения положения пикселей, иначе кодировать как последовательность длин серий повторяющихся пикселей.

По сравнению с базовым алгоритмом КДС:

— трудоемкость адаптивного алгоритма не изменилась — сохранилась линейная зависимость от количества пикселей в изображении;

— не затронута структура базового формата сжатия КДС для BMP изображений, т.е. не требуется специальный алгоритм декодирования.

Таким образом, изображения, сжатые предложенным алгоритмом, могут быть декодированы любым стандартным приложением, предназначенным для декодирования и просмотра изображений формата MS Windows® BMP.

В процессе экспериментальных исследований предложенного алгоритма сжатия без потерь для BMP изображений вычислялся коэффициент степени сжатия изображения

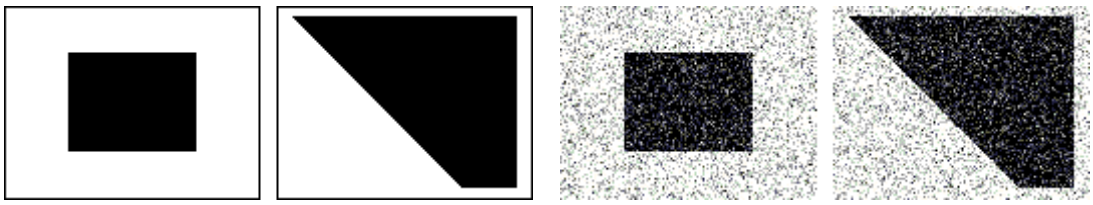
$$K = \frac{S_c}{S_0},$$

где S_0 — объем исходного несжатого изображения, байт;

S_c — объем изображения, сжатого базовым либо предложенным алгоритмами, байт.

Объемы исходных и сжатых BMP изображений вычислялись без учета служебных данных — заголовка изображения и палитры цветов, т.к. они остаются неизменными в процессе сжатия алгоритмом КДС.

Тестовые изображения представляли собой черный квадрат на белом фоне и черный усеченный треугольник на белом фоне размером 4096 x 3072 пикселей (рис. 1, *а*), сжимаемые алгоритмом КДС с высокой степенью сжатия. После наложения гауссовского белого шума со среднеквадратичным отклонением, равным 0,1 (рис. 1, *б*), изображения сжимаются алгоритмом КДС слабо, возможно даже с “отрицательным сжатием”.



а *б*

Рис. 1. Тестовые изображения без шума (а) и с гауссовским белым шумом (б)

Реализация предложенного адаптивного алгоритма КДС сравнивалась с реализациями алгоритма КДС GIMP версии 2.6.7 [4] и Adobe Photoshop версии CS3 [5] для сжатия без потерь BMP изображений в распространенных приложениях для обработки изображений. Эти реализации алгоритмов различаются методами выбора способов кодирования для исходных последовательностей значений пикселей в изображении. Полученные значения S_c и K приведены в таблицах 1, 2, соответственно.

Таблица 1

Объемы тестовых изображений, сжатых различными реализациями алгоритма КДС

Реализация алгоритма КДС	Объем сжатых тестовых изображений, S_c, байт			
	Черный квадрат		Черный треугольник	
	без шума	с шумом	без шума	с шумом
GIMP 2.6.7 RLE	116 740	12 817 702	116 322	12 820 466
Photoshop CS3 RLE	122 890	12 351 582	116 410	12 352 426
Предложенный алгоритм	46 118	12 261 802	91 736	12 261 640

Таблица 2

Коэффициенты степени сжатия изображений различными реализациями алгоритма КДС

Реализация алгоритма КДС	Коэффициент степени сжатия тестовых изображений, K			
	Черный квадрат		Черный треугольник	
	без шума	с шумом	без шума	с шумом
GIMP 2.6.7 RLE	0,009	1,019	0,009	1,019
Photoshop CS3 RLE	0,010	0,982	0,009	0,982
Предложенный алгоритм	0,004	0,974	0,007	0,974

С использованием предложенного алгоритма тестовые изображения сжимаются лучше, объемы S_c всех сжатых тестовых изображений меньше, коэффициент K имеет наименьшее значение для всех тестовых изображений. Тестовые изображения с шумом были сжаты с “отрицательным сжатием” K>1 только алгоритмом сжатия, реализованным в GIMP, т.е. объем сжатого тестового изображения с шумом больше, чем объем исходного.

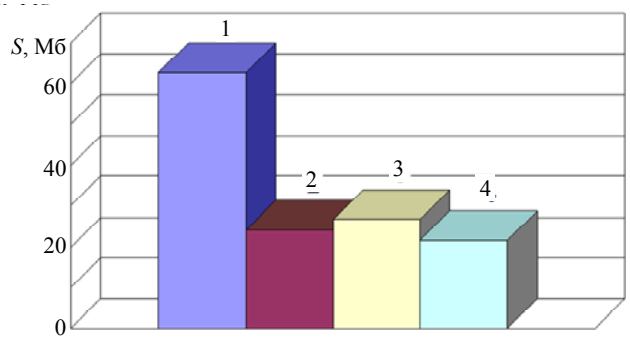


Рис. 2. Объемы реальных изображений, несжатых (1) и сжатых различными реализациями алгоритма КДС: GIMP 2.6.7 (2); Adobe Photoshop CS3 (3); предложенным адаптивным алгоритмом (4)

Разработанный алгоритм КДС был применен для сжатия без потерь реальных BMP изображений ресурсов ОС Windows. Количество изображений — 3187; средний объем изображения — 0,02 Мб; общий объем изображений S_0 — 62,64 Мб (рис. 2).

Полученный график плотности распределения вероятности P выигрыша E в сжатии BMP изображений (рис. 3) показал, что предложенный адаптивный алгоритм КДС превосходит по степени сжатия алгоритмы КДС для сжатия без потерь изображений формата BMP, реализованные в приложениях GIMP и Adobe Photoshop, в среднем на 14…17 %. В некоторых случаях выигрыш в сжатии E превышал 400%, при этом ни одно из BMP изображений не было сжато предложенным алгоритмом КДС с меньшей степенью сжатия, чем алгоритмами КДС, реализованными в GIMP и Adobe Photoshop.

Предложенный адаптированный к свойствам изображений алгоритм КДС сжатия без потерь может использоваться для сжатия изображений формата MS Windows® BMP до 256 цветов либо градаций серого с 4-мя или 8-ю битами на пиксель, допускающих использование базовых алгоритмов КДС. Также предложенный алгоритм может быть использован для сжатия изображений в ресурсах операционной системы MS Windows®. Такие изображения, встроенные в исполняемые модули DLL, EXE и др., хранятся в формате BMP, обычно с 4-мя или 8-ю битами на пиксель.

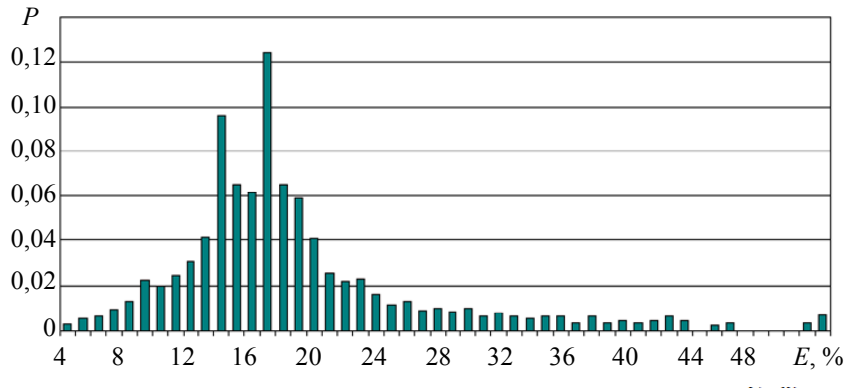


Рис. 3. Плотность распределения вероятности выигрыша в сжатии BMP изображений предложенным адаптивным алгоритмом КДС

Наконец, существует формат сжатия видео MS RLE video compres-

sion для ОС MS Windows®, использующий алгоритм КДС для сжатия последовательности кадров видео [6]. Для сжатия такого видео также можно использовать предложенный алгоритм.

Литература

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. — М.: Техносфера, 2005. — 1072 с.
2. Миано, Дж. Форматы и алгоритмы сжатия изображений в действии / Дж. Миано. — М.: Триумф, 2003. — 336 с.
3. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. — М.: ДИАЛОГ–МИФИ, 2003. — 384 с.
4. Официальный сайт приложения GIMP версии 2.6.7. — [Cited: June 12, 2010]. — Available from: http://www.gimp.org/.
5. Официальный сайт приложения Adobe Photoshop версии CS3. — [Cited: June 12, 2010]. — Available from: http://www.adobe.com/photoshop/.
6. Video for Windows — Wikipedia, the free encyclopedia. — [Cited: June 12, 2010]. — Available from: http://en.wikipedia.org/wiki/Video_for_Windows. — 12.06.2010.

Рецензент .д-р техн. наук, проф. Одес. нац. политехн. ун-та Крылов В.Н.

Поступила в редакцию 8 декабря 2009 г.