УДК 004.627: 004.932.2

Д.В. Дружинин

КОМБИНИРОВАННЫЙ АЛГОРИТМ СЖАТИЯ КЛЮЧЕВЫХ КАДРОВ ЭКРАННОГО ВИДЕО

Представлен комбинированный алгоритм сжатия, разработанный для изображений, являющихся кадрами экранного видео. Приведены результаты практического сравнения с алгоритмами семейства LZO (Lempel – Ziv – Oberhumer).

Ключевые слова: *экранное видео*, *сжатие изображений*, *быстрые алгоритмы сжатия.*

Экранное видео – видео происходящего на экране пользователя. При этом фиксируется активность пользователя: движения курсора мыши, скроллинг, сворачивание и открытие свёрнутого окна, перемещение окна, ввод текста и т. д.

Сжатие видео – одна из наиболее трудоёмких по времени задач, которые приходится решать не только профессионалам, но и рядовым пользователям. Одним из типов видеоданных, сжатие которых необходимо осуществлять в режиме реального времени, является экранное видео. Как правило, это видео высокого разрешения. Причём зачастую о полезности программ, осуществляющих сжатие и запись на жёсткий диск этого типа видеоданных, можно говорить только в том случае, когда их можно запустить в фоновом режиме.

Поскольку при сжатии видео ключевые кадры кодируются независимо от других кадров, существует необходимость в быстрых алгоритмах сжатия изображений. В соответствии с классификацией изображений, приведённой в [1], кадры экранного видео относятся к дискретно-тоновым изображениям. Для сжатия таких изображений, как правило, используются алгоритмы без потерь информации, так как при сжатии дискретно-тоновых изображений даже небольшой процент потерь может привести к значительному визуальному ухудшению качества изображения. Например, искажение всего нескольких пикселей буквы делает её неразборчивой, преобразует привычное начертание в практически неразличимое [1, с. 120]. К таким алгоритмам можно отнести RLE (Run-length encoding) и семейство алгоритмов LZ (Lempel – Ziv). Для сжатия экранного видео в режиме реального времени зачастую используется алгоритм LZO [2]. Например, LZO используется в свободном ПО (программном обеспечении) для сжатия экранного видео CamStudio [3].

В этой работе представлена вторая версия гибридного алгоритма, предназначенного для сжатия дискретно-тоновых изображений. Гибридный алгоритм первой версии был также разработан автором. Ранние модификации гибридного алгоритма подробно рассмотрены в [4, 5]. В этих работах также проводится практическое сравнение гибридного алгоритма с некоторыми другими алгоритмами сжатия. Вторая версия алгоритма отличается увеличенной скоростью работы и повышенной степенью сжатия. Далее в тексте при упоминании о гибридном алгоритме речь идёт именно о второй версии.

Также в работе представлен комбинированный алгоритм сжатия на основе гибридного алгоритма, который на финальном этапе сжатия использует LZO. Как

показано в разд. 3 «Результаты тестирования», такой комбинированный алгоритм позволяет значительно увеличить степень сжатия по сравнению с алгоритмами семейства LZO, способными сжимать экранное видео в режиме реального времени на широком спектре компьютеров пользователей.

1. Схема комбинированного алгоритма сжатия

Был предложен следующий комбинированный алгоритм сжатия:

На первом этапе исходное изображение обрабатывается гибридным алгоритмом версии 2. На выходе получается 3 массива:

1. Массив флагов. В этом массиве избыточность данных минимальна, поэтому флаги записываются в выходной поток без дополнительного сжатия.

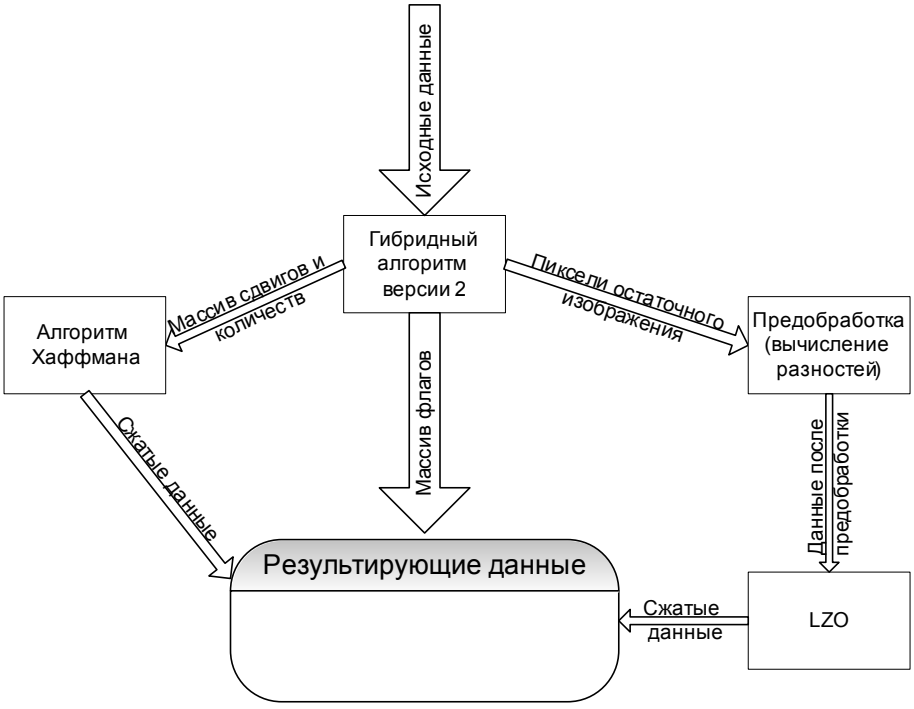

Рис. 1. Схема комбинированного алгоритма, разработанного для сжатия ключевых кадров экранного видео

2. Массив байтов сдвигов и количеств. В этом массиве также содержатся различные служебные данные (кроме флагов), используемые гибридным алгоритмом. Все данные, попадающие в этот массив, имеют однобайтовую природу. Данные в этом массиве имеют большую избыточность, по сравнению с массивом флагов. Действительно, для кадра экранного видео оказывается, что различные значения таких параметров, как количество подряд идущих пикселей одного цвета, а также расстояние до ближайшего встреченного пикселя такого же цвета, не являются равновероятными. То есть такие данные имеют статистическую избыточность. Поэтому для дополнительного сжатия этих данных можно использовать один из статистических методов сжатия. Было принято решение использовать ал-

горитм Хаффмана, так как кодирование метода Хаффмана работает достаточно быстро для осуществления сжатия в режиме реального времени.

3. Массив пикселей. Будем называть этот массив остаточным изображением. Это те пиксели исходного изображения, которые не были заменены в ходе кодирования гибридным алгоритмом некоторыми служебными данными. Как правило, статистическая избыточность в этом массиве невысока (например, алгоритм Хаффмана обычно способен сжать остаточное изображение всего на 1 – 3 %). Зато в этих данных присутствует некоторая пространственная избыточность. Часто в остаточном изображении можно увидеть группы подряд идущих пикселей по вертикали или горизонтали близких цветов. Такой тип пространственной избыточности характерен для остаточного изображения, так как гибридный алгоритм не может его устранить.

Было принято решение провести следующую предобработку: новое значение для каждой компоненты пикселя, начиная со второго, вычисляется как разность с соответствующей компонентой предыдущего пикселя:

$$c\_coded[i] = (c[i] - c[i-1]) \% 256; \text{ (язык Си++)}.$$

После этой предобработки вместо различных последовательностей пикселей близкого цвета появляются значения, близкие к нулю. И только первый пиксель в группе пикселей близкого цвета после такой предобработки получает значение, далёкое от нуля.

Затем применяется алгоритм LZO, который обеспечивает дополнительное сжатие при минимальных временных затратах. Было принято решение использовать алгоритм LZO_X_999 с уровнем сжатия 6 как обеспечивающий максимальное сжатие при допустимых затратах времени. Как будет показано в разд. 3 «Результаты тестирования» степень сжатия предложенной схемы сжатия не ниже, а для некоторых типов изображений значительно выше, чем у LZO_X_999 с уровнем сжатия 6.

2. Гибридный алгоритм второй версии

Гибридный алгоритм является органическим соединением двух алгоритмов: RLE и сдвигового. Рассмотрим вначале каждый из этих алгоритмов в отдельности.

2.1. RLE

В качестве составной части гибридного алгоритма используется специально разработанная реализация RLE, адаптированная для сжатия экранного видео. Канонический алгоритм RLE способен выявлять только горизонтальную или вертикальную избыточность (в зависимости от способа обхода пикселей изображения) [6, с. 289]. В состав гибридного алгоритма входит реализация RLE, способная выявлять как горизонтальную, так и вертикальную избыточность.

Вспомогательные структуры данных. При кодировании и декодировании используется массив флагов, где один флаг соответствует одному пикселю исходного изображения. Если этот флаг равен 0, соответствующий ему пиксель ещё не был закодирован, иначе – данный пиксель уже закодирован. Уже закодированные пиксели пропускаются при кодировании.

На каждом шаге алгоритма происходит подсчёт количества подряд идущих одинаковых пикселей в трёх направлениях: вправо от текущего пикселя, вниз от текущего пикселя, а также в прямоугольнике, левым верхним углом которого яв-

ляется текущий пиксель. Затем для кодирования выбирается то направление, в котором было найдено максимальное количество одинаковых пикселей. Для большинства типов изображений, например для непрерывно-тоновых, не имеет смысла вводить такое направление поиска, как прямоугольник с текущим пикселем в качестве левого верхнего угла. Но в случае кадров экранного видео использование такой области поиска оправдано, так как часто встречаются именно прямоугольные области одного цвета.

Обход пикселей изображения выполняется построчно сверху вниз, но если на данном шаге выбрана вертикальная либо прямоугольная группа пикселей, то устанавливаются соответствующие флаги в массиве флагов, после этого происходит возврат на исходную строку. При этом возможна следующая ситуация: пиксель p[i] уже был закодирован ранее, попав в вертикальную или прямоугольную группу пикселей одинакового цвета. Пусть текущий пиксель – это $p[i-1]$. Допустим также, что пиксель $p[i+1]$ имеет такой же цвет, как $p[i-1]$. В этом случае $p[i-1]$ и $p[i+1]$ могут быть закодированы, как горизонтальная группа пикселей.

Рассмотрим способ выбора флагов, записываемых в выходной массив:

1. В случае одиночного пикселя в массив флагов записывается один бит, равный 0.

2. В случае выбора прямоугольной области в массив флагов записывается последовательность битов 11.

3. В случае выбора горизонтальной области в массив флагов записывается последовательность битов 100.

4. В случае выбора вертикальной области в массив флагов записывается последовательность битов 101.

Самый короткий однобитовый флаг соответствует одиночному пикселю, так как тестирование показало, что при кодировании кадров экранного видео частота, с которой встречается одиночный пиксель, значительно превышает сумму частот всех остальных случаев. Частота, с которой встречается прямоугольная область, оказалась несколько выше частот встречаемости горизонтальной и вертикальной областей, поэтому для прямоугольной области была выбрана более короткая последовательность флагов.

Используется следующий формат закодированных данных.

В первом случае: *<Последовательность флаговых битов> <цвет пикселя>*.

Во втором случае: *<Последовательность флаговых битов> <цвет пикселя> <количество пикселей по горизонтали> <количество пикселей по вертикали>*.

В третьем и четвёртом случаях: *<Последовательность флаговых битов> <цвет пикселя> <количество пикселей>*.

При этом под количество пикселей отводится 2 байта во втором случае; 1 байт в третьем и четвёртом случаях.

Коэффициент сжатия в наихудшем случае: 25 / 24.

Коэффициент сжатия в наилучшем случае: $42 / (256 \cdot 256 \cdot 24)$.

Преимущества:

1. У такой реализации RLE повышена способность выявлять пространственную избыточность по сравнению с канонической реализацией.

Но при этом сохраняется главный недостаток канонического алгоритма RLE при сжатии экранного видео:

1. Если при построчном обходе пикселей изображения часто чередуются цвета, причём даже в том случае, когда набор цветов ограничен (например, текст), эффективность сжатия резко падает.

2.2. Сдвиговый алгоритм

Идея алгоритма: если при построчном обходе пикселей изображения незадолго до текущего пикселя встречался пиксель такого же цвета, то 3 байта, кодирующие цвет пикселя, можно заменить на 1-байтовую ссылку на пиксель с таким же цветом, а точнее – указать, на сколько пикселей нужно сдвинуться назад относительно текущего пикселя остаточного изображения, чтобы получить нужный цвет. Таким образом, может быть выстроено множество списков. Обход пикселей изображения выполняется построчно сверху вниз.

Вспомогательные структуры данных. Для ускорения работы алгоритма, как при кодировании, так и при декодировании используется хэш-таблица. При кодировании ключом хэш-таблицы является цвет, а значением – номер пикселя в остаточном изображении последнего просмотренного пикселя с таким цветом. При декодировании ключом хэш-таблицы является номер последнего просмотренного пикселя с таким цветом в остаточном изображении, а значением – цвет.

В случае, когда цвет встретился впервые или количество пикселей в остаточном изображении до предыдущего пикселя с таким же цветом превышает $2^8 - 1$, используется следующий формат: <0 (один бит)> <цвет пикселя>. Иначе используется формат: <1 (один бит)> <ссылка>.

Коэффициент сжатия в наихудшем случае: 25 / 24.

Коэффициент сжатия в наилучшем случае: 9 / 24.

Преимущества:

1) получаемый при кодировании формат хорошо поддаётся дальнейшему кодированию другими алгоритмами.

Недостатки:

1) даже в наилучшем случае степень сжатия невелика;

2) алгоритм медленно работает из-за обращения на чтение, а затем на запись к дополнительным структурам данных для каждого пикселя.

2.3. Гибридный алгоритм

На каждом шаге алгоритма кодирование выполняется в 2 стадии:

1. Выполняется часть алгоритма, основанная на сдвиговом алгоритме, то есть определяется, можно ли заменить 3 байта цвета пикселя на 1 байт ссылки.

2. Выполняется часть алгоритма, основанная на RLE. На этой стадии определяется, можно ли выявить группу пикселей одного цвета. Не будем перечислять варианты результирующего формата, так как это все возможные комбинации результирующих форматов RLE и сдвигового алгоритмов.

При этом используются вспомогательные структуры данных сдвигового алгоритма и RLE, рассмотренные выше.

Коэффициент сжатия в наихудшем случае: 26 / 24.

Коэффициент сжатия в наилучшем случае: $27 / (256 \cdot 256 \cdot 24)$.

Гибридный алгоритм обладает всеми преимуществами RLE. При его использовании достигается дополнительное сжатие за счёт применения идей сдвигового алгоритма. Гибридный алгоритм не наследует недостатки сдвигового алгоритма. Недостаток (1) устраняется за счёт сжатия алгоритмом RLE. А недостаток (2) присутствует в значительно меньшей степени, так как обращение к дополнительным структурам данных происходит не для каждого пикселя, а для каждой группы, то есть примерно в 40 раз реже при сжатии изображения, типичного для Windows XP и примерно в 86 раз реже при сжатии изображения, значительную

часть которого занимает текст (эти цифры получены опытным путём). За счёт применения идей сдвигового алгоритма также устранён недостаток алгоритма RLE – в случае частого чередования цветов при построчном обходе пикселей изображения в условиях ограниченного количества этих цветов степень сжатия значительно выше, чем при сжатии RLE [4].

Стоит отметить, что гибридный алгоритм плохо поддаётся распараллеливанию, так как в случае независимой обработки различных частей исходного изображения разными потоками будет утеряна информация о связях между этими частями, вследствие чего степень сжатия снизится.

3. Результаты тестирования

Были протестированы следующие алгоритмы: LZO_X_1, LZO_X_999 с уровнем сжатия 1, 4, 6, 9; гибридный алгоритм второй версии, представленный комбинированный алгоритм, использующий LZO_X_999 с уровнем сжатия 6. Также в тестировании принимала участие реализация алгоритма, соответствующего стандарту Deflate [6, с. 95], от Microsoft. Соответствующий класс реализован в .NET framework 2.0. Для алгоритмов семейства LZO приведены также время финального сжатия алгоритмом Хаффмана, а также размер закодированного изображения после финального сжатия.

При тестировании каждое изображение имело разрешение 1024×768 и глубину цвета в 32 бита. Таким образом, размер исходного изображения составляет 1024·768·4 байтов. Тестирование проводилось на платформе со следующими характеристиками: процессор Intel Core 2 Duo E6750 2,66 ГГц; оперативная память DDR2 2Гб; операционная система Windows XP. На момент написания данной работы тестовая платформа находится в среднем сегменте по производительности.

Замечание 1: для гибридного алгоритма второй версии данные о размере сжатых данных и времени сжатия приведены с учётом финальной обработки методом Хаффмана, так как гибридный алгоритм был изначально спроектирован для использования совместно с методом Хаффмана.

Для тестирования использовались скриншоты трёх типов:

1. Изображения, типичные для Windows XP (10 штук);
2. Изображения, значительную часть которых занимает текст (8 штук);
3. Изображения, содержащие графики, диаграммы (10 штук);

Эти скриншоты доступны по ссылке [7]. Рис. 2 – 4 представляют собой уменьшенные копии некоторых тестовых изображений, сохранённые в градациях серого цвета.

По результатам тестирования, представленным в табл. 1 – 3, видно, что финальное сжатие методом Хаффмана позволяет значительно увеличить коэффициент сжатия алгоритмов семейства LZO при сравнительно небольших затратах времени (порядка 7 – 10 мс на кодирование и столько же на декодирование). Алгоритм, соответствующий стандарту Deflate и LZO_X_1, продемонстрировал наихудшие результаты на большинстве тестов. Из серии алгоритмов LZO наибольший коэффициент сжатия имеет LZO_X_999 с уровнем сжатия 9. Но этот алгоритм выполняется слишком долго для его запуска в режиме реального времени на значительной части компьютеров, используемых пользователями. При тестировании было установлено, что из алгоритмов серии LZO максимальная степень сжатия, при условии возможности запуска алгоритма в режиме реального времени на подавляющей части компьютеров, используемых пользователями, достигается при использовании LZO_X_999 с уровнем сжатия 6.

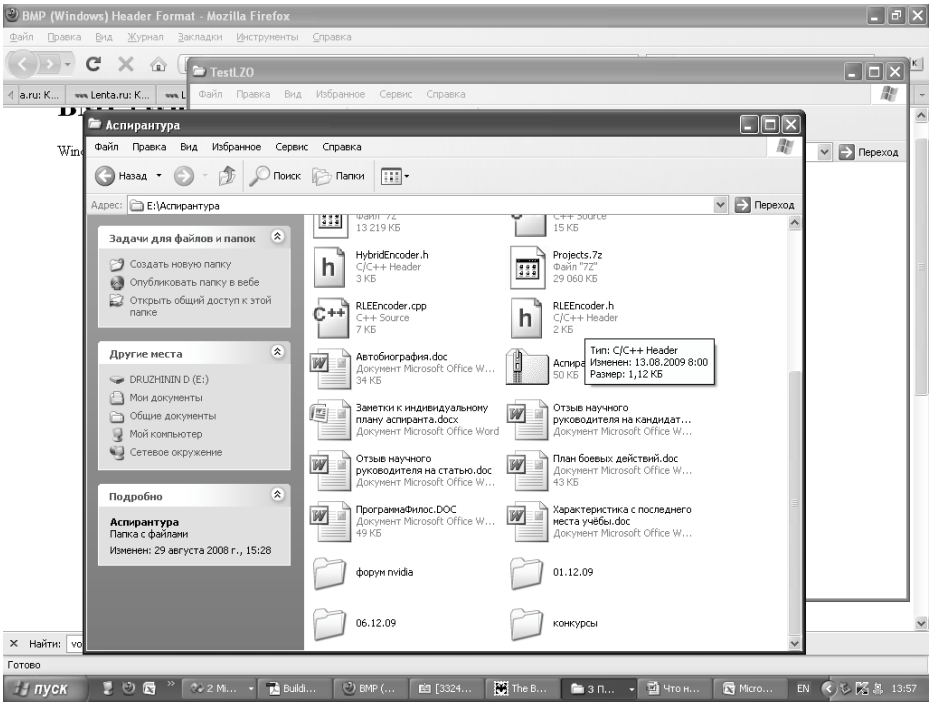

Рис. 2. Уменьшенная копия изображения WinXP_0.bmp

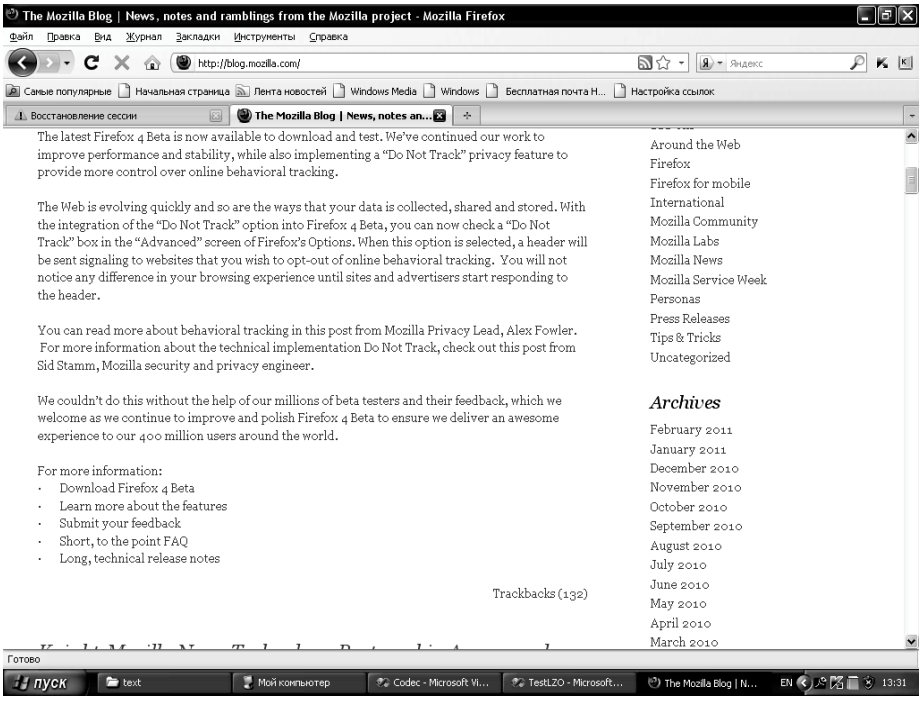

Рис. 3. Уменьшенная копия изображения text_4.bmp

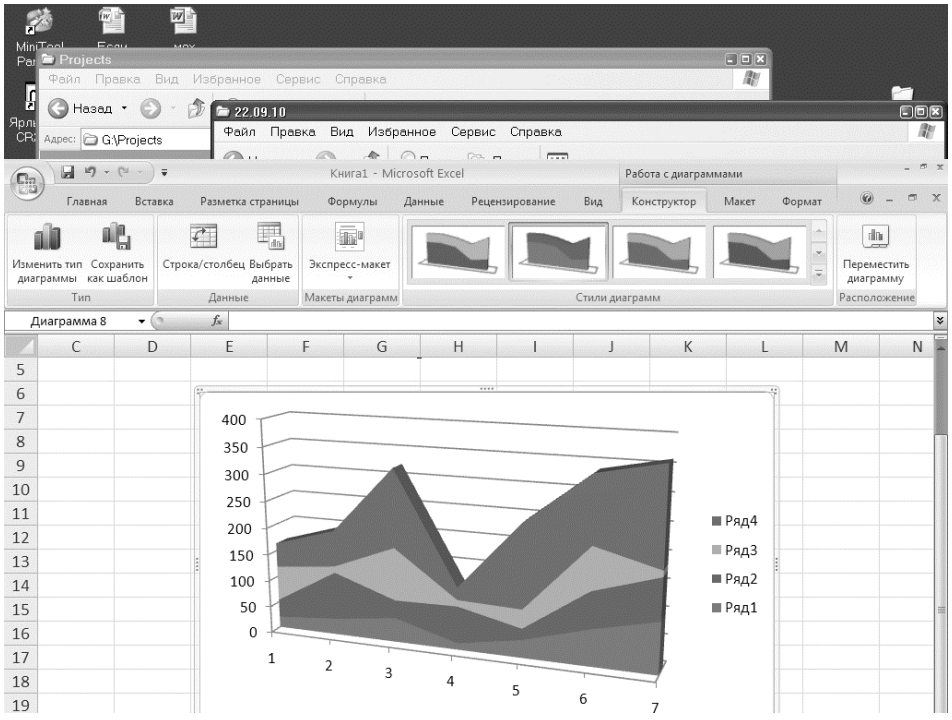

Рис. 4. Уменьшенная копия изображения diagram_0.bmp

Сам по себе гибридный алгоритм демонстрирует меньшую степень сжатия по сравнению со схемой сжатия на его основе. Рассмотрим более подробно результаты тестирования комбинированного алгоритма, использующего LZO_X_999 с уровнем сжатия 6, и LZO_X_999 с уровнем сжатия 9 (с использованием финального сжатия методом Хаффмана).

Таблица 1

Усреднённые данные о сжатии изображений, типичных для Windows XP

Алгоритм / Параметр	Время кодирования / декодирования (мс)	Размер после основного сжатия	Размер после сжатия алгоритмом Хаффмана
1. LZO_X_999, уровень 1	56 / 11	147886,6	124405
2. LZO_X_999, уровень 4	56, 6 / 10	129921,8	112427,4
3. LZO_X_999, уровень 6	67,7 / 9	120198,5	106388,6
4. LZO_X_999, уровень 9	370,6 / 8,8	113822,4	101859,9
5. LZO_X_1	11,2 / 11,7	159182,3	142159
6. Гибридный алгоритм v. 2	54,2 / 33,5	117388	
7. Комбинированный алгоритм, использующий LZO_X_999, уровень 6	58,1 / 24,6	91029,1	
8. Комбинированный алгоритм, использующий LZO_X_999, уровень 9	64,1 / 23,1	90679	
9. Deflate	48,6 / 12	157206,5	

Таблица 2

Усреднённые данные о сжатии изображений, значительную часть которых занимает текст

Алгоритм / Параметр	Время кодирования / декодирования (мс)	Размер после основного сжатия	Размер после сжатия алгоритмом Хаффмана
1. LZO_X_999, уровень 1	56,1 / 10,9	164025,4	118184,1
2. LZO_X_999, уровень 4	57,4 / 9,6	133731,6	102429,1
3. LZO_X_999, уровень 6	72,6 / 8,5	110345,4	91106,8
4. LZO_X_999, уровень 9	880 / 7,8	94430,6	81288,4
5. LZO_X_1	11,8 / 11	176658,4	138447,3
6. Гибридный алгоритм V. 2	62,4 / 33	98180,3	
7. Комбинированный алгоритм, использующий LZO_X_999, уровень 6	62,5 / 26,7	89682,6	
8. Комбинированный алгоритм, использующий LZO_X_999, уровень 9	65 / 26,8	89578	
9. Deflate	50,6 / 10,9	126848,4	

Таблица 3

Усреднённые данные о сжатии изображений, содержащих графики и диаграммы

Алгоритм / Параметр	Время кодирования / декодирования (мс)	Размер после основного сжатия	Размер после сжатия алгоритмом Хаффмана
1. LZO_X_999, уровень 1	53,9 / 9,1	124239,5	106542,4
2. LZO_X_999, уровень 4	55,4 / 8,7	113404,2	98963,6
3. LZO_X_999, уровень 6	65,4 / 8,6	107833,9	95126,4
4. LZO_X_999, уровень 9	293,5 / 7,6	103002,1	91764,1
5. LZO_X_1	10,1 / 10,3	136494,8	124248,7
6. Гибридный алгоритм V. 2	60,2 / 32	112324,1	
7. Комбинированный алгоритм, использующий LZO_X_999, уровень 6	62,5 / 27,6	94742,1	
8. Комбинированный алгоритм, использующий LZO_X_999, уровень 9	68,2 / 26,6	94522,5	
9. Deflate	47,1 / 10,9	145175,6	

По данным, представленным в табл. 4, видно, что комбинированный алгоритм имеет превосходство по степени сжатия на всех изображениях, типичных для Windows XP (в среднем на 17 %). На изображениях, значительную часть которых занимает текст, оба алгоритма демонстрируют близкие результаты (в среднем различие около 1,5 %), что подтверждается критерием знаков и критерием Вилкоксона [8]. Так, количество типичных и нетипичных сдвигов равны. А суммы рангов в соответствии с критерием Вилкоксона в типичном и нетипичном направлении равны соответственно 30 и 25 (очень близки). На изображениях, содержащих графики и диаграммы, оба алгоритма также демонстрируют близкие результаты (в среднем различие менее чем на 1 %). Количество сдвигов в типичном направлениях 4,5, в нетипичном направлении – 3,5. Сумма рангов в соответствии с критерием Вилкоксона в типичном направлении равна 22, в нетипичном случае – 14.

Таблица 4

Данные о размере закодированного файла (в байтах) для 28 тестовых изображений

Название исходного файла	Комбинированный алгоритм	LZO_X_999 уровень 6
WinXP_0.bmp	113975	128252
WinXP_1.bmp	82631	96198
WinXP_2.bmp	52561	60342
WinXP_3.bmp	74150	81067
WinXP_4.bmp	156418	180901
WinXP_5.bmp	109818	136935
WinXP_6.bmp	75795	94816
WinXP_7.bmp	78706	84810
WinXP_8.bmp	76326	93963
WinXP_9.bmp	89911	106602
text_0.bmp	102958	101112
text_1.bmp	106795	109794
text_2.bmp	63057	69093
text_3.bmp	98198	98203

Название исходного файла	Комбинированный алгоритм	LZO_X_999 уровень 9
diagram_0.bmp	94224	94236
diagram_1.bmp	103741	101878
diagram_2.bmp	106732	98256
diagram_3.bmp	87839	91097
diagram_4.bmp	106887	106721
diagram_5.bmp	90874	87176
diagram_6.bmp	153883	159781
diagram_7.bmp	79009	77775
diagram_8.bmp	66592	70697
diagram_9.bmp	57640	63647
text_4.bmp	84089	79868
text_5.bmp	85699	96272
text_6.bmp	91144	88090
text_7.bmp	85521	86422

Замечание: для критерия знаков и критерия Вилкоксона в качестве типичного направления сдвига был выбран случай, когда степень сжатия комбинированного алгоритма выше.

Заключение

Представленный в данной работе комбинированный алгоритм, основанный на гибридном алгоритме и LZO, подтвердил свою эффективность при тестировании. Такой комбинированный алгоритм позволяет увеличить степень сжатия изображений, типичных для Windows XP, в среднем на 17 % по сравнению с лучшими по степени сжатия представителями семейства алгоритмов LZO, способными сжимать экранное видео в режиме реального времени на широком спектре компьютеров пользователей (LZO_X_999 с уровнем сжатия 6). При этом комбинированный алгоритм и LZO_X_999 с уровнем сжатия 6 обеспечивают близкие степени сжатия изображений, содержащих текст, диаграммы или графики. Поэтому представленный комбинированный алгоритм может быть использован на практике для сжатия кадров экранного видео.

На данный момент комбинированный алгоритм встроен в кодек для обработки экранного видео Butterfly Screen Video Codec, ориентированный на минимизацию использования процессорного времени при сохранении высокой степени сжатия. Представленный комбинированный алгоритм используется не только при сжатии ключевых кадров, но и при сжатии изменившихся частей промежуточных кадров. Поэтому разработка представленного в данной работе комбинированного алгоритм является очередным шагом в оптимизации по уровню использования системных ресурсов и степени сжатия кодека Butterfly Screen Video Codec.

ЛИТЕРАТУРА

1. *Сэломон Д.* Сжатие данных, изображений и звука. М.: Техносфера, 2006. 365 с. (Мир программирования).
2. LZO. [Электронный ресурс]. URL: http://www.oberhumer.com/opensource/lzo (дата обращения 19.02.2011)

3. *Camstudio*. [Электронный ресурс]. URL: http://camstudio.org (дата обращения 19.02.2011)
4. *Дружинин Д. В.* Гибридный алгоритм сжатия изображения. Сравнение алгоритмов сжатия изображений. // Информационные технологии и математическое моделирование: Материалы VI Международной научно-практической конференции. Томск, 2007. Т. 2. С. 70–73.
5. *Дружинин Д.В.* Модификации гибридного алгоритма сжатия изображений // IV Научно-практическая конференция «Обратные задачи и информационные технологии рационального природопользования». Ханты-Мансийск, 2008. С. 218–222.
6. *Ватолин Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: Диалог-МИФИ, 2003. 384 с.
7. *Скриншоты*. [Электронный ресурс] / Д.В. Дружинин. URL: http://narod.ru/disk/15021751001/ screenshots.zip.html (дата обращения 19.02.2011).
8. *Большев Л.Н.*, *Смирнов Н.В.* Таблицы математической статистики. М.: Наука, 1983. 416 с.

Дружинин Денис Вячеславович
Томский государственный университет,
E-mail: dendru@rambler.ru

Поступила в редакцию 8 июня 2010 г.