ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.627: 004.932.2

Д.В. Дружинин

КОМБИНИРОВАННЫЕ АЛГОРИТМЫ СЖАТИЯ КЛЮЧЕВЫХ КАДРОВ ЭКРАННОГО ВИДЕО

Представлен комбинированный алгоритм сжатия, разработанный для изображений, являющихся кадрами экранного видео. Этот алгоритм на первом этапе использует гибридный алгоритм, ранее разработанный автором, а на втором этапе – библиотеку zlib. Приведены результаты практического сравнения с комбинированным алгоритмом, использующим на первом этапе гибридный алгоритм, а на втором – алгоритм LZO, а также некоторые другие алгоритмы сжатия.

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

Экранное видео – это видео происходящего на экране пользователя. Экранное видео возникает в результате фиксации активности 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. 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 (Lempel-Ziv-Oberhumer), так как позволяет быстро сжимать дискретно-тоновую графику, обеспечивая при этом приемлемую степень сжатия [2, 3]. Алгоритмы семейства zlib применяются для перекодирования экранного видео, так как обеспечивают высокую степень сжатия, хотя и требуют значительных временных затрат [4]. Гибридный алгоритм версии 2, предназначенный для сжатия дискретно-тоновых изображений, был

разработан автором и подробно описан в [5]. Также в [5] был представлен комбинированный алгоритм сжатия, использующий гибридный алгоритм на первом этапе и LZO на втором этапе.

1. Схема комбинированного алгоритма сжатия, использующего zlib

Был предложен следующий комбинированный алгоритм сжатия. На первом этапе исходное изображение обрабатывается гибридным алгоритмом версии 2. На выходе получается 3 массива:

1. Массив флагов.

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.

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.

На втором этапе все 3 массива, получаемых на выходе гибридного алгоритма, сжимаются по отдельности алгоритмом zlib с уровнем 9. Имеет смысл применять zlib к различным типам результирующих данных гибридного алгоритма по отдельности, так как каждый тип данных обладает специфическим типом избыточности. Оказалось, что zlib способен существенно сжать массив флагов (в среднем на 20 % для zlib уровня 9), в то время как применение LZO и алгоритма Хаффмана к этому типу данных практически не давало эффекта.

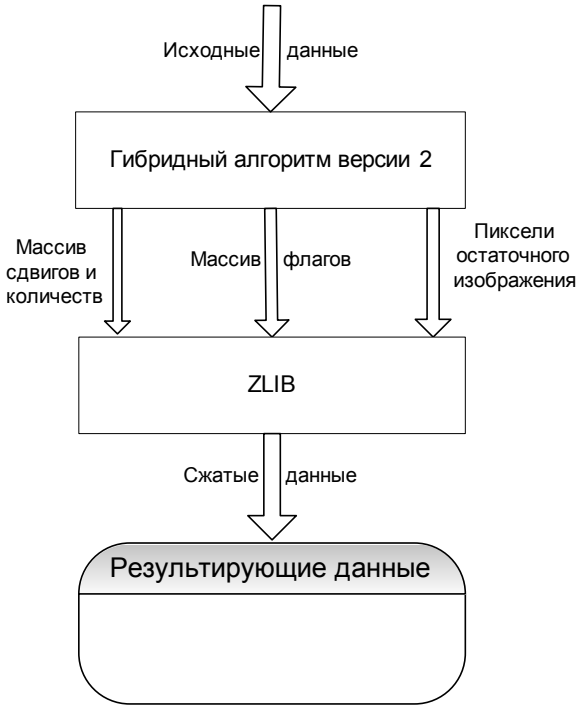

Рис. 1. Схема комбинированного алгоритма, использующего zlib

Замечание: Чем выше используемый уровень (от 1 до 9) алгоритма zlib, тем выше степень сжатия и временные затраты. Для LZO применяется тот же набор уровней.

2. Результаты тестирования

Были протестированы следующие алгоритмы: LZO_X_1, LZO_X_999 с уровнем сжатия 1, 4, 6, 9; zlib с уровнем сжатия 1, 4, 6, 9; гибридный алгоритм второй версии, комбинированный алгоритм, использующий LZO_X_999 с уровнем сжатия 6, 9; комбинированный алгоритм, использующий zlib с уровнем сжатия 6, 9; реализация FastAC арифметического сжатия [7]; реализация Main Concept JPEG 2000 [8]; реализации Microsoft алгоритмов JPEG, JPEG Lossless; реализация CharLS алгоритма JPEG_LS [9]. Также в тестировании принимала участие реализация алгоритма, соответствующего стандарту Deflate [6, с. 95], от Microsoft.

Для алгоритмов семейства LZO приведены также время финального сжатия алгоритмом Хаффмана, а также размер закодированного изображения после финального сжатия (табл. 1 – 3).

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	
10. zlib уровень 1	15 / 5	135744	
11. zlib уровень 4	32 / 4	110946,3	
12. zlib уровень 6	38 / 4	104116,6	
13. zlib уровень 9	89 / 4	99733,9	
14. Комбинированный алгоритм, использующий zlib уровень 6	56 / 22	80693,2	
15. Комбинированный алгоритм, использующий zlib уровень 9	57 / 22	80672,7	
16. Алгоритм Хаффмана	78 / 66	1389133,1	
17. Арифметическое сжатие	35 / 74	1378388,1	
18. JPEG 2000 (без потерь)	437 / 361	298352	
19. JPEG 2000 (с уровнем потерь 0,1)	44 / 32	229817	
20. JPEG Lossless	41 / 54	358629	
21. JPEG (с уровнем потерь 0,1)	34 / 43	171425	
22. RLE	4 / 4	342721	
23. JPEG-LS	25 / 23	210390	

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	
10. zlib уровень 1	15 / 5	122339,75	
11. zlib уровень 4	32 / 4	94714,5	
12. zlib уровень 6	43 / 4	83749,25	
13. zlib уровень 9	249 / 4	76921,5	
14. Комбинированный алгоритм, использующий zlib уровень 6	66 / 25	73976,5	
15. Комбинированный алгоритм, использующий zlib уровень 9	69 / 25	73857,4	
16. Алгоритм Хаффмана	52 / 42	880449,75	
17. Арифметическое сжатие	31 / 68	841289,5	
18. JPEG 2000 (без потерь)	481 / 390	384686	
19. JPEG 2000 (с уровнем потерь 0,1)	46 / 31	235785	
20. JPEG Lossless	38 / 50	475470	
21. JPEG (с уровнем потерь 0,1)	40 / 49	289965	
22. RLE	3 / 3	286454	
23. JPEG-LS	24 / 20	184841	

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	
10. zlib уровень 1	14 / 4	122351,1	

Окончание табл. 3

Алгоритм / Параметр	Время кодирования / декодирования (мс)	Размер после основного сжатия	Размер после сжатия алгоритмом Хаффмана
11. zlib уровень 4	32 / 4	101396,2	
12. zlib уровень 6	36 / 4	95471,7	
13. zlib уровень 9	77 / 4	90687,5	
14. Комбинированный алгоритм, использующий zlib уровень 6	62 / 27	82764,7	
15. Комбинированный алгоритм, использующий zlib уровень 9	62 / 25	82762,6	
16. Алгоритм Хаффмана	70 / 59	1232763,2	
17. Арифметическое сжатие	34 / 73	1216817,4	
18. JPEG 2000 (без потерь)	456 / 375	315728	
19. JPEG 2000 (с уровнем потерь 0,1)	45 / 32	235824	
20. JPEG Lossless	45 / 56	348097	
21. JPEG (с уровнем потерь 0,1)	34 / 43	163521	
22. RLE	4 / 4	316114	
23. JPEG-LS	27 / 24	239670	

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: для гибридного алгоритма второй версии данные о размере сжатого изображения и времени сжатия приведены с учётом финальной обработки методом Хаффмана. Для этого все результирующие данные гибридного алгоритма объединяются в один массив.

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 [10]. Рис. 2 – 4 представляют собой уменьшенные копии некоторых тестовых изображений. Комбинированный алгоритм, использующий zlib уровня 9, обеспечил наивысшую степень сжатия на всех трёх типах изображений при допустимых временных затратах. Комбинированный алгоритм, использующий zlib уровня 6, показал близкие результаты (чуть меньшую степень сжатия при незначительно меньшем времени сжатия). Сам по себе гибридный алгоритм демонстрирует меньшую степень сжатия по сравнению с комбинированным алгоритмом, построенном на его основе.

Сравним комбинированный алгоритм, использующий zlib уровня 9, с zlib уровня 9. Комбинированный алгоритм обеспечивает на 23,5 % более высокую степень сжатия для изображений, типичных для Windows XP, при близком времени выполнения. Преимущество комбинированного алгоритма при сжатии изображений, содержащих текст, составляет 4 %. При этом комбинированный алгоритм выполняется почти в 4 раза быстрее. При сжатии изображений, содержащих графики и диаграммы, комбинированный алгоритм демонстрирует на 9,5 % более высокую степень сжатия при близком времени выполнения.

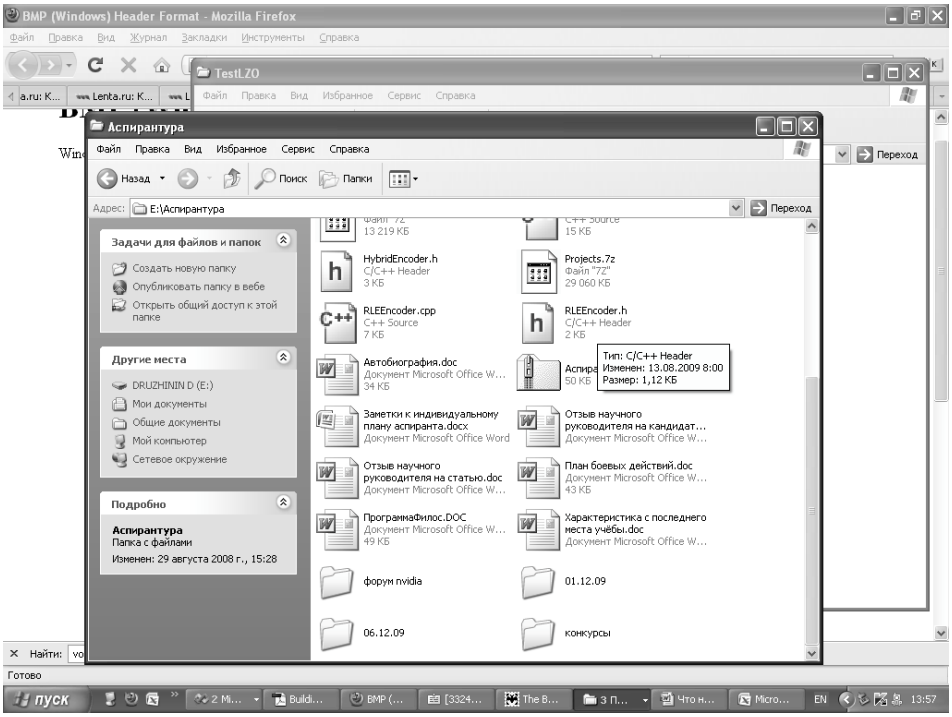

Рис. 2. Уменьшенная копия изображения WinXP_0.bmp

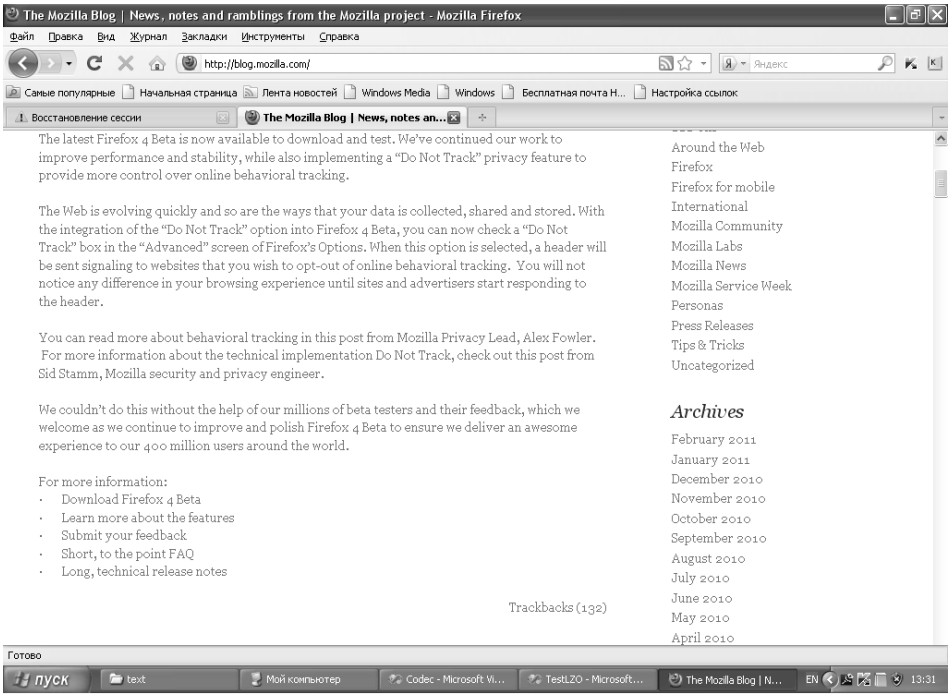

Рис. 3. Уменьшенная копия изображения text_4.bmp

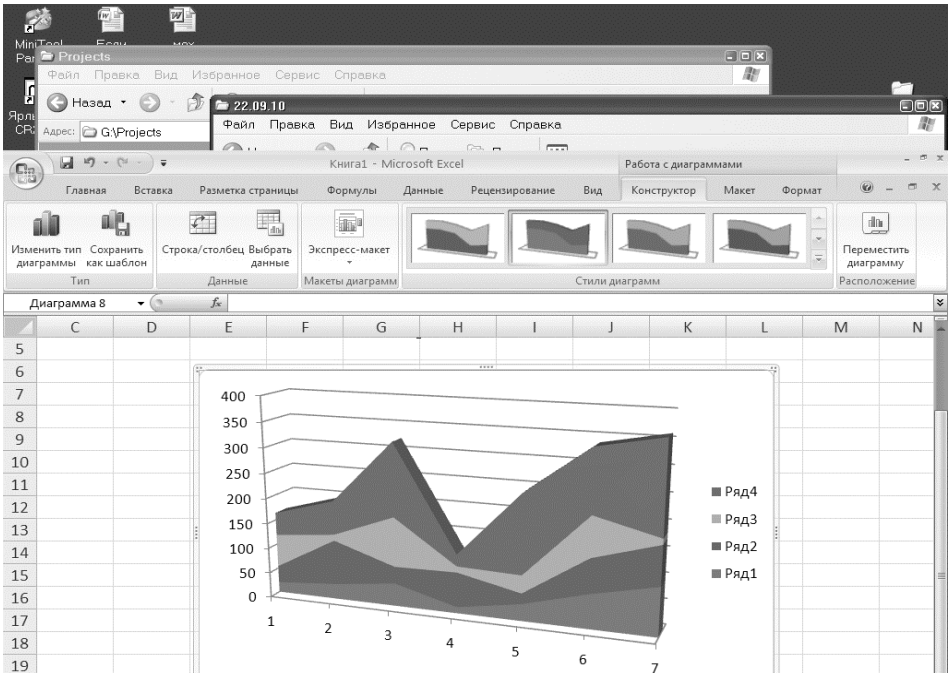

Рис. 4. Уменьшенная копия изображения diagram_0.bmp

Как видно по результатам тестирования, комбинированный алгоритм, использующий zlib уровня 9, превосходит по степени сжатия комбинированный алгоритм, использующий LZO и алгоритм Хаффмана. Алгоритмы JPEG-LS, JPEG 2000, JPEG показали степень сжатия значительно ниже, чем лидеры тестирования, так как эти алгоритмы ориентированы на сжатие фотографий (непрерывно-тоновой графики), а не дискретно-тоновой графики.

Заключение

Представленный в данной работе комбинированный алгоритм, основанный на гибридном алгоритме и zlib, подтвердил свою эффективность при тестировании. Такой комбинированный алгоритм позволяет значительно увеличить степень сжатия кадров экранного видео по сравнению с алгоритмами семейства LZO и zlib (на 23,5 % в случае изображений, типичных для Windows XP по сравнению с zlib с уровнем сжатия 9). При этом удалось увеличить скорость сжатия по сравнению с zlib с уровнем сжатия 9 до 4 раз (при сжатии изображений, содержащих текст). 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.
2. *LZO.* [Электронный ресурс]. URL: http://www.oberhumer.com/opensource/lzo.
3. *Camstudio.* [Электронный ресурс]. URL: http://camstudio.org.
4. *ZLIB.* [Электронный ресурс]. URL: http://zlib.net.
5. *Дружинин Д.В.* Комбинированный алгоритм сжатия ключевых кадров экранного видео. // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2011. № 3. С. 67–77.
6. *Ватолин Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. М.: Диалог-МИФИ, 2003. 384 с.
7. *FastAc.* [Электронный ресурс]. URL: http://www.cipr.rpi.edu/research/SPIHT/spiht3.html
8. *Main Concept.* [Электронный ресурс]. URL: http://www.mainconcept.com
9. *CharLS.* [Электронный ресурс]. URL: http://charls.codeplex.com
10. *Скриншоты.* [Электронный ресурс] / Д. В. Дружинин. URL: https://docs.google.com/file/d/0B_2xi7pVvd23RzAyNVJWWDJBSUU/edit?usp=sharing

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

Поступила в редакцию 7 мая 2013 г.

Druzhinin Denis V. (Tomsk State University). **Composite algorithms for screen video key frames compression.**

Keywords: screen video, image compression, fast compression algorithms.

Video compression is one of the most time-taking problems, which are solved not only by professionals, but also by ordinary users. Screen video is one of the video data types, which must be compressed in real-time mode. Often it is necessary to compress screen video in background mode at that. There is a necessity in fast image compression algorithms, because during video compression key frames are encoded independently. Screen video frames are referred to discrete-tone images. As a rule algorithms without information loss are used to compress such images, because while compressing such images even a small information loss percent can result in significant visual image degradation. RLE and family of LZ algorithms can be referred to such algorithms. LZO is actively used in screen video compression (for example, it is used in freeware screen video recorder CamStudio).

Composite algorithm for screen video key frames compression is introduced in this paper. This algorithm uses hybrid algorithm, developed by the author earlier, on the first step and zlib library on the second step. The paper also contains results of practical comparison with composite algorithm, which uses hybrid algorithm on the first step and LZO algorithm on the second step, and with several other algorithms. Three types of screenshots were used for testing:

1. Pictures, typical for Windows XP (10 pieces);
2. Pictures with text (8 pieces);
3. Pictures with graphics, diagrams (10 pieces).

Introduced in this paper composite algorithm, based on hybrid algorithm and zlib, confirmed its effectiveness. This composite algorithm demonstrates higher degree of compression (at the average 23,5 % more for pictures, typical for Windows XP comparing to zlib level 6). Meanwhile, introduced composite algorithm demonstrates higher compression speed (at the average 4 times faster comparing to zlib level 9 for pictures with text). Therefore, introduced combined algorithm can be used in practice for screen video frames compression.