УВЕЛИЧЕНИЕ СКОРОСТИ РАБОТЫ АЛГОРИТМА АДАПТИВНОГО СЖАТИЯ БИНАРНЫХ РАСТРОВЫХ ИЗОБРАЖЕНИЙ

А.В. Борусяк

Научно-исследовательский институт прикладной математики и кибернетики
Нижегородского государственного университета им. Н.И. Лобачевского
Национального исследовательского университета

В данной работе рассматривается задача увеличения скорости работы алгоритма адаптивного сжатия бинарных растровых изображений (БРИ) на основе статистического кодирования с использованием контекстного моделирования. Рассматривается влияние размера контекста максимального порядка на скорость сжатия. Рассматривается увеличение скорости алгоритма в зависимости от количества используемых потоков.
Ключевые слова: сжатие, оптимизация, бинарные изображения, контекстное моделирование

Введение

В настоящее время в связи с развитием информационных технологий, интернет технологий, передачей больших объемов графической информации, созданием удаленных баз данных, большое значение дается разработке эффективных моделей и алгоритмов сжатия данных. На данный момент предложено много различных универсальных алгоритмов сжатия. Но, в тоже время во многих областях остается актуальной разработка специализированных алгоритмов, ориентированных под конкретный тип данных. Такие алгоритмы зачастую позволяют добиться значительного преимущества при сжатии за счет знания внутренней структуры и специфики сжимаемых данных. Одной из таких областей является сжатие монохромных черно-белых бинарных изображений (МБИ). Одним из самых эффективных методов сжатия данных с определенной структурой и спецификой являются контекстные методы сжатия данных на основе метода PPM (Prediction by Partial Matching).

Алгоритм PCTB

Был разработан и реализован алгоритм сжатия МБИ на основе метода контекстного моделирования PPM[1], который был назван PTCB.

Для алгоритма была подобрана форма контекста и используются модели 31, 11, 7, 4 и нулевого порядков. Для эффективного хранения контекстных моделей в памяти используется структура хранения B-дерева.

В алгоритме использованы следующие методы и техники: собственный алгоритм оценки вероятности ухода, метод exclusion(исключение последнего встреченного символа), техника масштабирования последнего встреченного символа, модифицированный метод наследования информации.

В качестве статистического кодировщика используется арифметический кодер. Процесс декодирования симметричен процессу кодирования.

Данная реализация алгоритма показала высокую эффективность по коэффициенту сжатия ($К_{сж}$), но временные затраты были слишком большие. Благодаря оптимизации вычислений удалось снизить время, необходимое на кодирование [3]. Однако, даже после оптимизации временные затраты все равно остались достаточно большими. Кроме этого, на больших изображениях с малой избыточностью дерево контекстных моделей растет слишком быстро, что приводит к большому уровню потребляемой оперативной памяти (ОП) и частому вызову процедуры очистки дерева контекстных моделей. Процедура очистки дерева контекстных моделей необходима для ограничения объема потребляемой ОП. При этом, частый вызов данной процедуры ведет к замедлению работы алгоритма и уменьшению $К_{сж}$.

В данной работе рассматривается задача уменьшения времени работы алгоритма и количества потребляемой ОП за счет применения распараллеливания и уменьшения размера контекста максимального порядка (КМП).

Зависимость основных параметров сжатия от величины контекста

Размер КМП оказывает сильное влияние не только на степень сжатия, но также и на скорость работы алгоритма и количество потребляемой оперативной памяти. Было сделано предположение, что при уменьшении размера КМП до определенного уровня $К_{сж}$ почти не уменьшится, но при этом скорость сжатия значительно увеличится. В виду этого, были проведены эксперименты по выявлению зависимости $К_{сж}$, времени сжатия и размера потребляемой оперативной памяти от величины КМП. Результаты данного эксперимента можно наблюдать в Таблице 1. Как видно из проведенных экспериментов, при уменьшении размера контекста до 15 коэффициент сжатия падает незначительно, при этом наблюдается значительное уменьшение времени, необходимого для сжатия и уменьшение количества потребляемой оперативной памяти. Снижение величины КМП меньше 15 приводит к более значительному уменьшению коэффициента сжатия и менее значительно уменьшает потребляемую ОП и время, затрачиваемое на сжатие. В таблицах 1-3 строки соответствуют номеру сжимаемого файла, а столбцы соответствуют размеру контекста максимального порядка. При этом, снижение величины КМП ниже 20 практически не влияет на уровень потребляемой оперативной памяти.

Таблица 1. Зависимость времени сжатия (сек) от величины контекста

File\контекст	31	25	20	15	10	5	1
1	0,98	0,94	0,92	0,77	0,70	0,74	0,66
2	5,47	4,31	3,50	2,67	2,11	2,05	1,56
3	4,30	4,11	4,08	3,56	3,16	3,58	2,87
4	712,63	544,81	358,42	214,62	139,06	127,47	97,89
5	180,71	175,30	173,63	155,85	137,90	188,71	123,95

Таблица 2. Зависимость коэффициента сжатия от величины контекста

File\контекст	31	25	20	15	10	5	1
1	113,46	111,58	107,17	99,73	94,28	78,17	36,94
2	5,26	5,15	5,04	4,83	4,57	4,11	3,78
3	62,34	61,20	59,98	58,07	55,32	49,03	17,92
4	2,69	2,66	2,66	2,54	2,28	1,72	0,99
5	55,57	53,29	51,90	50,34	45,93	37,23	15,30

Таблица 3. Зависимость потребления оперативной памяти(Mb) от величины контекста

File\контекст	31	25	20	15	10	5	1
1	16	15,4	14,3	14,2	15	14,2	14,3
2	38,6	26,5	19	15,2	15,2	15,2	14,7
3	18,2	17,3	17,1	17,3	17,3	17,2	17
4	621	259,7	103,5	103	104	103	103
5	189	188,7	188,4	188,4	189	188	188,7

Распараллеливание алгоритма кодирования

Современные компьютеры чаще всего оснащены процессорами, содержащими несколько ядер, что позволяет осуществлять параллельные вычисления. С целью дальнейшего увеличения скорости сжатия для алгоритма сжатия индексированных изображений была реализована возможность кодировать и декодировать БРИ, задействуя распараллеливание процесса обработки на несколько потоков одновременно. В качестве основы алгоритма распараллеливания был взят алгоритм распараллеливания для сжатия индексированных изображений [3]. Была реализована возможность разбивать изображение на n частей. Примем вертикальный размер (высоту) изображения за H, горизонтальный размер(ширину) за W. Изображение разбивается следующим образом: если количество частей равно 4, то изображение делится по линиям соединяющим середины противоположных сторон. В случае $n \neq 4$ вертикальные стороны изображения делятся на n-1 равных частей с высотой равной H/n и одну с высотой (H-H/n*(n-1)). После разбиения изображения на несколько частей, каждая часть изображения сжимается как

отдельное изображение в отдельном потоке. Данный подход позволяет задействовать возможности современных компьютеров, равномерно распределив нагрузку на ядра процессора, при этом незначительно снизив эффективность сжатия. При реализации расчетов на двуядерном PC1(процессор – Core 2 Duo E7400 2.8 GHz) и на четырехъядерном PC2(процессор - Core i5-3230M 2.6 GHz) был получен следующий результат:

Таблица 4. Сравнение времени кодирования файла PGS2 (32294x25003, глубина цвета 1 бит, количество цветов 1) в зависимости от количества потоков

Количество потоков	Время PC1 (сек.)	Время PC2 (сек.)	Коэффициент сжатия (Ксж)
1	90,496	69,405	71,99
2	53,228	50,701	71,90
4	51,012	43,992	71,76
8	50,451	39,094	71,53
16	50,029	37,565	71,14
32	49,764	36,769	70,52
64	51,106	37,564	69,54

Где, в первом столбце таблицы 4 указано количество потоков, используемых для кодирования изображения. Во втором и третьем столбце показаны соответствующие времена кодирования в секундах для PC1 и PC2. В последнем столбце показан Ксж данного файла в зависимости от количество используемых потоков. Данный коэффициент зависит от количества потоков, но не зависит от компьютера, на котором производится кодирование, поэтому для обоих компьютеров коэффициенты совпадают. Из экспериментов видно, что при использовании 2-х ядерных и 4-х ядерных компьютеров, оптимальнее использовать 4 потока для кодирования.

Заключение

Были проведены эксперименты по увеличению скорости работы предложенного алгоритма за счет использования распараллеливания алгоритма на несколько потоков и уменьшения размера КМП. Оба подхода показали свою эффективность. Был выявлен размер КМП, уменьшение до которого значительно уменьшает время необходимое на кодирование файла и значительно снижает потребление оперативной памяти, при этом лишь незначительно снижая $К_{сж}$. Были проведены эксперименты по распараллеливанию алгоритму на несколько потоков. Используя эти два подхода совместно возможно добиться увеличения скорости сжатия в среднем до 8 раз на компьютере с 4-х ядерном процессором, снизить потребление оперативной памяти в 2 раза при потерях в $К_{сж}$ в среднем менее 10%.

Литература

1. Borusyak A.V., Vasin Yu.G., Zherzdev S.V. “Compression of Binary Graphics Using Context Simulation” // Pattern Recognition and Image Analysis, Vol.23, № 2, 2013, pp207-210
2. Borusyak A.V., Vasin Yu.G., Zherzdev S.V. “Optimizing the computational complexity of the algorithm for adaptive compression of binary raster images” The 11-th International Conference “Pattern Recognition and Image Analysis: new information technologies” Samara, September 23-28, 2013, Volume 1, pp.170-172
3. Борусяк А.В., Васин Ю.Г. Сжатие индексированных графических изображений с использованием контекстного моделирования // Вестник Нижегородского университета им. Н.И. Лобачевского, 2014, № 4-1, С. 486-492.
4. Ватолин Д., Ратушняк А., Смирнов М., Юркин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ – МИФИ, 2003.
5. Vasin Yu.G. and Zherzdev S.V. Information Techniques for Hierarchical Image Coding // Pattern Recognition and Image Analysis, Vol. 13, №. 3, 2003, pp. 539–548.
6. Набор тестовых изображений с сайта http://www.imagecompression.info/test_images