МЕТОД ИЕРАРХИЧЕСКОЙ КОМПРЕССИИ ИНДЕКСНЫХ ИЗОБРАЖЕНИЙ

Баврина[1] А.Ю. , Глумов[2] Н.И. , Сергеев[2] В.В. , Тимбай[1] Е.И.
[1] Самарский государственный аэрокосмический университет
[2] Институт систем обработки изображений РАН

Аннотация

В работе описывается метод безошибочной компрессии индексных изображений искусственного происхождения. Метод основывается на иерархическом представлении изображения в виде набора матриц или иерархических уровней (ИУ) уменьшенного размера и кодировании только информации, необходимой для восстановления очередного ИУ по восстановленным значениям предыдущих ИУ. Рассмотрены различные варианты реализации метода. Экспериментальные исследования показали эффективность предлагаемого метода по сравнению с широко используемыми стандартами безошибочного сжатия.

Введение

В работе предлагается метод безошибочной компрессии палитровых (индексных) изображений искусственного происхождения. Важным примером таких изображений являются растровые картографические изображения (рис. 1), изображения технических чертежей, диаграмм и т.д. Такие изображения создаются, хранятся и используются с помощью специализированных программных систем, как правило, в виде векторных изображений. Однако в некоторых целях (например, для передачи по Интернет) эти изображения должны быть преобразованы в растровую форму, что делает актуальной задачу их компрессии.

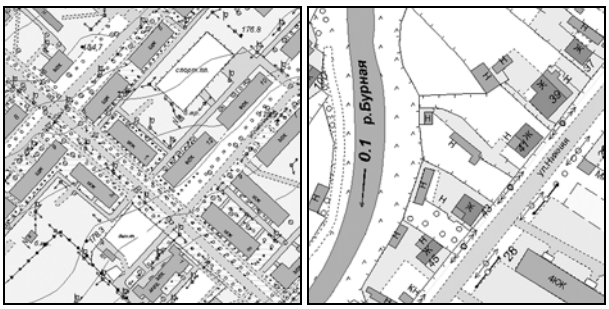

Рис. 1. Примеры картографических изображений

Как правило, изображения рассматриваемого класса содержат однородные области, очерченные контуром и залитые одним цветом, (либо некоторой текстурой с малым количеством цветов), некоторые знаки, линии, текст. При этом количество цветов на таких изображениях обычно ограничено несколькими десятками. На подобных изображениях практически не встречаются плавные изменения яркости, характерные для полутоновых и цветных (RGB) изображений.

Наиболее распространенные методы сжатия изображений не являются безошибочными и основаны на том, что небольшое изменение значения пиксела приводит к незначительному визуальному изменению изображения. Для компрессии палитровых изображений необходимо применять безошибочное сжатие, поскольку в них вместо значений яркости используются индексы (как правило, в формате байт на пиксел изображения), определяющие цвет в RGB-координатах с помощью отдельно сохраняемой таблицы (палитры). В результате ошибка в значении пиксела всего на единицу может привести к смене цвета, например, с синего на красный.

В настоящее время наиболее часто используются следующие стандарты безошибочного сжатия изображений: GIF[1], Lossless JPEG [2], JBIG [3]. Однако большинство из них не учитывают специфики изображений рассматриваемого класса и, следовательно, не могут обеспечить необходимую степень компрессии.

1. Краткое описание метода и базового алгоритма

Идея предлагаемого метода заключается в иерархическом представлении изображения в виде набора матриц или иерархических уровней (ИУ) уменьшенного размера и кодировании только информации, необходимой для восстановления очередного ИУ по восстановленным значениям предыдущих ИУ.

Пусть исходное изображение является матрицей $x(n_1, n_2)$ размером $N_1 \times N_2$, текущий l-ый уровень представляет собой матрицу $x^{(l)}(n_1, n_2)$ размером $\frac{N_1}{M_1^l} \times \frac{N_2}{M_2^l}$, где M_1, M_2 – коэффициенты масштабирования изображения при переходе к очередному ИУ. На текущем уровне строится гистограмма значений пикселов в блоках размером $M_1 \times M_2$, составляется список блоков по убыванию значений гистограммы и в очередной $l+1$-ый уровень вместо каждого блока записывается соответствующий индекс блока из списка. Очевидно, что для восстановления всех ИУ (и, следовательно, исходного изображения) достаточно иметь все списки, сформированные на уровнях, и матрицу старшего уровня размером $\frac{N_1}{M_1^L} \times \frac{N_2}{M_2^L}$, где L – количество ИУ.

Компрессия изображения обеспечивается благодаря наличию большого количества одинаковых блоков на изображениях рассматриваемого класса, информация о которых в формируемых списках записывается однократно. Кроме того, для получения большей степени сжатия списки подвергаются ста-

тистическому кодированию (например, алгоритмом арифметического кодирования [4]).

Для описания алгоритма сжатия на основе предлагаемого метода примем следующие допущения. Во-первых, ограничимся рассмотрением блоков размеров 2×2. Во-вторых, компрессируемое изображение (особенно в случае больших размеров, что характерно для изображений рассматриваемого класса) может при необходимости разбиваться на непересекающиеся фрагменты, каждый из которых компрессируется независимо от других. Далее рассматриваем в качестве изображения независимо обрабатываемые фрагменты, размеры которых являются степенью двойки, и определяются с учетом ограничений конкретного алгоритма обработки.

Алгоритм компрессии заключается в последовательном формировании и сжатии списков значений четверок для всех ИУ, начиная с нулевого уровня $x^{(0)}(n_1,n_2)=x(n_1,n_2)$. Пусть на каждом уровне формируется (и упорядочивается по частоте в порядке убывания) список значений четверок $C^{(l)}=\left\{C_k^{(l)}(0),C_k^{(l)}(1),C_k^{(l)}(2),C_k^{(l)}(3)\right\}_{k=0}^{K^{(l)}}$. Тогда отсчеты следующего ИУ формируются по правилу:

$$x^{(l+1)}(n_1,n_2)=k,$$
$$k:\begin{cases}x^{(l)}(2n_1,2n_2)=C_k^{(l)}(0)\\x^{(l)}(2n_1+1,2n_2)=C_k^{(l)}(1)\\x^{(l)}(2n_1,2n_2+1)=C_k^{(l)}(2)\\x^{(l)}(2n_1+1,2n_2+1)=C_k^{(l)}(3)\end{cases}.$$

Матрица старшего уровня $x^{(L)}(n_1,n_2)$, как и списки $C^{(l)}, 0\le l<L$, подвергается статистическому кодированию и сохраняется в массиве сжатой информации.

Параметрами данного алгоритма являются размеры фрагмента и количество ИУ, определяемые с учетом ограничений $K^{(l)}\le 255, 2^L\le N_1, 2^L\le N_2$. Следует отметить, что выполнение первого ограничения вытекает из условия сохранения всех данных (матриц ИУ и списков) в байтовом формате.

При декомпрессии матрица старшего уровня и списки ИУ декодируются, после чего последовательно восстанавливаются матрицы всех ИУ:

$$\begin{cases}x^{(l)}(2n_1,2n_2)=C_k^{(l)}(0)\\x^{(l)}(2n_1+1,2n_2)=C_k^{(l)}(1)\\x^{(l)}(2n_1,2n_2+1)=C_k^{(l)}(2)\\x^{(l)}(2n_1+1,2n_2+1)=C_k^{(l)}(3)\end{cases}, \text{ если } x^{(l+1)}(n_1,n_2)=k.$$

Однако непосредственное применение этого алгоритма неэффективно вследствие огромного количества комбинаций значений пикселов в блоках, значительная часть которых может встретиться однократно на изображении (особенно на границах областей, линий, символов). Ниже предлагаются пути сокращения объемов данных в списках, формируемых на ИУ, что приводит к руемых на ИУ, что приводит к значительному повышению эффективности компрессии.

2. Повышение эффективности базового алгоритма

Использование одного индекса для однократно встречаемых блоков

Пусть $T^{(l)}$ – количество четверок в списке $C^{(l)}$, с частотой более единицы, т.е. частоты элементов $C_k^{(l)}, k\ge T^{(l)}$ равны единице.

Тогда использование правила для формирования матрицы следующего ИУ

$$x^{(l+1)}(n_1,n_2)=\begin{cases}k, k<T^{(l)}\\T, k>=T^{(l)}\end{cases}$$
$$k:\begin{cases}x^{(l)}(2n_1,2n_2)=C_k^{(l)}(0)\\x^{(l)}(2n_1+1,2n_2)=C_k^{(l)}(1)\\x^{(l)}(2n_1,2n_2+1)=C_k^{(l)}(2)\\x^{(l)}(2n_1+1,2n_2+1)=C_k^{(l)}(3)\end{cases} \quad (1)$$

приведет к созданию более однородного изображения и значительному уменьшению длины списка на следующих ИУ.

Введение порогового значения $T^{(l)}$ позволяет смягчить ограничение на размеры фрагмента, в данном случае требуется выполнение условия $T^{(l)}\le 255$.

На рисунке 2 показан пример работы данного алгоритма для изображения размером 8×8. Матрица $x^{(0)}$ представляет исходное изображение; матрицы $x^{(1)}$, $x^{(2)}$ и списки $C^{(0)}$, $C^{(1)}$ получены с помощью правила (1).

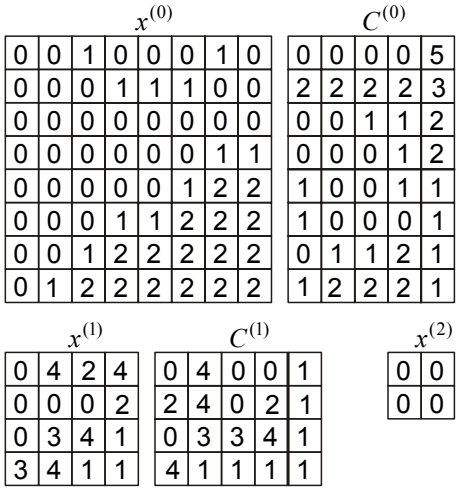

Рис. 2. Пример кодирования изображения для $N_1=N_2=8$

При декомпрессии четверка уровня $x^{(l)}$ соответствует элементу списка $C_k^{(l)}$, где

$$k = \begin{cases} x^{(l+1)}(n_1,n_2),\ x^{(l+1)}(n_1,n_2) < T^{(l)} \\ x^{(l+1)}(n_1,n_2) + P,\ x^{(l+1)}(n_1,n_2) = T^{(l)} \end{cases},$$

P – количество обработанных $x^{(l+1)}(n_1,n_2) = T^{(l)}$.

Использование шаблонов

Объем списка $C^{(l)}$ можно уменьшить за счет использования шаблонов. Вводится 8 шаблонов (рис. 3) с целью более эффективного кодирования элементов списка с числом различных значений не более 2. Большинство таких элементов имеет достаточно высокие значения частот.

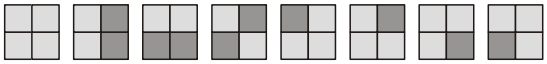

Рис. 3. Шаблоны значений отсчетов четверок с числом различных значений не более двух

Те элементы списка, которые соответствуют одному из шаблонов, вместо 4-х байт кодируются 3-мя байтами: номером шаблона и двумя цветами шаблона. Для реализуемости декомпрессии необходимо переупорядочить список таким образом, чтобы в начале шли элементы, к которым можно применить шаблон. Дополнительно сохраняется количество элементов, закодированных с использованием шаблонов. К элементам, порядковые номера которых больше порога, не применяется кодирование шаблона, так как для восстановления необходимо сохранять их порядок.

Адаптивный выбор порога

Выбор порога $T^{(l)}$ существенно влияет на степень компрессии. При уменьшении величины порога $T^{(l)}$ увеличивается объем списка $C^{(l)}$, но при этом матрица следующего уровня становится более однородной, что улучшает компрессию. Следовательно, выбор оптимального порога может обеспечить значительное повышение эффективности сжатия.

Для оценки сжатия данных текущего и следующего уровня можно использовать энтропию, вычисленную для данных списка текущего уровня и значений матрицы следующего уровня (хотя на следующем уровне будет сжиматься не матрица, а новый список, предполагаем связь между их энтропиями). Для выбора оптимального значения порога предлагается использовать критерий:

$$H\left(C^{(l)},T^{(l)}\right)\cdot V\left(C^{(l)},T^{(l)}\right) + a\,H\left(x^{(l)},T^{(l)}\right)\cdot V\left(x^{(l)}\right) \xrightarrow[T^{(l)}]{} \min, \quad (2)$$

где H – энтропия, $T^{(l)}$ – значение порога, начиная с которого элементы списка кодируются одним числом, $V\left(C^{(l)},T^{(l)}\right)$ – объем данных списка текущего уровня, $V\left(x^{(l)}\right)$ – объем данных матрицы следующего уровня, a – параметр (значение подбирается на этапе настройки алгоритма для заданного класса изображений).

Пусть $T_{opt}^{(l)}$ – оптимальное значение параметра оптимизации $T^{(l)}$ критерия (2). После определения $T_{opt}^{(l)}$ строится список, в котором элементы с частотой, меньшей или равной $T_{opt}^{(l)}$ кодируются одним числом и заносятся в список в порядке обхода.

В процессе оптимизации, частоты $T^{(l)}$, индексы которых в списке больше 256, не рассматриваются. Таким образом, алгоритм не накладывает ограничений на размер компрессируемого фрагмента.

Предлагаемый алгоритм дает значение близкое к оптимальному, которое может быть получено только путем полного перебора всех возможных значений порога (от 0 до $K^{(l)}$), однако трудоемкость предлагаемого алгоритма значительно ниже.

3. Экспериментальные исследования

Для оценки эффективности предложенного метода компрессии палитровых изображений искусственного происхождения были проведены экспериментальные исследования на тестовых изображениях, полученных из цифровых карт. Компрессия производилась на 14 изображениях размера 1024×1024 различной сложности и с разным количеством цветов.

Производилось сравнение разработанного метода (в 4 вариантах реализации алгоритма сжатия) с такими методами безошибочной компрессии, как GIF, JPEG-LS [5] и HGI (метод компрессии на основе иерархической сеточной интерполяции [6, 7]). Кроме того, производилось сравнение с широко распространенным архиватором WinZIP (версия 8.0, режим наилучшего сжатия).

Для предлагаемого метода, исходя из ограничений, накладываемых алгоритмами на размер изображения, при необходимости производилось разбиение изображений на блоки.

В таблице приведены размеры (в байтах) сжатых изображений P1,P2,…P14 для исследуемых методов сжатия. В последних двух строках таблицы содержатся суммарный объем сжатых файлов и объем сжатых файлов относительно исходных изображений. Отдельно приводится диаграмма объема сжатых файлов относительно исходных изображений (рис. 4).

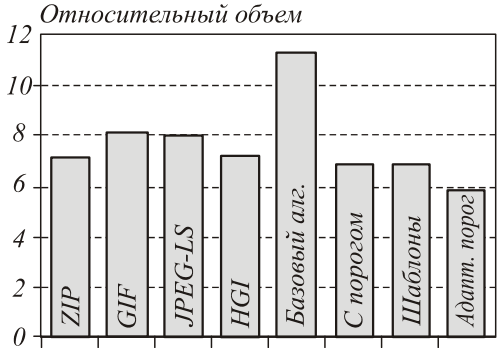

Рис. 4. Объем сжатых файлов относительно исходных изображений

	ZIP	GIF	JPEG-LS	HGI	Базовый алгоритм	С порогом	С шабло-нами	Адаптив-ный порог
P1	14611	18928	17475	25889	41466	14166	14142	11529
P2	16602	20619	17857	21149	38203	10115	10172	9520
P3	19824	29389	33915	43769	55748	21975	22191	17886
P4	47294	60034	53373	48051	87998	44088	43980	35293
P5	54686	69253	65814	54293	101927	63353	63234	44377
P6	59362	67435	59901	59600	93945	57185	57613	45210
P7	70717	80633	77570	67008	109483	68558	68916	56824
P8	78666	85156	79543	77529	119587	73974	74584	61072
P9	93937	100754	90507	89716	133814	83023	83469	68572
P10	103538	108372	108195	97900	151260	95276	95946	79364
P11	104725	124219	122482	100401	159420	104005	103917	93713
P12	117243	124559	158338	121533	178372	117494	117093	106471
P13	123153	130137	121559	116655	178691	114054	114185	99938
P14	139952	160949	158991	133297	203983	134754	134650	124686
Сумма	**1044310**	**1180437**	**1165520**	**1056790**	**1653897**	**1002020**	**1004092**	**854455**
Относительный объем, %	**7,11%**	**8,03%**	**7,93%**	**7,19%**	**11,25%**	**6,81%**	**6,83%**	**5,81%**

Таблица. Результаты экспериментальных исследований

Как и ожидалось, непосредственная реализация базового алгоритма не обеспечивает приемлемой эффективности сжатия. Однако небольшая модификация алгоритма с фиксированием одного индекса для повторяющихся блоков уменьшает объем сжатых данных на 4,44%. Исследования показали, что использование шаблонов не привело к дальнейшему улучшению алгоритма (лучшие показатели имеются только для отдельных изображений), однако пути совершенствования алгоритма в этом направлении далеко не исчерпаны.

Наибольшее сжатие из рассмотренных вариантов реализации предлагаемого метода дает алгоритм с адаптивным выбором порога. Этот алгоритм обеспечивает лучшее сжатие как по сравнению с известными стандартами GIF (в среднем на 2,22%) и JPEG-LS (в среднем на 2,12%), так и с архиватором WinZIP (в среднем на 1,20%).

Предлагаемый метод не является симметричным (по отношению времени компрессии / декомпрессии) – декомпрессия происходит быстрее, в особенности для варианта с адаптивным выбором порога. Кроме того, при декомпрессии возможно частичное восстановление изображения в требуемом масштабе (мультиразрешение), что делает метод привлекательным для приложений, связанных с передачей крупноразмерных изображений искусственного происхождения через Интернет.

Благодарности

Работа выполнена при поддержке Министерства образования РФ, Администрации Самарской области и Американского фонда гражданских исследований и развития (CRDF Project SA-014-02) в рамках российско-американской программы "Фундаментальные исследования и высшее образование" (BRHE); а также при поддержке Российского фонда фундаментальных исследований (РФФИ), проект № 04-01-96507.

Литература

1. Мюррей Д., Ван У. Райпер Энциклопедия форматов графических файлов: пер. с англ. // К.: Издательская группа BHV, 1997. 672 с.
2. Marcelo Weinberger, Gadiel Seroussi, Guillermo Sapiro, Michael W. Marcellin The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS // Hewlett-Packard Computer Systems Laboratory, HPL-98-193, November 1998.
3. JBIG, Progressive Bi-level Image compression // International Standard ISO/IEC 11544, ITU-T Recommendation, T. 82, 1993.
4. G.G. Langdon An introduction to arithmetic coding // IBM Journal of Research and Development, 1984. Vol. 28. No. 2. Pp. 135-149.
5. HP Labs LOCO-I/JPEG-LS Home Page, http://www.hpl.hp.com/loco.
6. M.V. Gashnikov, N.I. Glumov, V.V. Sergeyev Information-Processing Technology of Image Compression for Real-Time Systems// Pattern Recognition and Image Analysis, 2003. Vol. 13. No. 2. Pp. 205-207.
7. M.A. Chicheva, M.V. Gashnikov, N.I. Glumov, V.V. Sergeyev Hierarchical Approach to Image Compression for Remote Sensing Systems // Proceedings of 6th German-Russian Workshop "Pattern Recognition and Image Understanding" (OGWR-6-2003), Katun village, Altai Region, Russian Federation, 2003. Pp. 145-148.