УДК 629.7

М. В. Стержанов, аспирант (БГУИР); И. В. Байдаков, магистрант (БГУИР)

БЫСТРЫЙ АЛГОРИТМ ПОСТРОЕНИЯ СЖАТОГО ПРЕДСТАВЛЕНИЯ БИНАРНОГО РАСТРА

В статье предлагается алгоритм эффективного построения кодированного длинами серий (RLE) бинарного растра. Многие алгоритмы анализа изображений требуют использования сжатого представления. В статье обсуждаются известные подходы к решению данной задачи. Предлагаемый метод использует структуру пирамиды для быстрой обработки больших изображений. К рассмотрению прилагается эмпирическая оценка, подтверждающая эффективность данного метода построения. Преимуществом предлагаемого метода является низкая вычислительная сложность и высокое быстродействие. Заключительные результаты экспериментального сравнения производительности подтверждают эффективность разрабатываемого подхода. Алгоритм может быть применен в широком спектре задач обработки изображений.

This article proposes the algorithm for effective Run Length Encoding (RLE) of binary raster. RLE is compact and effective method for binary image representations. Many algorithms of image analysis require the usage of compressed representation. Rough and pyramid based methods are examined. Advocated method uses pyramid structure for fast processing of large images. Different approaches of pyramid structure construction are considered. The examination includes empirical evaluation acknowledging the selection of pyramid structure building method. Low computational complexity and high performance are the benefits of the proposed algorithm. The conclusive results of experimental performance comparison prove the effectiveness of proposed approach. The algorithm can be applied in a wide variety of image processing tasks.

Введение. Известно, что при выполнении различных операций над растровым изображением важным является формат представления изображения. Эффективным способом представления бинарного изображения является кодирование длин серий (Run Length Encoding, RLE) [1].

В классическом RLE-кодировании хранятся длины серий, мы используем модификацию метода, храня информацию о концевых точках серий (Run Ends Encoding, REE).

Для сжатого представления бинарного растра нами используется следующая структура данных. Каждый столбец матрицы изображения представляется упорядоченным по возрастанию координат списком вертикальных серий. Изображение хранится в виде массива списков серий (МСС). Первый элемент МСС соответствует самому левому столбцу изображения, содержащему черный пиксель. Последний элемент МСС соответствует самому правому столбцу изображения, содержащему черный пиксель. Если столбец матрицы изображения не содержит черных пикселей, то соответствующее значение элемента МСС равно нулю.

Кодирование с помощью концов серий является не просто удобным методом расходования памяти для хранения бинарного изображения, оно также хранит семантическую информацию. Так каждый отрезок на изображении представляется последовательностью смежных серий.

Реализация многих алгоритмов обработки изображений, кодированных концами серий, является более эффективной по сравнению с реализациями, использующими матричное представление изображения. В качестве примеров можно привести эффективную реализацию некоторых алгоритмов математической морфологии [2], выделения связных компонент [3], частичной скелетизации [4].

Цель данной работы – создание и реализация быстрого алгоритма построения REE бинарного растра. Основным требованием к алгоритму является высокая производительность, так как программы по обработке изображений функционируют в интерактивном режиме.

Основная часть. REE-представление может быть получено, используя так называемый метод «грубой силы» [5]. Для этого необходимо сканировать изображение вертикальными строками. Анализируя скан-строку снизу вверх, выделяются последовательности смежных пикселей максимальной длины, которые добавляются в МСС. Нетрудно показать, что нижней границей числа просмотра элементов изображения с размером $N \times N$ для любого алгоритма построения REE является $O(N^2)$, так как каждый элемент матрицы изображения посещается ровно один раз. Очевидно, что метод «грубой силы» является оптимальным по числу просмотров элементов изображения. Однако при обработке широкоформатных изображений его производительность оказывается невысокой.

Мы предлагаем представлять изображение в форме пирамиды для получения REE-кодирования. Классические пирамиды изображений впервые были представлены последовательностью изображений экспоненциально уменьшающихся размеров в работе [6]. Нижним (базо-

вым) уровнем пирамиды является исходное изображение. Каждый последующий уровень (L) строится по предыдущему ($L-1$) при помощи операции фильтрации. Основными областями применений пирамид изображений являются: поиск по масштабу, пространственный поиск, отслеживание особенностей.

Структура классической пирамиды может быть описана с помощью двух параметров: коэффициента сжатия и «сжимающего окна». Коэффициент сжатия r определяет пропорцию изменения размеров уровней $L-1$ и L. Учитывая особенности аппаратной реализации, коэффициент сжатия зачастую выбирается кратным 2. «Сжимающее окно» (как правило, квадрат $n \times n$) ставит в соответствие ячейке на текущем уровне L множество ячеек на низлежащем уровне $L-1$.

Структура пирамиды является иерархичной: в ней можно выделить отношение смежности (между ячейками на одном уровне L) и отношение «родитель – потомок» (между ячейками на уровнях L и $L-1$).

Под уровнем пирамиды L понимается трансформированное (уменьшенное) L раз исходное изображение. Базовым (нижним, нулевым) уровнем является исходное изображение. Каждый уровень представляет собой матрицу, состоящую из ячеек. Ячейки базового уровня могут принимать только два значения: 0 и 1. Значение ячейки небазового уровня находится в пределах от 0 до коэффициента сжатия r.

Каждая ячейка на уровне L (за исключением базового уровня) имеет множество дочерних ячеек на низлежащем уровне $L-1$, по которым она строится. Аналогично каждая ячейка на уровне L (за исключением вершины пирамиды) имеет множество родителей на уровне $L+1$. Каждая ячейка имеет множество братьев, находящихся на одном уровне с ней.

Построение пирамиды осуществляется снизу вверх, т. е. от нижних уровней к верхним. Пусть изображение представлено матрицей I, состоящей из N строк и M столбцов. Первый уровень будет иметь уменьшенные в r раз размеры исходного изображения. Значением ячеек матрицы I_1 первого уровня является количество черных пикселей «сжимающего окна» базового уровня. Для каждого последующего уровня значение ячейки определяется как среднее значение пикселей «сжимающего окна» предыдущего уровня.

Возможно построение пирамиды изображений с помощью интегральной матрицы изображений, описанных в работе [7].

Интегральная матрица изображения вычисляется рекурсивно:

$$s(i,\ -1) = 0, \tag{1}$$

$$b(-1, j) = 0, \tag{2}$$

$$s(i, j) = s(i, j-1) + I(i, j)\,, \tag{3}$$

$$b(i, j) = b(i-1, j) + s(i, j), \tag{4}$$

где $s(i, j)$ – сумма значений пикселей по столбцам; I – исходное изображение; $b(i, j)$ – интегральная матрица изображений.

Формулы (1) и (2) введены из-за рекурсивной природы вычисления матрицы.

Алгоритм построения пирамиды основан на быстром вычислении суммы значений пикселей в произвольной прямоугольной области [8]. Средние значения определяются в непересекающихся прямоугольных или квадратных областях исходного изображения и формируют значения пикселей нового уровня пирамиды. При вычислении следующего уровня пирамиды размеры непересекающихся областей предыдущего уровня увеличиваются в r раз. Отметим, что для того, чтобы найти i-й уровень пирамиды, нет необходимости вычислять предыдущие ($i-1$) уровни пирамиды изображений.

Введем понятие шаблона BlackPixelIterator. Пусть исходный растр представлен классическим образом – 1 бит на пиксель. Для уменьшения операций обращения к памяти будем хранить пиксели в машинном слове (32 бита). Многие алгоритмы можно значительно ускорить, если производить обработку только черных пикселей: передавать на вход только координаты черных пикселей, считая, что все остальные пиксели являются белыми. Это не подходит для методов, требующих доступа к произвольному пикселю. Например, не подходит для вычисления интегральной матрицы. Количество черных пикселей в инженерных изображениях гораздо меньше, чем белых. Пусть $B(I)$ – количество черных пикселей изображения I, $A(I)$ – количество пикселей (площадь) изображения I. Пусть F – подпрограмма, которая реализует некоторый алгоритм, входными данными которого являются координаты черных точек. Шаблон BlackPixelIterator осуществляет итерацию по всему изображению I, поэтому количество обращений к памяти будет $A(I)$ / 32. Однако количество вызовов подпрограммы F будет равно $B(I)$.

Мы разработали следующий быстрый способ построения пирамиды (FastPyramid). Он опирается на то, что черные пиксели в бинарном растре представлены единицами, а белые – нулями. Следовательно, нет необходимости учитывать значения, представленные белыми пикселями. Поэтому для вычисления значений уровней пирамиды мы учитываем только черные пиксели, т. е. накапливаем суммы «вверх». Для этого подходит использование шаблона BlackPixelIterator. За одну итерацию по исходному изображению (базовому уровню) мы

строим всю пирамиду – суммируем только черные пиксели. Для ускорения базовое изображение используется в качестве нулевого уровня пирамиды, т. е. применяется шаблон «адаптер». Поэтому количество операций суммирования будет равно $B(I) \cdot (K - 1)$, где K – количество уровней пирамиды. Количество операций записи в память будет также равняться $B(I) \cdot (K - 1)$, так как мы каждый черный пиксель учитываем сразу на всех уровнях пирамиды. Количество обращений к памяти (чтение) будет $A(I) / 32$.

Приведем результаты эксперимента трех способов построения пирамиды на 127 изображениях при коэффициенте сжатия, равном 16 (табл. 1).

Таблица 1

Анализ производительности методов построения пирамиды

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
Классический	4	192	32,9	29,5
Интегральная матрица	8	861	84,5	112,2
Быстрый	2	168	20,1	20,0

Примечание. Min – минимальное время построения пирамиды, Max – максимальное время построения пирамиды, Avg – среднее выборочное время построения пирамиды, Disp – квадратный корень среднеквадратичного отклонения времени построения пирамиды.

Представленные данные свидетельствуют о том, что предложенный метод является самым быстрым, а метод на основе интегральной матрицы – самым медленным. Метод BlackPixelIterator сокращает число обращений к памяти изображения, поэтому он строит пирамиду быстрее классического метода.

Для построения пирамиды по интегральной матрице характерно следующее: количество операций суммирования действительно сократилось [8]. Но количество обращений к памяти для построения первого уровня пирамиды увеличилось более чем в 2 раза. Пусть размер исходного изображения будет $N \times N$. При построении интегральная матрица для нулевого уровня производит N^2 обращений к памяти базового изображения и $B(I)$ обращений к памяти с самой матрицей. При построении матрицы первого уровня происходит N^2 обращений к интегральной матрице нулевого уровня.

Очевидно, что если значение ячейки B пирамиды равно r, то все ячейки, входящие в дерево потомков B, также имеют значение r. А это в свою очередь означает, что на базовом уровне все пиксели, участвовавшие в формировании значения B, являются черными. Такую ячейку будем называть «черной». Если значение ячейки G пирамиды находится в интервале от 0 до r, то среди всех ячеек, входящих в дерево потомков G, найдется хотя бы одна ячейка W со значением 0. В данном случае среди пикселей базового уровня, участвовавших в формировании G, будут как черные, так и белые. Такую ячейку будем называть «серой». Если значение ячейки W пирамиды равно 0, то все ячейки, входящие в дерево потомков W, также имеют значение 0. В данном случае среди пикселей базового уровня, участвовавших в формировании W, будут только белые. Такую ячейку будем называть «белой».

Итак, пусть пирамида бинарного растра построена. Каждый уровень пирамиды (за исключением базового) может быть представлен тремя REE-кодами: черным, серым и белым. Черный REE-код состоит строго из «черных» ячеек, серый – из «серых», белый – из «белых». Поскольку наша цель заключается в построении REE-кода черных пикселей базового уровня, то мы будем рассматривать только черные и серые REE-коды на каждом уровне (за исключением базового). Черный и серый REE-коды уровней пирамиды строятся сверху вниз. Для этого рассматривается самый верхний уровень пирамиды, состоящий из одной ячейки. Если эта ячейка «белая», основной алгоритм построения REE завершает свою работу, так как исходное изображение не имеет черных пикселей. Если ячейка вершины пирамиды «черная», основной алгоритм также заканчивает свою работу, возвращая REE, в котором число серий равно ширине изображения, а размер каждой серии равен высоте изображения. Если ячейка вершины пирамиды является «серой», то в сером REE будет одна серия единичного размера, а черный REE будет пустым. Построение REE каждого оставшегося уровня L осуществляется с использованием информации о REE верхнего уровня $(L + 1)$, так как «черные» ячейки имеют только «черных» потомков, а «серые» ячейки имеют хотя бы одного потомка, не являющегося «черным».

Итак, пусть рассчитаны REE-коды серой и черной составляющей на уровне L (в данном случае на верхнем уровне). Обозначим через G_{REE_L} и B_{REE_L} «серую» и «черную» REE-компоненты уровня L. Увеличим G_{REE_L} и B_{REE_L} в r раз до достижения размеров уровня $L - 1$. Присвоим REE-кодам уровня $L - 1$ трансформированное значение REE-кодов уровня L.

Таким образом, получено предварительное значение REE-кодов уровня $L - 1$, которое будет уточняться с помощью следующих шагов.

Шаг 1. Рассмотрим каждую серию X, принадлежащую B_{REE_L-1} и имеющую значения ($x1$, $a1$, $b1$). Проверим значение ячейки A уровня $L - 1$ пирамиды в позиции ($x1$, $b1 + 1$). Если данная ячейка является «черной», то исходная серия X увеличивается в размерах, так как ячейка A

принадлежит ей. Затем анализируется ячейка с координатами ($x1$, $b1 + 2$). Процесс повторяется до тех пор, пока не достигнута граница изображения, либо рассматриваемая ячейка является «черной». Аналогичная операция выполняется для ячейки с координатами ($x1$, $a1 - 1$), только в данном случае рассматриваются ячейки, находящиеся снизу от серии X.

Шаг 2. Рассмотрим каждую серию X, принадлежащую G_{REE_L} и имеющую значения ($x1$, $a1$, $b1$). Рассмотрим ячейку уровня $L - 1$ с координатами ($x1$, $b1$). Если она является «белой», то данная ячейка не принадлежит серии X, и серия X уменьшается в размерах. Затем анализируется ячейка с координатами ($x1$, $b1 + 1$). Данный процесс продолжается до тех пор, пока рассматриваемая ячейка является «белой». Аналогичная операция выполняется для ячейки с координатами ($x1$, e), только в данном случае рассматриваются ячейки, лежащие ниже ячейки ($x1$, e).

Шаг 3. Рассмотрим два списка серий с одинаковой координатой pos: G_{REE_List} и B_{REE_List}. Для каждой серии B_{CUR} из B_{REE_List} будем искать серию G_{CUR} из списка G_{REE_List}, такую, что координаты G_{CUR} и B_{CUR} пересекаются. Если серия G_{CUR} найдена, то изменим ее координаты таким образом, чтобы они не пересекались с B_{CUR}.

Шаг 4. Рассмотрим каждую серию X, принадлежащую G_{REE_L-1}. Проанализируем значение ячеек пирамиды уровня $L - 1$ с координатами, равными координатам серии. Если рассматриваемая ячейка «черная», то она удаляется из серии X и добавляется в черный REE-код, если «серая», то остается в «серой» REE-составляющей.

Заключение. Приведем результаты экспериментального сравнения быстродействия алгоритмов построения сжатого REE-кода (табл. 2). Тестирование выполнялось пакетным способом на 127 тестовых штриховых изображениях при значении сжимающего коэффициента, равном 16. Обрабатывались типовые штриховые изображения (схемы электрические принципиальные, различные чертежи, планы).

Таблица 2

Анализ производительности методов построения сжатого растра

Метод	Min, мс	Max,мс	Avg, мс	Disp, мс
«Грубой силы»	4	385	43,87	59,16
Быстрый	2	168	20,13	20,03

Примечание. Min – минимальное время построения REE, Max – максимальное время построения REE, Avg – среднее выборочное время построения REE, Disp – квадратный корень среднеквадратичного отклонения времени построения REE.

На основании предлагаемого шаблона BlackPixelIterator разработан и реализован быстрый метод построения сжатого REE-представления. Также в работе проводится сравнительный анализ методик построения пирамиды изображений, оценивается их сложность и обоснованность практического применения. Предлагаемый метод построения пирамиды является самым быстрым, что подтверждают результаты эксперимента. Преимуществом предлагаемого подхода является низкая вычислительная сложность и высокая эффективность, что было доказано экспериментально. Благодаря этому методика может применяться в широком спектре задач обработки бинарных изображений, использующих сжатое представление растра.

Литература

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с.

2. Буча, В. В. Алгоритмы интерактивного выделения и векторизации объектов на цифровых изображениях местности: дис. ... канд. техн. наук / В. В. Буча. – Минск, 2006. – 133 л.

3. Shapiro, L. Connected Component Labeling and Adjacency Graph Construction / L. Shapiro // Topological algorithms for digital image processing / T. Kong [et al.]. – Amsterdam, 1996. – P. 1–31.

4. Стержанов, М. В. Алгоритм векторизации штриховых бинарных изображений / М. В. Стержанов, И. В. Байдаков // ММРО: материалы 14-й Всерос. конф., Суздаль, 21–26 сент. 2009 г. / Универ; редкол.: И. И. Иванов [и др.]. – М., 2009. – С. 449–452.

5. McConnell, J. Analysis of Algorithms: An Active Learning Approach. Second Edition / J. McConnell. – New York: Jones & Bartlett, 2008. – 451 p.

6. Pyramid Methods in Image Processing / E. H. Adelson [et al.] // RCA Engineer. – 1984. – Vol. 29(6). – P. 33–41.

7. Viola, P. Robust real-time object detection / P. Viola, M. Jones // Second Int. Workshop on Computer Vision. – Vancoever, 2001. – P. 137–154.

8. Макаров, А. О. Быстрая обработка изображений на основе интегральных матриц изображений / А. О. Макаров, В. В. Старовойтов // Искусственный интеллект. – 2006. – № 3. – С. 597–602.

Поступила в редакцию 31.03.2010