УДК 004.925

М. С. Пелевин
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Применение алгоритма Крускала для построения иерархии сегментированного изображения

Приведен пример применения алгоритма Крускала для построения сегментированного изображения с сохранением его в системе непересекающихся множеств как иерархического изображения. Проведен анализ скорости работы для простого некумулятивного сегментатора «Заливка».

Сегментация изображений, алгоритм Крускала, иерархия изображения, система непересекающихся множеств

В современных системах обработки изображений при сегментировании [1] требуется получить результат для нескольких значений параметра, определяющего результат сегментации. Для большинства систем используется подход, при котором экспериментально устанавливаются оптимальные для обработки в данной системе параметры. Такой подход не всегда удобен, в частности, когда система сталкивается с новыми изображениями, для которых программа еще не была «подкручена». В таких случаях удобно получить все возможные виды изображения при различных значениях параметра сегментирования.

Как правило, для этих целей используется последовательное либо параллельное вычисление сегментированного изображения, что накладывает дополнительные требования на возможности вычислительной системы (дополнительная оперативная память, наличие возможности распараллеливания и т. д.).

Иерархией изображений в общем случае называется последовательность вложенных разбиений изображений с убывающим числом сегментов. Пример такого разбиения изображен на рис. 1, где множества A1, A2, A3 являются сегментами для нижнего разбиения. Объединение множеств A2 и A3 дает новое множество B2, которое вместе с множеством B1 (эквивалентным множеству A1) при объединении дает множество C1, являющееся верхним разбиением в иерархии.

Для построения иерархии изображений обычно используется последовательный пересчет для каждого нового уровня, где уровень определяется значением параметра. Значение параметра монотонно увеличивается или уменьшается в зависимости от специфики конкретного алгоритма.

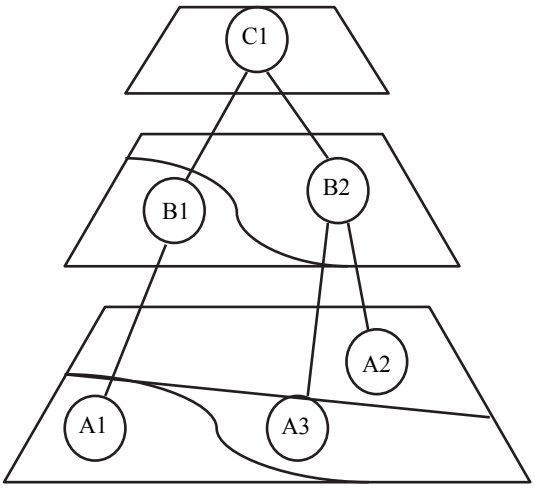

Рис. 1

Система непересекающихся множеств является структурой данных, для которой определено некоторое изначальное множество элементов и несколько операций:

1. Создание множества *x*.

2. Поиск множества для заданного элемента множества. Такая операция главным образом дает ответ на вопрос: «Какому множеству принадлежит данный элемент».

3. Объединение двух множеств в одно.

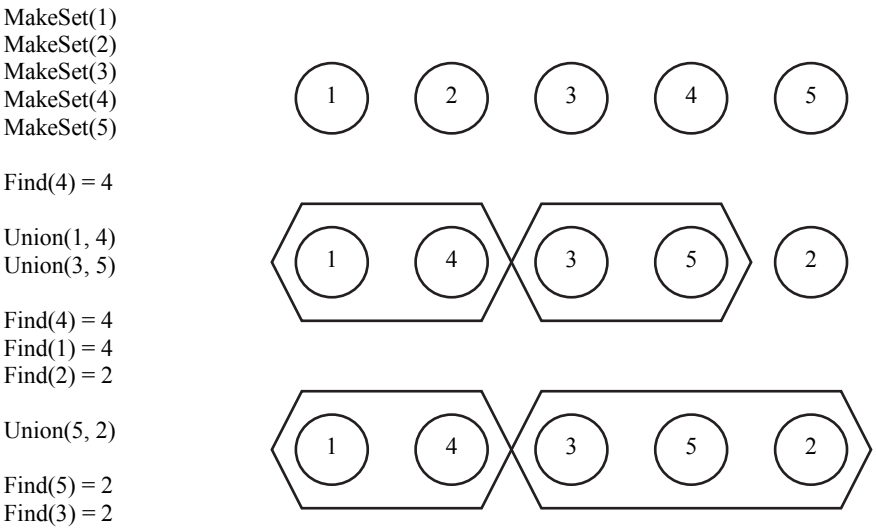

Рис. 2

На рис. 2 приведен пример работы с системой непересекающихся множеств, где каждой операции соответствует определенное название: создание множества – MakeSet(x), поиск множества – Find(x), объединение множеств – Union(x, y).

Если в приведенном примере на последнем множестве применить операцию Union(1, 5), то получится единственное множество, содержащее все элементы множества.

Отметим, что последовательность выполненных операций соответствует определению иерархии, приведенному ранее. Необходимо также обеспечить возможность сохранения дополнительной информации о том, как выглядело множество на конкретном шаге объединения. Структура данных, которая хранит информацию обо всех изменениях, называется персистентной [2].

Хранение персистентой системы непересекающихся множеств для изображения возможно с использованием двух дополнительных массивов, размерность каждого из которых равна размерности исходного изображения.

Пример хранения изображения в таком массиве для множества, изображенного на рис. 2, приведен далее. Первый массив хранит сами данные; второй – ссылку на тот элемент множества, который в данный момент является для элемента родительским; третий массив хранит номер операции или группы операций, который привел к изменению родительского элемента:

1. Создание элементов множества

Вершины	1	2	3	4	5
Ссылка на родительскую вершину	1	2	3	4	5
Номер операции	0	0	0	0	0

2.Union(1, 4) и Union(3, 5)

Вершины	1	2	3	4	5
Ссылка на родительскую вершину	4	2	5	4	5
Номер операции	1	0	1	0	0

3.Union(5, 2)

Вершины	1	2	3	4	5
Ссылка на родительскую вершину	4	2	5	4	2
Номер операции	1	0	1	0	2

Из результатов последней операции можно заключить следующее:

1. Осталось 2 независимых множества (определяется числом элементов, которые по-прежнему ссылаются сами на себя).

2. Идентификаторами данных множеств являются элементы 2 и 4, поэтому любая операция Find(x) будет давать одно из этих двух значений.

3. На первом шаге было сделано 2 объединения, на втором – 1.

Для получения структуры данных на любом из шагов достаточно задать шаг, для которого необходимо получить состояние системы непересекающихся множеств (например, 1), а для всех ссылок на родительские элементы, номер шага которых больше заданного, «сбросить» значение родительской ссылки, т. е. установить значение, равное исходному (т. е. самому себе).

Если теперь взять иерархию (являющуюся деревом) и для каждой связи проставить номер шага, на котором была сделана операция объединения, то получится структура, которую можно непосредственно сохранить в персистентной системе непересекающихся множеств.

Рассмотрим изображение в качестве графа, для каждой точки которого определены 4 ребра, соединяющих ее с соседними точками слева, снизу, справа и сверху. Для такого графа возможно построить минимальное остовное дерево. Такое дерево будет также являться иерархией изображения, где веса дерева соответствуют номеру шага, на котором была выполнена операция объединения. Пример взвешенного графа размерностью 3 × 3 изображен на рис. 3, где светлыми линиями отмечены ребра графа, которые не входят в минимальное остовное дерево, но остаются ребрами самого графа.

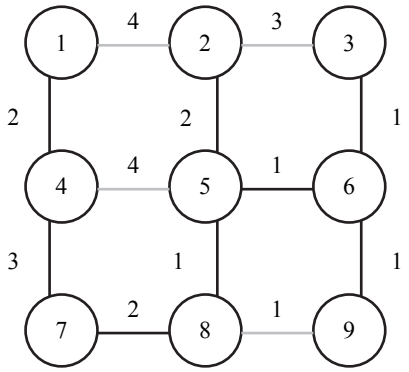

Рис. 3

Покажем, что алгоритм Крускала для исходного изображения будет создавать именно такую иерархию. Для этого положим, что используется алгоритм «Заливка»[1]. Для заливки определяется параметр максимального расхождения в цвете между двумя точками. Если для любого ребра этот параметр больше реального значения, то две точки объединяются в один сегмент, если нет – то игнорируются. Пример результирующего дерева изображен на рис. 4.

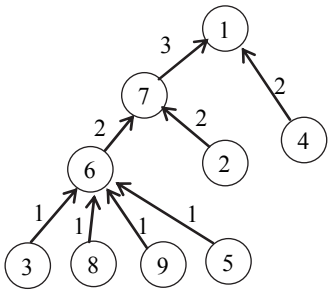

Рис. 4

Чтобы построить исходный граф изображения, необходимо аналогично алгоритму заливки проставить в качестве веса графа разницу между двумя точками (вершинами). Для черно-белого изображение эта разница является результатом вычитания яркостей двух точек, взятым по модулю.

Поскольку алгоритм Крускала сначала сортирует ребра в порядке их увеличения, а затем начинает объединять таким образом, чтобы не создавалось цикличных путей в графе, очевидно, что такая последовательность полностью совпадает с операциями поиска и объединения для системы непересекающихся множеств. Так как веса располагаются в порядке увеличения, они выполняют роль номера операции. Таким образом, можно определить вид изображения для конкретного параметра, используя свойства персистентной системы множества.

Работа данного алгоритма состоит из двух частей:

1. Расчет весов ребер графа изображения.
2. Поиск минимального остовного дерева.

Так как время вычисления веса ребра постоянно, то время расчета весов всех ребер будет в точности совпадать с их количеством, т. е. $2nm - m - n$. Алгоритм Крускала требует $e \log e + a$, где a столь мало, что его можно считать константой и не брать в расчет, а e – число ребер графа, т. е. $2mn - n - m$ (m и n – ширина и высота исходного изображения соответственно). Заполнение дерева на основе полученного минимального остовного дерева требует $O(nm - 1)$ времени.

Таким образом, суммарное время работы алгоритма составляет $e + e \log e + nm$, где $e = 2nm - m - n$, т. е. алгоритм работает со скоростью, сопоставимой с $O(e \log e)$. Здесь O – верхняя оценка скорости работы алгоритма. Таким образом, полученное значение никак не зависит от количества возможных значений исходного параметра, что позволяет обрабатывать мультиспектральные и черно-белые изображения с одинаковой алгоритмической скоростью с таким же результатом, как и при последовательном сегментировании для различных параметров.

[1] http://en.wikipedia.org/wiki/Flood_fill.

СПИСОК ЛИТЕРАТУРЫ

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2006.

2. Sleator D., Tarjan R. Making Data Structures Persistent // J. of Computer and System science. 1989. Vol. 38, № 1. P. 39.

M. S. Pelevin
Saint-Petersburg state electro technical university «LETI»

APPLICATION OF KRUSKAL'S ALGORITHM FOR CONSTRUCTING A HIERARCHY OF SEGMENTED IMAGE

This article discusses application of Kruskal's algorithm for constructing a segmented image as hierarchical structure with persisting in union-find set data structure. A short analysis of algorithm performance for simple non-cumulative segmentator named Floodfill is given.

Image segmentation, Kruskal- algorithm, image pyramid, union-find set