Методики выделения связных компонент в штриховых бинарных изображениях

Максим Стержанов
БГУИР
Минск, Беларусь
accept@bk.ru

Аннотация

В данной работе рассматривается задача выделения связных компонент (СК) штриховых бинарных изображений. Производится классификация и описание алгоритмов. Рассматриваются некоторые методики оптимизации. Осуществляется исследование и сравнительный анализ методов, оценивается их производительность. Предлагается модификация классического алгоритма выделения СК на сжатом растре.

Ключевые слова: *бинарный растр, связная компонента, штриховое изображение.*

1. ВВЕДЕНИЕ

Под выделением связных компонент понимают присвоение уникальной метки каждому объекту изображения [1]. При последующем анализе данные метки служат в качестве идентификаторов при обращении к объектам. Это делает операцию выделения связных компонент неотъемлемой частью почти всех приложений распознавания образов и компьютерного зрения. Например, перед тем как компьютер может определить или классифицировать любой объект изображения (автомобиль, человека, внутренний орган) группы смежных пикселей должны быть идентифицированы и промаркированы. Каждая выделенная группа пикселей соответствует объекту на изображении. Такая группировка смежных пикселей позволяет исследователю получить необходимые для последующего анализа свойства объектов, такие как высота, ширина, периметр, площадь. Очевидно, что задача выделения связных компонент является фундаментальной задачей обработки изображений. Следует отметить, что для многих приложений данная операция является наиболее времязатратной, т. е. критической. По этим причинам выделение связных компонент до сих пор остается активной областью исследований. Некоторые недавние работы по данной тематике представлены в [2-3]. В настоящей работе проводится классификация и сравнительный анализ алгоритмов выделения связных компонент с целью определения алгоритма, имеющего наилучшую производительность при работе со штриховыми изображениями, объекты которых представлены совокупностью линий, имеющих относительно одинаковую толщину на всем протяжении. Особенностями штриховых изображений являются: большой размер (A0), смешанная графическая и текстовая информация, различные типы объектов (отрезки, символы, дуги кривых). Примерами штриховых изображений являются технические чертежи, электрические схемы, таблицы, поэтажные планы зданий.

2. НЕКОТОРЫЕ ОПРЕДЕЛЕНИЯ

Дадим некоторые определения. Два пикселя (B или W) называются связными, если они являются соседями (расстояние между ними равно 1) в выбранной метрике. Заметим, что определение связности симметрично для черных и белых пикселей. Связная компонента (connected component) изображения (СК) – это связное множество пикселей в соответствии с выбранным типом метрики [4]. Мы будем использовать 8-связную метрику. На рис.1 показано бинарное изображение с пятью связными компонентами. СК можно считать единственной структурной единицей растрового изображения.

Под маркировкой связных компонент бинарного изображения B будем понимать формирование маркированного изображения LB, в котором каждому пикселю присвоена метка связной компоненты, которой принадлежит данный пиксель.

Метка представляет собой некоторый идентификатор, используемый в качестве уникального имени объекта. Хотя иногда применяются символьные метки, для маркировки связных компонент часто более удобными оказываются метки в виде положительных целых чисел. На рис 1.б показаны маркированные связные компоненты бинарного изображения рис. 1.а.

1	1	0	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0
0	1	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0
0	0	1	1	1	0	1	0	0
1	1	1	0	1	0	0	0	0
0	0	0	0	0	0	0	1	1

а)

1	1	0	0	1	0	3	0	0
0	1	0	0	1	0	3	0	0
0	1	1	1	1	0	3	3	3
0	0	0	0	0	0	0	0	3
2	2	2	0	0	0	0	0	0
0	0	2	2	2	0	4	0	0
2	2	2	0	2	0	0	0	0
0	0	0	0	0	0	0	5	5

б)

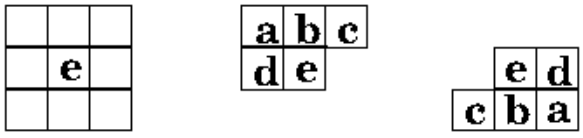
Рисунок 1: Маркировка связных компонент: а) исходное изображение; б) маркированное изображение.

Алгоритмы выделения СК могут быть разделены на следующие пять групп [5]: многопроходные, двухпроходные, однопроходные, алгоритмы с использованием иерархических структур представления изображения, параллельные алгоритмы. В данной работе рассматриваются алгоритмы, принадлежащие первой, второй и третьей группам.

Обозначим через I матрицу изображения. Если I(i,j)=0, то пиксель является фоновым. Если I(i,j)=1, то пиксель принадлежит объекту. Через L обозначим двумерную матрицу, которая используется для хранения информации о метках и имеющую размеры, равные размерам изображения.

3. МНОГОПРОХОДНЫЕ АЛГОРИТМЫ

Алгоритмы данного семейства последовательно сканируют изображение. При первом, начальном проходе каждый объектный пиксель получает временную метку. На каждой итерации значения меток уточняются. Алгоритм заканчивает свою работу тогда, когда не может быть сделано ни одного изменения значения метки [6]. Метки, которые назначаются пикселям объектов до финальной маркировки, называются промежуточными. Скан-проходы осуществляются в противоположных направлениях и чередуются. При прямом проходе изображение сканируется слева направо и сверху вниз. При обратном проходе изображение сканируется справа налево и снизу вверх. Каждый раз при обнаружении черного пикселя его соседи, принадлежащие скан-маске (рис. 2), исследуются для определения метки, которая будет присвоена рассматриваемому пикселю. Если в скан-маске содержатся только фоновые пиксели, то рассматриваемый пиксель получает новую промежуточную метку. Если скан-маска содержит только один пиксель интереса, то рассматриваемый пиксель получает метку соседа. Если скан-маска содержит несколько точек интереса, то их промежуточные метки являются эквивалентными, среди них выбирается метка, представляющая все эквивалентные метки (метка-представитель). В данном случае пикселю присваивается значение выбранной метки-представителя. В простейшем случае в качестве метки представителя выбирается метка с наименьшим значением. Назовем данный алгоритм MPS (Multi-Pass Simple).



а) б) в)

Рисунок 2: Используемая нотация: а) 8 соседей пикселя; б) скан-маска прямого прохода; в) скан-маска обратного прохода.

Обозначим пиксели скан-маски через *a*, *b*, *c*, *d*, *e* и будем использовать эти обозначения вместо их (i, j) координат в дальнейшей нотации. Тогда $L[e]$ будет обозначать метку текущего пикселя. Через *l* обозначим целое число, инициализированное значением 1. Присвоение промежуточных меток в течение первого скан-прохода может быть представлено в виде:

$$L[e]=\begin{cases}0, если\ I[e]=0\\ l,(l=l+1), если\ \forall i\in(a,b,c,d),I[i]=0\\ \min(L[i])\,(i\in(a,b,c,d)\,|\,I[i]=1),\ в\ противном\ случае\end{cases} \quad (1)$$

При последующих скан-проходах метки пикселей получают значение, выраженное следующей формулой:

$$L[e]=\min(L[i]),i\in(a,b,c,d)\,|\,I[i]=1,$$
$$если\ I[e]=1\ и\ \exists i\in(a,b,c,d),такое,что\ I[i]=1 \quad (2)$$

Недостатком алгоритма MPS является большое количество скан-проходов, необходимое для конвертации промежуточных меток. Например, при обработке тестового изображения «Зерносушилка» (см. рис. 3) размерами 6206×5883 пикселей требуется 116 прямых и обратных проходов по изображению.

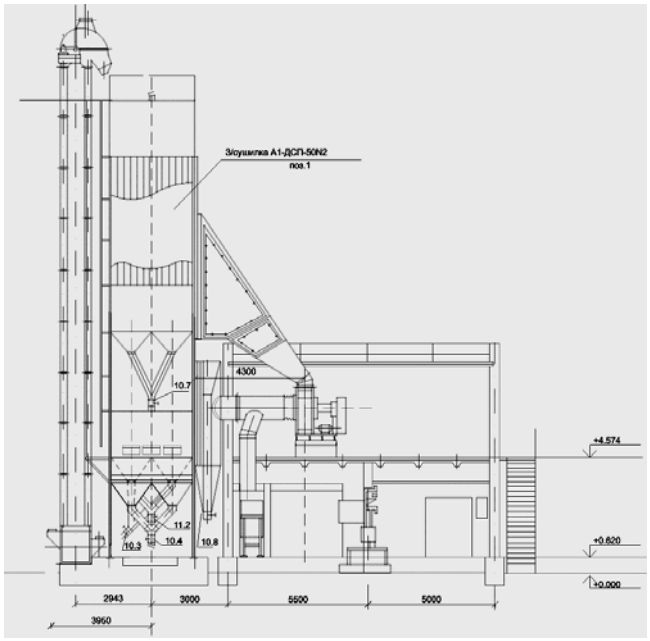


Рисунок 3: Тестовое изображение «Зерносушилка»

Алгоритм может быть улучшен при помощи использования таблицы связности (connection table), хранящей информацию об эквивалентных метках. Таблица связности, предложенная в работе Сузуки [7] является одномерным массивом, имеющим число элементов, равное числу промежуточных меток. Назовем данный алгоритм MPS_CT. Обозначим через *T* таблицу связности. При первом проходе значения массивов *L* и *T* принимают следующие значения:

$$L[e]=\begin{cases}0; если\ I[e]=0\\ l,T[l]=l,l=l+1; если\\ \forall i\in(a,b,c,d)\,I[i]=0\\ \min(T[L[i]]),T[L[i]]=L[e]; если\\ \forall i\in(a,b,c,d)\,|\,I[i]=1\end{cases} \quad (3)$$

При последующих проходах происходит изменение меток пикселей объектов, которые содержат смежные черные пиксели, принадлежащие скан-маске. Значения меток определяются при помощи следующей формулы:

$$L[e]=_{i\in(a,b,c,d)|I[i]=1}\min(T[L[i]]),$$
$$\forall i\in(a,b,c,d)\,|\,I[i]=1,T[L[i]]=L[e] \quad (4)$$

Использование таблицы связности способствует более быстрому распространению меток по сравнению с аналогичными алгоритмами. Проведенные нами эксперименты показывают, что в большинстве случаев данному алгоритму требуется не более 4 скан-проходов. В таблице 1 приведем результаты тестирования многопроходных алгоритмов. Во всех экспериментах тестирование производительности осуществлялось пакетным способом на 127 широкоформатных штриховых изображениях.

Таблица 1. Производительность многопроходных алгоритмов.

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
MPS	1367	130596	21689	33662
MPS_CT	876	13054	3815	3746

Примечание: В таблицах 1-5 приведено время выделения СК: *min* – минимальное, m*ax* – максимальное, a*vg* – среднее выборочное, *Disp* – квадратный корень среднеквадратичного отклонения.

4. ДВУХПРОХОДНЫЕ АЛГОРИТМЫ

Алгоритм, описанный в статье [8], часто называется классическим, так как он основан на классическом алгоритме поиска связных компонент на графах. Существуют различные модификации алгоритма, его общая структура заключается в следующем. Изображение последовательно сканируется сверху вниз и слева направо. Новые метки СК присваиваются первому пикселю обнаруженной СК, затем присвоенная метка распространяется на смежные пиксели. Рассмотрим пример на рис. 4.

1	0	0	1
1	1	0	1
1	0	0	1
1	1	1	1

а)

1	0	0	2
1	1	0	2
1	0	0	2
1	1	1	A

б)

Рисунок 4: Пример маркировки: а) исходное изображение; б) маркированное изображение.

В первой строке имеется два черных пикселя, разделенные двумя белыми. Первый из них получает метку со значением 1, второй – метку со значением 2. Во второй строке первый пиксель получает метку со значением 1, т. к. его сосед сверху уже помечен. Второй пиксель во второй строке также получает метку со значением 1, т. к. его сосед слева помечен. Аналогичные действия выполняются и для третьей строки матрицы изображения. Рассмотрим пиксель A в четвертой строке. Его сосед слева имеет метку со значением 1, а сосед сверху имеет метку со значением 2. Следовательно, все пиксели, имеющие метки со значением 1 и 2 принадлежат одной связной компоненте, другими словами, эти метки являются эквивалентными. Существующие модификации алгоритма отличаются в решении следующих вопросов:

1. Какая метка должна быть присвоена пикселю A?
2. Каким образом хранится информация об эквивалентных метках?
3. Каким образом информация об эквивалентности преобразуется при окончательной разметке?

Классический алгоритм [8] требует большой глобальной таблицы для хранения информации об эквивалентности меток. На первом проходе, называемом фазой сканирования, все объектные пиксели изображения получают временные метки. Если одному пикселю может быть присвоено несколько меток, то выбирается метка с минимальным значением, а информация об эквивалентности меток заносится в таблицу эквивалентностей. В классическом алгоритме каждая запись в этой таблице представляет собой упорядоченную пару значений, соответствующих эквивалентным меткам. Затем при помощи построения транзитивного замыкания множества эквивалентностей, обнаруженных на первом шаге, выделяются классы эквивалентности. Данная процедура является фазой анализа. Каждому классу эквивалентности присваивается уникальная метка, обычно значение минимальной метки в классе. На втором проходе, называемом фазой окончательной разметки, каждая промежуточная метка заменяется меткой соответствующего класса эквивалентности.

Серьезной проблемой при использовании классического алгоритма является размер таблицы эквивалентностей, которая может иметь очень большой размер при обработке широкоформатных изображений. В предлагаемой нами реализации классического алгоритма таблица эквивалентностей представляет собой хэш-таблицу, реализованную методом цепочек. Данный алгоритм будем называть TPC (Two pass classic).

Одним из решений проблемы неконтролируемого роста размера глобальной таблицы эквивалентностей заключается в использовании небольшой локальной таблицы эквивалентностей, хранящей информацию о метках, принадлежащих двум скан-строкам изображения – текущей и предыдущей. Следовательно, максимальное число отношений эквивалентности равно числу столбцов матрицы изображения. Каждая строка матрицы изображения последовательно сканируется в течение двух итераций. На первой итерации происходит заполнение локальной таблицы эквивалентностей. Затем производится анализ таблицы эквивалентностей, определяются метки-представители, которые будут присвоены пикселям текущей строки при повторном скан-проходе (вторая итерация). Следовательно, в дальнейшем анализе будут использоваться обновленные значения меток. Следует отметить, что второй скан-проход осуществляется в направлении, противоположном первому. При применении данной техники будем добавлять суффикс _LM к названию алгоритма.

Эффективным методом, позволяющим осуществить компактное хранение информации об эквивалентности и быстрый поиск, является использование структуры данных для объединения-поиска [9] (union-find). Алгоритм объединения-поиска, который динамически строит классы эквивалентности после обнаружения связанных отношением эквивалентности объектов, стал широко использоваться в различных прикладных задачах. Структура данных, ориентированная на объединение и поиск, позволяет эффективно формировать классы эквивалентности, представленные в виде древовидных структур данных и выполнять операции над ними. Добавление этой структуры данных в алгоритм маркировки оказалось полезным усовершенствованием классического алгоритма поиска связных компонент на графах.

Опишем структуру данных для объединения-поиска. Каждое непересекающееся множество хранится в форме древовидной структуры, в каждом узле которой хранится метка и ссылка на один родительский узел. Как правило, метка, присвоенная корневому элементу дерева, выбирается в качестве окончательной метки для всех промежуточных меток, входящих в дерево. Процедура поиска принимает в качестве параметра метку *X,* прослеживает родительские узлы вверх к

корню дерева и находит метку корневого узла дерева, которому принадлежит *X*. Процедуре объединения передаются две метки – *X* и *Y*. Для этих меток находятся корневые узлы. Если корневые узлы не совпадают, то узел одной из меток назначается родительским узлом для узла второй метки.

Традиционным способом для представления деревьев является использование указателей. При использовании динамического выделения памяти узлы дерева размещаются в памяти в произвольном порядке. Следовательно, процедура поиска будет обращаться к памяти неэффективно. Многие авторы предлагают хранение дерева при помощи массива, так как ячейки массива размещаются в смежных ячейках памяти.

Если глубина дерева является большой, то поиск корня для листов будет медленным. Поэтому может применяться компрессия пути, т.е. при добавлении узла *T*, имеющего корень *S*, к новому корню *C* для всех узлов, принадлежащих пути от *T* до *S* устанавливается корень *C*. Реализацию классического алгоритма с применением структуры данных для объединения поиска назовем TPC_UF, реализацию с применением компрессии назовем TPC_UF_PC.

Таблица 2. Производительность двухпроходных алгоритмов

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
TPC	1234	175868	26204	45954
TPC_UF	244	5312	1070	1308
TPC_UF_LM	292	1000	509	200
TPC_UF_PC	147	624	302	138
UF_LM_PC	290	991	496	203

5. ОДНОПРОХОДНЫЕ АЛГОРИТМЫ

Наиболее простыми методами маркировки связных компонент являются рекурсивные алгоритмы поиска. Процесс поиска сводится к следующим операциям. Находится непомеченный черный пиксель. Ему присваевается новая метка и вызывается процедура поиска *search* всех черных необработанных соседей. Для каждого из найденных соседей производится рекурсивный вызов *search.* Очевидно, что для избежания множественных рекурсивных вызовов поиск в ширину реализуется с помощью структуры данных очередь. Изначально в очередь заносятся координаты исходного черного пикселя. Затем из очереди извлекается элемент, он помечается текущей меткой СК, находятся все непомеченные соседи и помещаются в очередь. Обработка продолжается до тех пор, пока очередь не опустеет. Данная стратегия носит название поиск в ширину (Breadth-first search, BFS). В качестве структуры данных для хранения информации о подлежащих обработке пикселях может использоваться стек. В таком случае будет осуществляться поиск в глубину (Depth-first search, DFS).

Наиболее эффективным алгоритмом данного семейства является алгоритм обхода контуров (CT), предложенный в работе [10]. Алгоритм осуществляет проход по изображению, при встрече непомеченного объектного пикселя проверяется его принадлежность внутренней либо внешней границе объекта. Если пиксель является граничным, то создается новая метка и осуществляется прослеживание контура. Все контурные точки получают значение созданной метки. В противном случае соседний слева пиксель является уже помеченным, текущий пиксель получает его метку.

Таблица 3. Производительность однопроходных алгоритмов

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
DFS	159	663	318	147
BFS	170	686	325	154
CT	127	340	192	62

6. РЕАЛИЗАЦИЯ НА СЖАТОМ РАСТРЕ

Эффективным методом представления изображения является кодирование длин серий (RLE). Многие алгоритмы нахождения СК могут быть реализованы на сжатом растре [11]. Каждая серия получает дополнительное поле для хранения метки СК. При работе на сжатом растре количество обращений к объектам изображения значительно уменьшается, т. к. каждая серия является группой пикселей и нет необходимости обрабатывать каждый объектный пиксель в отдельности. Более того, фоновые пиксели вообще не участвуют в обработке. Количество промежуточных меток также сокращается, следовательно, уменьшается время, требуемое на разрешение эквивалентностей.

В работе [2] авторы утверждают, что их алгоритм является эффективным при обработке штриховых изображений. На первом проходе анализируется смежность серий в *i* и *i*-1 скан-строке. Если серия не имеет смежных, ей присваивается новая метка *k*, которая заносится в позицию *k* массива *R* меток-представителей. Пусть две метки *A* и *B* принадлежат одной СК, т. е. являются эквивалентными. Тогда всем ячейкам массива *R*, имеющим значение *B* присваивается значение *A*. На втором проходе каждая серия получает значение метки из массива *R*. Данный алгоритм назовем по именам авторов – HCSW.

Нами предлагается следующая модификация классического двухпроходного алгоритма. На первом проходе сохраняется информация об эквивалентности серий. Предлагается использование следующей структуры данных. Каждая серия конфигурируется не числовым идентификатором метки, а указателем на структуру метки, которая содержит числовой идентификатор, являющийся одновременно и порядковым номером СК. Эквивалентные метки имеют одинаковое значение идентификаторов и хранятся с помощью однонаправленного списка.

Списки структур меток размещаются в ячейках таблицы эквивалентностей, которая является одномерным массивом, причем индекс в таблице для каждого списка равен идентификатору метки.

Если рассматриваемая серия не имеет смежных, то создается новый список, содержащий одну структуру метки со значением *k*, он помещается в *k*-ю позицию таблицы. Затем значение *k* увеличивается на единицу.

Пусть две серии являются смежными и имеют разные числовые значения идентификаторов в структуре метки. Пусть *a* и *b* - значения идентификаторов, причем $a < b$. Все идентификаторы, находящиеся в позиции *b* таблицы имеют значение *b*. Для всех элементов списка изменим это значение на *a* и переместим данный список в конец списка, находящегося в позиции *a*. Чтобы объединение списков осуществлялось быстро, мы для каждого списка храним

указатель на последний элемент. Благодаря этому избегается полный просмотр списка с целью нахождения последнего элемента. После объединения списков сдвинем все ячейки таблицы эквивалентностей на одну позицию вниз, начиная с позиции $b + 1$. Для каждого сдвигаемого списка изменим значение идентификатора метки, т.е. сделаем его равным значению позиции в таблице.

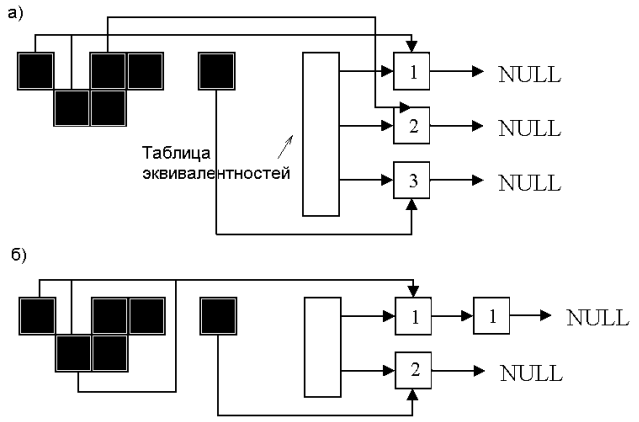


Рисунок 5: Пример работы алгоритма: а) обработка первой скан-строки; б) обработка второй скан-строки.

Второй проход алгоритма является фазой финальной маркировки. Пиксели каждой серии добавляются к некоторой связной компоненте в соответствии с порядковым идентификатором метки. Заметим, что в данном алгоритме отсутствует отдельная фаза нахождения финальных значений меток, т. к. данные действия выполняются на первом шаге алгоритма.

Назовем предлагаемую модификацию классического алгоритма CL_MOD. Приведем результаты тестирования производительности алгоритмов выделения СК на сжатом растре в таблице 4.

Таблица 4. Производительность алгоритмов на сжатом растре

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
MP	12	1050	300	410
MP_CT	29	187	74	48
CL_MOD	4	28	15	6
UF	10	489	119	126
UF_PC	6	23	12	5
HCSW	6	33	15	7

7. МЕТОДИКА «РАЗДЕЛЯЙ И ВЛАСТВУЙ»

Известной техникой, применяемой при обработке изображений большого размера, является методика "Разделяй и властвуй", которая позволяет разбить исходную задачу на подзадачи меньшей размерности. В случае с выделением СК изображение разбивается на N прямоугольных областей.

Для каждой области выполняется выделение СК традиционным способом (например, классическим алгоритмом). Затем найденные компоненты объединяются, т. е. происходит "сшивка" областей.

Проведенные нами эксперименты показали, что применение данного подхода позволяет значительно увеличить производительность методов попиксельного анализа, использующих глобальную таблицу для разрешения эквивалентностей, в частности, классического алгоритма[8].

Таблица 5. Производительность некоторых алгоритмов с использованием техники «разделяй и властвуй»

Метод	Min, мс	Max, мс	Avg, мс	Disp, мс
MP	423	4298	1865	1341
MP_CT	444	1476	930	359
TPC	172	2409	827	688
TPC_UF	138	625	265	142
TPC_UF_LM	211	665	343	116
TPC_UF_PC	147	624	302	138
UF_LM_PC	290	991	496	203

8. ЗАКЛЮЧЕНИЕ

В данной работе классифицируются алгоритмы выделения связных компонент, описываются особенности их программной реализации. Проведенные нами эксперименты показали, что оптимальными алгоритмами для выделения СК на штриховых изображениях являются алгоритмы, использующие сжатое представление изображения. Предлагается модификация классического алгоритма разметки с использованием сжатого растра. Предложенный алгоритм имеет высокую производительность, прост в реализации и эффективен при обработке широкоформатных изображений. Рассматривается методика «разделяй и властвуй» как эффективное средство увеличения производительности алгоритмов выделения СК, использующих попиксельный анализ.

9. ЛИТЕРАТУРА

[1] Л. Шапиро, Дж. Стокман, *Компьютерное зрение*, М.: Бином. Лаборатория знаний, 2006.

[2] L. He, Y. Chao, K. Suzuki , K. Wu, “Fast connected-component labeling“, *Pattern Recognition*, Vol. 42, No. 9., 2009. – pp. 1977-1987.

[3] H.Hedberg, F.Kristensen, and V.Owall, “Implementation of a labeling algorithm based on contour tracing with feature extraction", *IEEE International Symposium on Circuits and Systems(ISCAS)*, 2007, pp. 1101–1104.

[4] С. В. Абламейко, Д. М. Лагуновский, Обработка изображений: технология, методы, применения, Амалфея, Минск, 2000.

[5] K. Wu, E. Otoo, and K. Suzuki, “Optimizing two-pass connected-component labeling algorithms”, *Pattern Analysis & Applications*, 2009, Vol. 12, No. 2., pp. 206–220.

[6] R.M. Haralick, Some Neighborhood Operations, *Real Time/Parallel Computing Image Analysis*, Plenum Press, New York, 1981.

[7] K. Suzuki, I. Horiba, N. Sugie, “Linear-time connected-component labeling based on sequential local operations“, *Comput. Vis. Image Underst.*, Vol. 89, No. 1, 2003, pp. 1–23.

[8] A. Rosenfeld, P. Pfaltz, “Sequential Operations in Digital Picture Processing”, *Journal of the Association for Computing Machinery*, Vol. 12., 1966.

[9] C. Fiorio, J. Gustedt, “Two linear time union-find strategies for image processing”, *Theor. Comput. Sci.*, Vol. 154, No. 2, 1996, pp. 165-181.

[10] F. Chang, C.-J. Chen, and C.-J. Lu, “A linear-time component-labeling algorithm using contour tracing technique,” Comput. Vis. Image Underst., vol. 93, no. 2, 2004, pp.206–220.

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

Благодарности

Автор выражает благодарность своим родителям за поддержку, своему другу Ивану Байдакову за создание тестовой среды и проведение тестов, и Борису Борисовичу Гребенщикову за неповторимую атмосферу, создаваемую его музыкой.

Об авторе

Максим Стержанов – аспирант БГУИР. email: accept@bk.ru.

Abstract

In this paper we analyze a problem of connected component labeling for monochrome line drawings. We provide both a classification and description of the existing algorithms. Some optimization methods are also considered, with experimental measurements and comparative analysis of algorithms being provided. We also present a modification of the classical algorithm for connected component labeling which utilises a compressed representation of the raster image.

Keywords: *binary raster, connected component, line image.*