УДК 004.932.1

# РЕАЛИЗАЦИЯ ИНТЕРАКТИВНОЙ СЕГМЕНТАЦИИ ДЛЯ СЕНСОРНЫХ УСТРОЙСТВ НА БАЗЕ ОС ANDROID

**Круглов А.В., Югфельд И.Д.**

*ФГАОУ ВПО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина», Екатеринбург, e-mail: avkruglov@yandex.ru*

В данной статье приведены особенности реализации алгоритма интерактивной сегментации «волшебная палочка» применительно к мобильным устройствам. Наиболее подходящий под условия ограниченной аппаратной производительности метод был получен при помощи модификации канонической реализации алгоритма с использованием рекурсии. Измененный алгоритм интерактивной сегментации построен на основе волнового алгоритма. С целью дальнейшей оптимизации исследуемого метода проведен ряд дополнительных модификаций, обусловленных спецификой предметной области – задачей выделения торцов на изображениях штабелей бревен. В результате проведенной работы получена программная реализация алгоритма интерактивной сегментации «волшебная палочка» для мобильных устройств на базе ОС Android. Данный программный модуль используется в приложении по учету круглого леса для решения задачи ручного выделения торцов бревен.

**Ключевые слова: интерактивная сегментация, волшебная палочка, ОС Android, волновой алгоритм, распознавание образов, выделение объектов, учет лесоматериалов**

# IMPLEMENTATION OF THE INTERACTIVE SEGMENTATION FOR SENSOR DEVICES BASED ON OS ANDROID

**Kruglov A.V., Yugfeld I.D.**

*Ural Federal University named after the first President of Russia B.N. Yeltsin, Ekaterinburg, e-mail: avkruglov@yandex.ru*

This paper presents the features of the implementation of the interactive segmentation algorithm called «magic wand» for mobile devices. The most appropriate method for the conditions of the limited hardware performance was obtained by modifying the canonical recursive realization of the algorithm. Modified algorithm for interactive segmentation is based on the wave algorithm. For the further optimization a number of additional modifications of the algorithm caused by the specifics of the subject area – the problem of detection of the head ends of the logs – are implemented. As a result of this work, the software implementation of interactive segmentation algorithm «magic wand» for mobile devices under OS Android was produced. This software module is used in the accounting logs application to solve the problem of manual detection of the head ends of the logs.

**Keywords: interactive segmentation, «magic wand» algorithm, OS Android, wave algorithm, pattern recognition, object selection, individual log tally**

При обработке изображение зачастую рассматривается не как однородная сцена, а как совокупность объектов переднего плана и фона изображения [4]. В общем случае при работе с тем или иным изображением часто возникает необходимость отделить одну, значимую для пользователя часть, которая интересует его в данный момент (объект), от всего остального (фон). Алгоритм интерактивной сегментации, известный также как «волшебная палочка», позволяет это сделать. Его особенностью является то, что пользователь может самостоятельно простыми действиями определить, что является для него целевым объектом, а что относится к фону.

Наибольшее применение данный инструмент нашел в растровых графических редакторах. Тем не менее существует целый класс промышленных программ, использующих в своей основе методы обработки и анализа изображений. В этом случае предъявляются повышенные требования как к программной реализации алгоритма – высокая надежность и точность работы, так и к аппаратной платформе, на которой предстоит работать. Так, например, все большее распространение получают мобильные системы контроля и учета.

В частности, в лесной отрасли задача учета круглого леса может быть решена посредством анализа изображений торцов штабелей бревен. Инструмент пользовательского редактирования на основе интерактивной сегментации необходим для оперативной корректировки результатов автоматического выделения. В данной статье будут рассмотрены особенности реализации данного алгоритма с учетом аппаратных требований к системе учета – мобильные (сенсорные) устройства.

**Идея алгоритма**

Пользователь указывает точку внутри выделяемого объекта на изображении. Алгоритм определяет цвет выбранного пикселя и анализирует соседние точки на предмет совпадения цветов. По причине воздействия на объект света, теней, а также неоднородной структуры самого объекта его цвет не однороден. Следовательно, необходимо задать максимально допустимое отклонение цвета, при помощи которого и определяется принадлежность пикселя детектируемому объекту.

**Реализация алгоритма**

Алгоритм определения нужных пикселей является алгоритмом закрашивания замкнутых областей. Основная идея заключается в получении исходной точки (стартовой) и анализе соседних точек из 4-связной области на предмет похожести. Если точка удовлетворяет условию, то сохраняем ее в маске. Псевдокод для этого алгоритма:

```
void select(float x, float y, Color colorToMatch){
        Color color = GetColor(x, y);
        if ( colorsSimilar(color, colorToMatch) ) {
                setPixelInMask(x, y);
                select(x - 1, y, colorToMatch);
                select(x + 1, y, colorToMatch);
                select(x, y - 1, colorToMatch);
                select(x, y + 1, colorToMatch);
        }
}
```

В таком виде алгоритм предполагает применение рекурсии. Рекурсия – весьма полезный инструмент, но его применение подразумевает использование практически не контролируемого ресурса: стека вызовов. А учитывая, что параметры картинки (высота и ширина в пикселах), а также площадь выделяемого объекта в пикселах могут быть заранее не известными, то применение рекурсии может с большой вероятностью привести к переполнению стека вызовов или захватить слишком много оперативной памяти. По причине критических требований к надежности работы применение рекурсивного метода в данном случае нежелательно. Альтернативный вариант реализации дает идея волнового алгоритма.

Волновой алгоритм, как правило, используется для поиска кратчайшего пути от одной точки к другой. Суть волнового алгоритма состоит в следующем:

1. Имеется поле размерностью MxN, начальная точка *A* и конечная точка *B*. Каждая точка поля имеет свойство проходимости (она может быть проходима или непроходима). Требуется найти кратчайший путь из точки *A* в точку *B*.

2. Из начального элемента (точка *A*) распространяется в 4-х направлениях волна [5], как показано на рис. 1. Элемент, в который пришла волна, образует фронт волны. На рисунках цифрами обозначены номера фронтов волны. Каждый элемент первого фронта волны является источником вторичной волны. Элементы второго фронта волны генерируют волну третьего фронта и т.д. Процесс продолжается до тех пор, пока не будет достигнут конечный элемент. Ну, или пока не станет ясно, что его не достигнуть.

3. Строится сама трасса. Её построение осуществляется от конечного элемента к начальному.

В данном случае технология построения трассы (самого кратчайшего пути) рассматриваться не будет ввиду того, что для задачи реализации интерактивной сегментации «волшебная палочка» трасса не нужна.

Так как в текущем случае не нужно искать наименьшее расстояние от начальной точки, до конечной, то нет смысла цифрами с расстоянием помечать точки (пиксели). Вместо этого пиксель отмечается как удовлетворяющий/неудовлетворяющий критерию вхождения в выделяемую область. Алгоритм останавливается, когда вокруг выделяемого объекта образуется замкнутая рамка из пикселей, не отвечающих установленному критерию.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 12 | 11 | 12 | | | | | | | | | | | | |
| | | | | 12 | 11 | 10 | | | | 12 | | 12 | | | | | | | |
| | | **B** | 12 | 11 | | 9 | 10 | 11 | 12 | 11 | | 11 | | | | | | | |
| | | 12 | 11 | 10 | 9 | 8 | 9 | 10 | | 10 | 11 | 10 | 11 | 12 | | | | | |
| | 12 | | 10 | 9 | | 7 | | 9 | 8 | 9 | | 9 | | 11 | 12 | | | | |
| 12 | 11 | 10 | | 8 | 7 | 6 | 7 | | 7 | | 7 | 8 | 9 | 10 | | | | | |
| | 10 | 9 | 8 | 9 | | 5 | 6 | | 6 | 5 | 6 | 7 | 8 | 11 | 12 | | | | |
| 10 | | 8 | 7 | | 5 | 4 | | 6 | 5 | 4 | 5 | 6 | 7 | | 11 | 12 | | | |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | | 3 | 4 | | 8 | 9 | 10 | | | | |
| 10 | 9 | 8 | 7 | 6 | 5 | | 1 | **A** | 1 | 2 | 3 | | 7 | 8 | 9 | | | | |
| | 10 | | | 7 | 6 | 3 | 2 | 1 | 2 | | 4 | 5 | 6 | | 10 | 11 | 12 | | |
| 12 | 11 | 10 | 9 | 8 | | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | |
| | 12 | | 8 | 7 | 6 | 5 | 4 | | 4 | 5 | | 7 | | 9 | 10 | 11 | 12 | | |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 6 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | | |
| | 12 | | 10 | | 8 | | 6 | 7 | | 7 | 8 | 9 | 10 | 11 | 12 | | | | |
| | | 12 | 11 | 10 | 9 | | 7 | 8 | 9 | | 9 | 10 | | 12 | | | | | |
| | | | 12 | 11 | 10 | 11 | | 9 | 10 | 11 | 10 | 11 | 12 | | | | | | |
| | | | | 12 | 11 | 12 | 11 | 10 | | 12 | | 12 | | | | | | | |
| | | | | | 12 | | 12 | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |

*Рис. 1. Иллюстрация работы волнового алгоритма*

Псевдокод алгоритма основной части:

```
mask.put(p, (byte) 2);
next.add(p);
while (!next.isEmpty()){
        replaceQueuees();
        for( i=0;i<current.size();i++ ){
                checkNeighbors ( current.poll() );
        }
}
```

current, next – очереди из точек текущего и следующего уровней соответственно (На рис. 1, если в очереди current находятся точки с пометкой 1, то в очереди next находятся точки с пометкой 2 и так далее). Псевдокод метода проверки соседних точек:

```
void checkNeighbors (Point p){
                checkForSimilarity( p, new Point(p.x, p.y-1) );
                checkForSimilarity( p, new Point(p.x, p.y+1) );
                checkForSimilarity( p, new Point(p.x-1, p.y) );
                checkForSimilarity( p, new Point(p.x+1, p.y) );
}
```

Здесь проверяются 4 соседних пикселя (верхний, нижний, левый, правый) на схожесть с текущим пикселем.

Псевдокод метода проверки пикселей на схожесть:

```
void checkForSimilarity( Point startPoint, Point checkPoint ){
        if ( isPixelExists(checkPoint) ){
                if ( areColorsSimilar( img.getPixel(startPoint.x, startPoint.y),
                                        img.getPixel(checkPoint.x, checkPoint.y) ) ){
                        mask.put( checkPoint, SIMILAR );
                        next.add( checkPoint );
                }
                else mask.put( checkPoint, NOT_SIMILAR );
        }
}
```

Метод isPixelExists(checkPoint) проверяет не существуют ли пиксели с такими координатами, то есть не произошел ли выход за пределы изображения.

mask – структура, в которую добавляются проверенные пиксели с пометкой вхождения/не вхождения в результирующую выделяемую область.

Псевдокод метода проверки пикселей на схожесть по цвету:

```
boolean areColorsSimilar(int c1, int c2){
int differenceR = Abs( c1.red -c2.red );
        int differenceG = Abs( c1.green -c2.green );
        int differenceB = Abs( c1.blue -c2.blue );
        int maxDifference = Max( differenceR, differenceG, differenceB );
        return maxDifference < step ? true : false;
}
```

step – чувствительность алгоритма.

Чтобы получить список пикселей, входящих в результирующую область, нужно из mask выбрать только те пиксели, которые помечены как удовлетворяющие критерию схожести.

Описанный алгоритм был применен в мобильном приложении на базе ОС Android для решения задачи детектирования торцов бревен.

**Модификации алгоритма**

*1. Настройка чувствительности*

Величина параметра чувствительность (разница между численными значениями цветовых компонент пикселей) настраивается исходя из особенностей изображений, подаваемых на вход алгоритма. Если картинка содержит детектируемые объекты, являющиеся однородными по цвету, то чувствительность может быть нулевой, это самый простой случай. Если же на вход подается изображение с неоднородным цветом объекта, то чувствительность настраивается исходя из величины его цветового диапазона.

Настройка чувствительности – очень тонкий момент, так как при высокой чувствительности алгоритм может не распознать объект целиком, а при малом значении этого параметра могут попасть в выделяемую область пиксели, которые на самом деле не принадлежат объекту. Регулирование этого параметра сильно зависит от предметной области.

В задаче детектирования торцов бревен опытным путем выявлено значение чувствительности равное семи.

*2. Аппроксимация*

При невозможности определения идеальной чувствительности улучшить показатели точности распознавания алгоритма

также может проведение аппроксимации, если заранее известна форма детектируемого объекта. На выходе алгоритма образуется список координат пикселей, вошедших в детектируемый объект, в зависимости от формы детектируемого объекта его можно аппроксимировать в круг, квадрат, треугольник и другие фигуры.

В задаче детектирования торцов бревен выделяемая область аппроксимировалась в эллипс. Диаметры овалов вычислялись по формулам maxX – minX и maxY – minY, где maxX, maxY – наибольшие координаты по X и У соответственно; minX, minY – наименьшие координаты по X и У соответственно.

*3. Задание минимального и максимального размеров объекта*

Если исходя из условий решаемой задачи возможно оценить минимальный и максимальный размер целевого объекта по отношению к размеру изображения, это позволит значительно оптимизировать работу алгоритма. В качестве критериев могут выступать такие метрики, как площадь объекта по отношению к площади изображения в пикселах, соотношение высоты объекта или ширины и изображения и т.д.

В задаче детектирования торцов бревен установлен минимально допустимый диаметр детектируемого объекта – овала, зависящий от размера изображения, и он составляет 1/20 часть от среднего арифметического высоты и ширины картинки в пикселях. Также установлена максимально допустимая площадь выделяемого торца бревна, она составляет 1/125 часть от площади всего изображения. Значения этих ограничений получены путем анализа набора тестовых изображений.

*4. Способ определения схожести*

В зависимости от особенностей обрабатываемых изображений можно выбирать различные методы определения попадания

*Рис. 2. Скриншот экрана приложения*

пиксела в выделяемую область. В приведенном примере схожесть определялась исходя из определения максимального отклонения среди трех компонентов цветовой модели RGB и сравнения с заранее установленной чувствительностью. Для сравнения можно использовать другие цветовые модели или сравнивать отклонения не по всем компонентам, а только по одной и т.д.

В задаче детектирования торцов круглых бревен для сравнения использовались все компоненты модели RGB, так как другие цветовые модели не показали лучших результатов.

## Результат работы

На рис. 2 представлен скриншот окна мобильного приложения для детектирования торцов бревен [2], на котором продемонстрирован результат работы алгоритма интерактивной сегментации.

## Выводы

Модификация «волшебной палочки» с использованием волнового алгоритма позволила устранить рекурсию. Отказ от рекурсии исключил возможность переполнения стека вызовов, а в Android по умолчанию его размер составляет 8 Кбайт (около 260 вызовов), то есть алгоритм с рекурсией будет способен распознать объект с приближенной площадью не более 260 пикселей, что не удовлетворяет условиям задачи выделения торцов бревен на изображении. Таким образом, созданный алгоритм является более предсказуемым и стабильным, нежели рекурсивная «волшебная палочка». Возможности модификации алгоритма с учетом предметной области также повышают точность распознавания.

## Список литературы

1. Кристофидес Н.. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 стр.

2. Круглов А.В., Югфельд И.Д.. Проектирование базы данных для приложения расчета кубатуры круглого леса под управлением платформы Android. – Программные системы и вычислительные методы. — 2014. – № 4. – С. 431–436.

3. Обход препятствий: волновой алгоритм (Алгоритм Ли) [Электронный ресурс]. – Режим доступа: http://suvitruf.ru/2012/05/13/1176.

4. Хропов А.А., Карпов А.С., Лемпицкий В.С., Иванов Д.В., Кузьмин Е.П.. Алгоритмические основы растровой графики [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/993/163/info.

5. Ярославский Л.П.. Введение в цифровую обработку изображений. – М.: Сов. радио, 1979. – 312 с.

6. Andy Finnell / How to implement a magic tool [Электронный ресурс]. – Режим доступа: http://www.losingfight.com/blog/2007/08/28/how-to-implement-a-magic--tool.