**Нахождение связных областей на изображениях**

**Введение**

При работе с изображениями, при анализе их содержимого нередко возникают задачи выделения на исходных изображениях непрерывных участков с близкими по некоторым свойствам наборами точек.

В этой статье мы осуществим небольшое исследование уже готовых способов нахождения связанных участков (областей), а так же изучим дополненный и улучшенный алгоритм роста регионов.

**Основные понятия**

Связанная область (участок) – набор точек на изображении, в котором любые две точки соединены друг с другом через последовательность соседей.

Виды связности:

1. 4-связность - соседями точки считаются точки, имеющие с данной точкой одну общую сторону.

http://www.mallenom.ru/images/article004_1.jpg

**Рис 1.«4-связность»**

Где С – сосед, Т – текущая (данная) точка.

1. 8-связность – соседями точки считаются точки, имеющие с данной точкой общую сторону или общий угол.

http://www.mallenom.ru/images/article004_2.jpg

**Рис 2. «8-связность»**

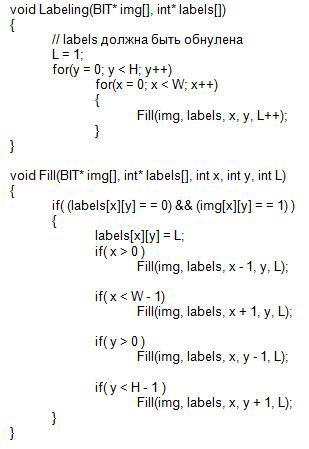
Где С – сосед, Т – текущая (данная) точка.

**Бинарные изображения**

Несмотря на то, что целевыми изображениями данной статьи являются цветные и полутоновые, все-таки стоит изучить так же и упрощенные - черно-белые (бинарные), поскольку к ним сходятся некоторые задачи анализа цветных изображений.

Допустим, мы имеем черно-белое изображение. Таковым будем называть изображение, состоящее из точек только двух цветов: цвета фона и цвета объектов. Чаще всего обозначают 0 – фоновый цвет, 1 – цвет объекта. В таком случае выделенные области всегда будут одинаковыми при неизменном типе связности (4- или 8- связном). В основном используют два алгоритма выделения областей: рекурсивный и циклический (итеративный).

**Рекурсивный алгоритм:**

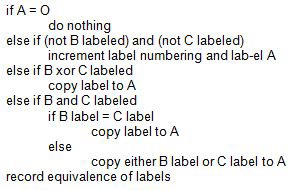


Минусами данного алгоритма являются низкая скорость работы и высокий потребляемый объем памяти. Другой подход – циклический или итеративный, в литературе часто именуется «алгоритм последовательного сканирования». Его тоже следует изучить.

**Циклический алгоритм:**

Обход по изображению выполняется из верхнего левого угла. На внешнем цикле – перебор строк, на внутреннем – перебор столбцов строки. Точки с цветом фона опускаем. Нетрудно заметить, что при таком подходе всегда уже обработаны соседи сверху и слева.





После обхода всего изображения, следует его повторить и осуществить повторное выделение областей, учитывающее эквивалентности областей.

**Цветные и серые изображения**

Множество простых методов реализует такое решение: многоцветное изображение конвертируется в двухцветное изображение простыми способами. При этом рассматриваются цвет или яркость точек. После преобразования к изображению применяются рассмотренные выше, либо аналогичные алгоритмы.

Методы, которые используют определенные пользователем параметры (порог яркости, пороги группировки цвета) являются плохими из-за слабой гибкости. Если меняется яркость/контраст/цветовой баланс, то все параметры приходится переопределять, причем вручную. Намного более применимы алгоритмы, изменяющие эти параметры в зависимости от статистики анализируемого изображения. Они намного более независимы от применения разных корректировок окраски.

**Алгоритм k-средних**

В общем виде его псевдокод выглядит так:

*Исходные данные:*

* Множество цветовXi, i = 1..K (для каждой из точек);
* N – Количество групп, на которые алгоритм поделит множество.

*Результат:*

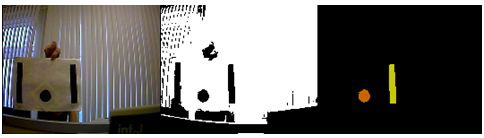
* N усредненных цветов точек Mj, j = 1..N (средний цвет каждой из групп);
* отнесение каждой из точек Xi в какую-либо из N групп.

*Ход решения:*

1. Случайным образом выбрать N средних цветов групп Mj, j = 1..N.
2. Составить таблицу расстояний от цвета каждой точки Xi до каждого среднего цвета групп Mj. Каждую из точек Xi отнести к какой-либо группе по критерию минимума расстояния до цвета Mj этой группы.
3. Пересчитать средние цвета Mj всех групп.
4. Повторять шаги 2, 3 пока цвета групп не установятся в неизменяющееся значение.

Допустим, мы применяем этот метод к одномерному набору точек, и количество групп выбираем равное 2. После окончания работы алгоритма точки одной из групп закрашиваем белым, другой – черным. Таким образом, мы выполняем этот алгоритм для нахождения светлых объектов на темном фоне, либо наоборот. На примере ниже – темные объекты в естественном окружении.

Основные преимущества такого метода – высокая скорость и несложная реализация. Но не всегда он приводит нас к хорошему решению: в данном примере одна из черных полос в ходе алгоритма превратилась в фон.



Оставшиеся алгоритмы на нахождения связанных областей условно разделяют на два множества:

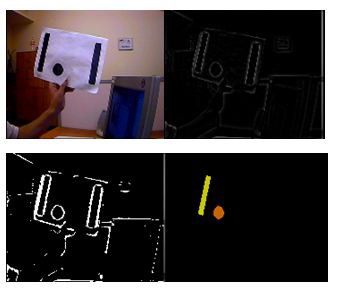
* Анализирующие перепады яркости.
* Осуществляющие прямой поиск областей.

**Алгоритмы, анализирующие перепады яркости (границы).**

Это множество методов осуществляет поиск областей по границам – областям с высокими перепадами яркости на изображении. Порядок работы для них всех схож:

1. Выделение областей с большими перепадами яркости («границ»).
2. Поиск областей по выделенным границам.

Теперь повторим поиск решения той же задачи, что и в предыдущем случае (черные объекты в естественном окружении), но уже с использованием новых алгоритмов.



Выше приведены изображения, показывающие работу одного из методов на основе анализа границ. С помощью фильтра осуществляется поиск зон с перепадами яркости. Производится бинаризация (алгоритмом k-средних). Поиск связанных областей осуществляется методом последовательного сканирования.

Эта задача отлично подходит для описания плюсов и минусов такого способа. Он намного более гибкий, нежели обычное преобразование изображения к черно-белому, поскольку мы имеем возможность регулирования чувствительности метода (меняя настройки фильтра). Метод практически независим от изменения параметров изображения (контраст, яркость, шумы и т.д.), поскольку использует не конкретные значения цвета точек, а перепады (высокие частоты).

Но есть и минусы:

* Низкая устойчивость: маленький разрыв границ способен привести нас к несрабатыванию алгоритма. Анализируя результат с помощью методов математической морфологии, мы можем сгладить ошибку, но при этом начинает играть роль второй минус.
* Низкая точность. Граница может занимать большую долю описываемого ею объекта, но при этом ее пространство может быть отнесено к фону.
* Низкая скорость работы – каждый из трёх этапом является медленным, суммарное время работы очень высоко.

Часть этих недостатков возникают из-за разбиения метода на этапы, ведь на каждом из них производится отброс лишней информации, но из-за неточностей каждого из них отбрасывается так же и полезная информация – возникает ошибка. По ходу работы эта ошибка накапливается и к концу алгоритма достигает значительного размера.

**Методы, основанные на прямом поиске областей.**

Это множество методов пробует выделить области непосредственно, а не косвенным путем, как предыдущие.

*Пусть:*

* A - изображение, (i, j) - координаты в A. A и пара (i, j) образуют множество наборов (A, i, j).
* f(A, i, j) > Rn - функция, которая переводит изображение A и координаты (i, j) в новое пространство Rn.
* x &isin Rn – элемент из Rn, а |v| - его размер (модуль).
* C – класс значений набора (A, i, j), аg(C) > Rn - функция, отображающая его в Rn.
* v1 = g(C), v2 = f(A, i, j), F(v1 - v2) - мера близости класса C и вектора (A, i, j).

Таким образом, С задает некую область в изображении A.

Предположим, что мы уже выделили некий класс С на изображении. В таком случае мы можем добавлять к нему соседние элементы, для которых функция L(f(A, i, j) - g(C))–мера расстояния - меньше выбранного порога.

Рассмотрим упрощенный случай – выделение по яркости:

* f(A, i, j) – яркость в точке i, j.
* g(C) - усредненная яркость точек класса.
* L(B) - есть модуль яркости (так как в данном случае пространство Rn получается одномерным).

В таком случае точку мы заносим в класс в том случае, когда ее яркость отличается от усредненной яркости по точкам этого класса на величину, меньшую порога. Такой набор заданных нами функций довольно хорошо применим, но в общем случае можно получить еще более хорошие результаты.

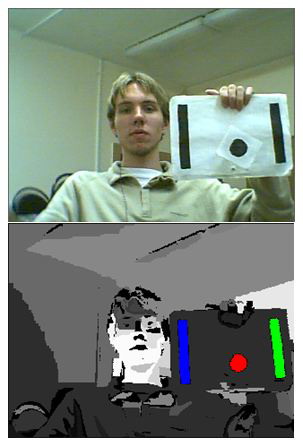
Если мы выберем более сложные пространства в качестве параметров алгоритма, например набор статистических параметров точки (разницу каждой из RGB-компонент по сравнению с соседями, дисперсию яркостей точки и соседей и т.п.). Кроме того, можно выбрать функцию модуля отличную от стандартной функции.

Уже существует алгоритм для простого случая разделения по яркости – алгоритм роста регионов (regiongrowing). Он, в принципе, несильно отличается от метода выделения областей обычного черно-белого изображения, но его легко можно обобщить на случай других функций f, g, Rn. Но важным минусом этого алгоритма является низкая скорость работы и огромное потребление памяти, порой приводящее к полному ее заполнению. Эта проблема решается итеративным обобщенным алгоритмом.

Начало алгоритма все также из верхнего левого угла:

1. Объявляем левый верхний пиксель изображения новым классом.
2. Q–порог разницы яркости между классами.
3. Рассматривая точки первого ряда, мы считаем отклонение от точки пикселя слева и, в зависимости от результата сравнения с порогом Q, либо заводим новый класс, либо заносим эту точку в класс соседа слева.
4. В каждом следующем ряду первый пиксель сравниваем с пикселем сверху (аналогично предыдущему пункту).
5. Для всех пикселей НЕ с первого ряда и НЕ с первого столбца – сравниваем с двумя пикселями, которые сверху и слева, вычисляем 2 отклонения L1 и L2 - от классов верхнего и левого пикселей соответственно.
6. Если оба отклонения больше порога – заводим новый класс.
7. Если одно из отклонений больше порога, а другое меньше – заносим эту точку в класс с отклонением, меньшим порога.
8. Если отклонение допустимо для обоих классов:
   * Если отклонение между этими классами меньше порога, то объединяем классы, и заносим точку в полученный класс.
   * Если отклонение между этими классами больше порога, то мы заносим точку в класс, отклонение от которого меньше.

Этот метод позволяет использовать различные отображения f, g, L и пространства, но при этом относительно них он равнозначен. Нетрудно увидеть, что метод рассматривает четырехсвязные области.



**Рис 3. В качестве пороговой яркости выбрана яркость 50 (при диапазоне 0..255)**

Плюсы такого алгоритма:

* Большой простор применения, поскольку можно выбирать самые различные отображения и пространства.
* Высокая скорость.
* Низкий потребляемый объем памяти.
* Высокоточная результативность.
* Изменением порога можно менять чувствительность метода.

В этом разделе стоит так же рассмотреть и другие методы, основанные на прямом поиске областей.

**Слияние регионов (Region Merging):**

Делим изображение на области по некоторому критерию. Затем начинаем постепенно сливать области, обладающие достаточной степенью схожести до тех пор, пока между всеми областями не будет низкая степень схожести. Его сложность в выборе первоначального разбиения и выборе критерия схожести.

**Разбиение и слияние регионов (Region splitting & merging):**

Разбиваем, как и в предыдущем случае, все изображение на области. Далее все разнородные области делим на однородные. Потом, как и в методе слияния регионов, начинаем сливать все области, но только в случае, если после этого сохраняется однородность полученной области. Его сложность наследует сложность предыдущего способа, добавляя проблему выбора критерия однородности.

**Метод "водораздела" (Watershed Segmentation):**

Работает только с полутоновыми изображениями. Все изображение представляется некоторой географической поверхностью – яркость каждой точки характеризует высоту ландшафта. После этого находятся области стабильных минимумов и области стабильных максимумов.