Математика и её приложения: ЖИМО. 2011. Вып. 1 (8). С. 151 – 156.

УДК 512.54

С. И. Хашин[1], Ю. А. Хашина[2]

Реализация алгоритма непрерывной сегментации изображений

Ключевые слова: распознавание образов, сегментация изображений

Описана реализация алгоритма "непрерывной сегментации" статических изображений. Указаны дополнительные этапы, позволяющие построить сегментацию с заданными свойствами. Приводится численное сравнение этого алгоритма сегментации с тремя другими алгоритмами.

Keywords: image recognition, image segmentation.

We describe the implementation of the algorithm of the "continuous segmentation" of static images. We introduce the additional steps that allow to build a segmentation with tailored properties. We present a numerical comparison of the segmentation algorithm with three other algorithms.

1. Введение

Сегментацией изображения будем называть его разбиение на однородные области (сегменты) по некоторому признаку. В настоящее время разработано множество различных методов сегментации изображений (см. напр., [1, 5]). Один из основных методов сегментации — выделение границ между сегментами на основе анализа градиентов функции яркости. Главным препятствием к построению сегментации изображения этим методом является то, что построенные границы не образуют замкнутых кривых, имеют большое количество разрывов, или даже просто состоят из отдельных незамкнутых кривых.

В работе [2] для решения этой проблемы предлагается метод, основанный на некотором приближении к R-функции от яркости.

Определение. Пусть $f(x,\, y)$ — гладкая класса C^2 функция в некоторой области на плоскости. Назовем ее *R-функцией* величину

$$R(x,\, y) = R_f(x,\, y) = f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}. \tag{1}$$

[1]Ивановский государственный университет; E-mail: khash2@mail.ru. Работа выполнена при финансовой поддержке РФФИ (проект 11-07-00653а).

[2]Ивановский государственный университет; E-mail: khashina_julia@mail.ru. Работа выполнена при финансовой поддержке РФФИ (проект 11-07-00653а).

Если ∇f — градиент функции f и S — квадрат градиента $S = f_x^2 + f_y^2$, то R_f можно записать в виде скалярного произведения:

$$R_f(x,\, y) = \frac{1}{2}(\nabla f,\, \nabla S).$$

Если f — функция одного переменного (т. е. $f(x, y) = f(x)$), то R_f обращается в нуль в точках перегиба функции $f(x)$. Если функция f достаточно гладкая, то уравнение $R_f = 0$ задает набор замкнутых кривых на плоскости, которые можно использовать для построения сегментации области определения функции f.

Если функция f задана таблично, например, своими значениями в целочисленных точках, то можно взять некоторую ее гладкую интерполяцию (класса не ниже C^2) и к ней применить формулу (1). Однако, построенная таким образом кривая оказывается чересчур сложной и плохо подходит для практических вычислений. Например, изображение, состоящее из одной белой точки на черном фоне сегментируется на 8 областей.

В работе [2] предлагается вместо этого вычислять значения первых и вторых частных производных функции f в целочисленных точках по обычным формулам

$$\begin{aligned}
f_x &\approx (f(x+1,\, y) - f(x-1,\, y))/2,\\
f_y &\approx (f(x,\, y+1) - f(x,\, y-1))/2,\\
f_{xx} &\approx f(x+1,\, y) - 2f(x,\, y) + f(x-1,\, y),\\
f_{xy} &\approx (f(x+1,\, y+1) - f(x+1,\, y-1) -\\
&\quad - f(x-1,\, y+1) + f(x-1,\, y-1))/4,\\
f_{yy} &\approx f(x,\, y+1) - 2f(x,\, y) + f(x,\, y-1),
\end{aligned} \tag{2}$$

и распространять их на всю плоскость с помощью билинейной интерполяции. Построенную таким образом функцию будем называть *r-функцией* таблично заданной функции f.

В работе [3] предложен алгоритм сегментации, который, после ряда упрощений, сводится к следующему.

Пусть f — таблично заданная функция яркости, т. е. просто матрица размера $mx \times my$ из целых чисел, принадлежащих отрезку [0, 255].

Шаг 1. Умножаем все значения в матрице f на 256 для уменьшения влияния ошибок округления в дальнейшем.

Шаг 2. Сглаживаем значения в матрице f с некоторым радиусом d.

Шаг 3. Во всех точках (x, y) находим величины f_x, f_y, f_{xx}, f_{xy}, f_{yy} по формулам (2) — аппроксимация частных производных.

Шаг 4. Задаем сегментацию s исходного прямоугольника размера $mx \times my$ на два сегмента (номера 1 и 2) по правилу: точку (x, y) относим к сегменту 1, если в ней величина $R(x, y)$, найденная по формуле (1), отрицательна, и к сегменту 2 — в противном случае.

Шаг 5. Каждый из полученных двух сегментов разбиваем на связные компоненты. Напомним, что сегмент называется *связным*, если от любой его точки можно добраться до любой другой, переходя каждый раз к одной из восьми соседних точек и оставаясь при этом внутри сегмента.

2. Результаты работы алгоритма

Рассмотрим для примера стандартное тестовое изображение *lena.bmp* размера $512 * 512$ точек ($256K$ точек). Приведем получающиеся размеры сегментов при различном радиусе сглаживания исходного изображения. В первой строке таблицы r — радиус сглаживания ($r = 0$ — без сглаживания), $s_1, \ldots, s_5$ — размеры наибольших 5-и сегментов, $n_4, \ldots, n_1$ — количество сегментов размеров 4, 3, 2, 1.

r:	0	1	2	4	8
s_1	123772	90439	68551	73848	92324
s_2	122458	79415	66924	30964	34508
s_3	2917	15716	20562	21652	20418
s_4	736	6945	15781	19648	13243
s_5	515	6637	5790	16382	10365
			...		
n_4	146	162	60	86	55
n_3	222	220	140	130	88
n_2	410	370	326	271	149
n_1	1112	1121	1203	766	369

Мы видим, что основной проблемой при использовании этого алгоритма является чересчур подробная детализация, т. е. слишком большое количество сегментов. Даже сглаживание с весьма большим радиусом не приводит к радикальному улучшению ситуации.

3. Дополнительные шаги, улучшающие алгоритм

Первые 4 шага составляют основу алгоритма и выполняются без изменений. После шага 4 все изображение оказывается разбито на два сегмента (номера 1 и 2). Для улучшения качества работы выполняем следующие дополнительные шаги.

Шаг 4а. *Морфологическое сглаживание с помощью клеточного автомата.* Если точка принадлежит одному сегменту, а не менее 6 ее соседей — другому, то относим и эту точку к другому сегменту.

Шаг 4б. *Удаление маленьких сегментов.* Если некоторая связная компонента одного из сегментов имеет размер меньше заданной границы $minSegmSize$, то относим ее к другому сегменту.

Величина $minSegmSize$ зависит от заданного количества сегментов N, ее можно положить равной $mx * my/(N * 50)$.

Шаг 4в. *Удаление сегментов с малым значением длины градиента на границе.* Если некоторое среднее значение длины градиента на границе некоторой связной компоненты одного из сегментов имеет размер меньше заданного предела, то относим эту связную компоненту к другому сегменту.

Предел задается в виде $\alpha * MGrad$, где $MGrad$ — среднее значение длины градиента по всему изображению а α — заданная константа.

После этих дополнительных шагов выполняем шаг 5, т. е. разбиваем имеющиеся два сегмента на связные компоненты.

Как показывает опыт, очень часто некоторые сегменты оказываются весьма причудливой формы, тянутся узкими полосами через все изображение. Такие сегменты следует разбить на более простые части. Для этого предлагается следующий алгоритм.

Шаг 5а. *Разбиение сложных сегментов на более простые части.* Каждый сегмент уже предполагается связным. Удалим из заданного сегмента все его граничные точки (т. е. переведем их в 0-й сегмент). Будем повторять эту процедуру до тех пор, пока оставшаяся часть сегмента не перестанет быть связной, либо количество точек сократится более, чем в два раза.

Если после такого удаления граничных точек сегмент остался связным, будем считать, что его не надо разрезать на части. В этом случае просто восстанавливаем сегмент в исходном виде.

Если же уменьшенный сегмент распался на несколько частей, то переводим эти части в отдельные сегменты. Нарастим затем каждый из новых сегментов за счет несегментированных точек, т. е. точек, удаленных из исходного сегмента в начале шага.

В результате работы этого алгоритма исходный сегмент разбивается на более простые части. Однако, каждая из этих частей все равно может оказаться слишком сложной. Процедуру дробления надо применить к каждой из полученных частей, до тех пор, пока алгоритм сможет разбивать их на части.

4. Базовые алгоритмы для работы с сегментами

Сегментация прямоугольника размера $mx \times my$ задается матрицей s того же размера. При этом число $s[x, y]$ задает номер сегмента, к которому относится точка (x, y). Номера сегментов принимают натуральные значения (т. е. целые, большие 0). Точки (x, y), в которых $s[x, y] = 0$, будем называть *несегментированными.*

4.1. Обход точек в псевдослучайном порядке. Многие операции над сегментами требуют выполнения некоторых операций для всех точек плоскости. Причем нежелательно перебирать их подряд

$$(0,\,0),\ (1,\,0),\ \ldots,\ (mx-1,\,0),\ (0,\,1),\ \ldots,\ (mx-1,\,my-1),$$

а надо делать это в некотором (псевдо)случайном порядке.

Для этого выберем простое число p, лишь немного превосходящее $mx * my$, и первообразный корень a по модулю p. Тогда последовательность вычетов по модулю p,

$$1,\ a,\ a^2,\ \ldots,\ a^{p-1},$$

пробегает все вычеты от 1 до $p-1$ в некотором порядке. Если число k (вычет по модулю p) принадлежит отрезку $[1,\, mx * my]$, то сопоставим ему точку $((k-1)/mx,\, (k-1)\cdot \bmod mx)$; если вычет не попадает в этот интервал, то не будем сопоставлять ему никакой точки.

На основе предложенного метода можно написать функцию

```
nextPoint(int &x, int &y, int mx, int my, int p, int a),
```

которая для каждой данной точки (x, y) находит следующую точку. Предложенный порядок обхода точек вполне можно считать псевдослучайным для наших целей.

Замечание. Проверка чисел на простоту и нахождение первообразных корней не такие уж простые операции. Для наших целей вполне достаточно заготовить несколько сот простых чисел от 100 до 10^8 и первообразных корней к ним. Доступ к ним можно реализовать в виде функции

```
getPA( int mxmy, int &p, int &a ).
```

4.2. Завершение сегментации. После работы некоторого алгоритма часть точек может остаться несегментированной, т. е. принадлежать к нулевому сегменту. *Завершением сегментации* будем называть процесс присоединения всех несегментированных точек к ближайшему сегменту.

Будем перебирать все несегментированные точки в псевдослучайном порядке. Если такая точка имеет сегментированных соседей, то присоединим ее к тому сегменту, точки которого составляют большинство среди восьми ее соседей.

4.3. Удаление маленьких сегментов. Удалим все сегменты размера меньше заданного, т. е. переведем их в нулевой сегмент, в несегментированные точки. После этого выполним процедуру *завершения сегментации*, описанную в предыдущем пункте.

5. Заключение

Предложенный алгоритм, хотя и включает довольно трудоемкие вычисления, дает результат заметно превосходящий другие алгоритмы сегментации.

Список литературы

1. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. – М.: Техносфера, 2006 – 1072 с.
2. *Кручинин А. Е., Хашин С. И.* Сегментация изображения путем выделения непрерывных границ // Вестник ИвГУ. – 2007. – Вып. 3. – С. 80–83.
3. *Хашин С. И., Хашина Ю. А.* Свойства R-сегментации изображения // Математика и ее приложения: Журн. Иванов. матем. об-ва. – 2007.– Вып. 1(4). – С. 93–98.
4. *Хашин С.И.* Оценка качества сегментации изображения // Вестник ИвГУ. – 2010. – Вып. 2. – С. 112–118.
5. *Яне Б.* Цифровая обработка изображений. – М.: Техносфера, 2007. – 583 с.

Поступила в редакцию 20.12.2011