ЭФФЕКТИВНЫЙ АЛГОРИТМ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ПИРАМИДЫ ГАУССА

Плесков А. В., Писаревский В.Н.

Нижегородская лаборатория программных технологий,
Россия, г. Н.Новгород, ул. Тургенева, д. 14.,
Pleskov@nstl.nnov.ru

ABSTRACT

In this paper we consider an efficient implementation of the gray-scaled and color image segmentation algorithm via the Gauss pyramid linking. The differences from the typical pyramid segmentation technology and output data structures are described. The algorithm has been implemented in C. Also we have done special optimization for Pentium III processor. In the paper we describe experimental results of gray-scaled and color image segmentation, which demonstrate high computational efficiency of the algorithm.

ВВЕДЕНИЕ

Сегментация является одним из наиболее важных этапов обработки и распознавания изображений, цель которой состоит в выделении на изображениях связных областей (объектов) с примерно одинаковыми характеристиками яркости или цветности. Причем в последнее время в связи с необходимости автоматической обработки и распознавания все возрастающего потока как статических так и динамических видео данных, наиболее важное значение придается синтезу эффективных процедур сегментации, работающих в режиме реального времени и позволяющих получать надежные помехоустойчивые результаты.

В этой связи одним из перспективных методов сегментации можно считать метод сегментации, основанный на представлении изображений пирамидами. Преимущества использования данного метода сегментации, на наш взгляд, основаны на преимуществах пирамидального или иерархического подхода к обработке и распознаванию изображений. Последний дает возможность, во первых, от уровня к уровню резко уменьшать объем информации на изображении с сохранением его глобальной структуры, что позволяет получать предварительные результаты обработки по малому объему информации на верхних уровнях пирамиды. Во вторых, благодаря использованию сглаживающих функций при построении пирамиды удается эффективно бороться с различного рода шумами и следовательно повышать помехоустойчивость алгоритмов обработки. И наконец, благодаря наличию межуровневых связей между отдельными элементами пирамиды можно уточнять предварительные результаты переходя от выше лежащих к нижележащим уровням.

Впервые метод пирамидальной сегментации предложил Burt в работе [1]. Позднее стали известны модификации этого алгоритма, в частности взвешенная сегментация [2]. Совсем недавно появилась работа [3], авторы которой предложили новый алгоритм пирамидальной сегментации цветных изображений, названный ими soft segmentation. Основное отличие такой сегментации от традиционной пирамидальной сегментации заключается в том, что в процессе ее выполнения в каждом пикселе исходного изображения вычисляются вероятности его принадлежности той или иной выделенной области.

Необходимо заметить, что во всех перечисленных работах для сегментации использовалась традиционная 4*4 пирамида, в которой каждый элемент следующего уровня вычислялся как среднее по окрестности 4*4 элементов предыдущего уровня. В данной работе предлагается модификация алгоритма пирамидальной сегментации, предложенного в [1], для сегментации полутоновых и цветных изображений, а также ее эффективная реализа-

ция на языке С. Основные отличия нашего алгоритма от базового заключаются в следующем. Во первых, мы применяем для сегментации известную пирамиду Гаусса. Во вторых, для обеспечения требуемой точности в алгоритм сегментации были введены два порога. Первый порог позволяет регулировать точность установления связей между элементами соседних уровней пирамиды. Второй порог регулирует точность сегментации на верхнем уровне пирамиды. В третьих, наш алгоритм позволяет вычислить для каждой связной компоненты не только среднее значение яркости и площадь, но и координаты вершин описанного прямоугольника. При сегментации цветных изображений в качестве меры близости между цветовыми векторами мы использовали несколько простых в вычислительном отношении метрик в цветовом пространстве RGB.
Данный алгоритм был реализован на языке С и оптимизирован под архитектуру процессора Pentium III. В работе приводятся результаты вычислительных экспериментов сегментации цветных изображений, подтверждающие высокую вычислительную эффективность работы алгоритма.

АЛГОРИТМ

Пирамида Гаусса

Пусть исходное оцифрованное изображение задано матрицей вида $F = \{f_{ij}\}$, $1 \le i \le n$, $1 \le j \le m$. Пирамидой Гаусса для изображения F называется рекурсивная последовательность изображений $G^l = \{g_{ij}^l\}, l \ge 0$ следующего вида:

$$G^0 \equiv F, \quad g_{i,j}^l = \sum_{p=-2}^{2}\sum_{q=-2}^{2} w_{p+2,q+2} \cdot g_{2i+p-1,2j+q-1}^{l-1} \quad \text{, для } l > 0,$$

где w_{pq} - весовые коэффициенты дискретной функции Гаусса $W[5*5]$, которую можно определить, как $W = V^T \cdot V$, а вектор V имеет следующий вид:

$$V = 1/16 \cdot \{1,4,6,4,1\}.$$

Очевидно, размер изображения G^l составляет $n/2^l * m/2^l$ пикселей, а максимальное число уровней пирамиды $l_{\max}$ для изображения $F[n*m]$ определяется как. $l_{\max} = \min\{\lfloor \log_2 n \rfloor, \lfloor \log_2 m \rfloor\}$. Таким образом каждый последующий слой G^l пирамиды является сокращенной в 4 раза сглаженной моделью предыдущего слоя G^{l-1}. В настоящее время пирамида Гаусса широко применяется в различных задачах обработки и распознавания изображений.
Между элементами соседних уровней пирамиды можно установить межуровневые связи или ссылки. Ссылки, идущие от элементов уровня l к элементам уровня $l+1$ определяют, в вычислении каких элементов уровня $l+1$ участвует тот или иной элемент уровня l. В случае пирамиды Гаусса каждый элемент с четными индексами имеет также 4 ссылки на элементы уровня $l+1$ с индексами (i'', j''), $i'' \in \{i/2, i/2+1\}$, $j'' \in \{j/2, j/2+1\}$, а каждый элемент с нечетными индексами имеет 9 ссылок на элементы с индексами $i'' \in \{i/2, i/2+1, i/2+2\}$, $j'' \in \{j/2, j/2+1, j/2+2\}$. Если один из индексов четный, а другой нечетный, имеем соответственно 6 ссылок. На границах каждого уровня пирамиды $l+1$ вычисляются значения дополнительных пикселей для того, чтобы обеспечить недостающие ссылки граничных элементов пирамиды уровня l. Обозначим множество таких ссылок для каждого элемента g_{ij}^l - $father(i, j, l)$.

Аналогично определяются обратные связи, идущие от уровня $l+1$ к уровню l. В случае пирамиды Гаусса каждый элемент g_{ij}^{l+1} имеет 25 ссылок на элементы уровня l с индексами $i' \in \{2i-3, 2i-2, 2i-1, 2i, 2i+1\}$, $j' \in \{2j-3, 2j-2, 2j-1, 2j, 2j+1\}$. Обозначим множество таких ссылок для каждого элемента g_{ij}^{l} - $son(i, j, l)$.

Описание алгоритма

Предлагаемый алгоритм сегментации состоит из выполнения трех последовательных этапов.

На первом этапе по заданному изображению F строится пирамида Гаусса до заданного уровня $l_0 < l_{\max}$. Значение l_0 определяется на основе априорных данных о предполагаемом числе связных компонент на изображении. Однако, в отличие от алгоритма [1], в нашем случае число элементов верхнего уровня l_0 может быть меньше, чем число выделяемых связных компонент благодаря введению порогового контроля при установлении связей типа $father(i, j, l)$ между соседними уровнями пирамиды.

На втором этапе происходит итеративное оценивания признаков предварительно выделяемых связных компонент. И наконец на третьем этапе происходит окончательная сегментация, вычисление параметров построенных связных компонент и формирование результирующего растра. Так как построение пирамиды является стандартной операцией, далее подробнее остановимся на описании второго и третьего этапов алгоритма.

Для каждого элемента (i, j) уровня l пирамиды определим следующий набор признаков связных компонент:

- $c[i, j, l][t]$ - текущее значение яркости (цветности) ;
- $a[i, j, l][t]$ - площадь части связной компоненты, принадлежащей данному элементу;
- $p[i, j, l][t]$ - указатель (пара номеров индексов) на один из элементов $father(i, j, l)$;
- $rect[i, j, l][t]$ - параметры описанного прямоугольника, включающие его индексы левого верхнего угла, ширину и высоту, для части связной компоненты, принадлежащей данному элементу.

Здесь $t > 0$ - есть номер итерации.

Вычисление введенных признаков выполняется на втором этапе алгоритма и представляет собой итеративную процедуру.

Для $l = 0$ установим $c[i, j, 0][t] = f_{ij}$, $a[i, j, 0][t] = 1$. Для $t = 0$ и $0 < l \le l_0$ $c[i, j, l][0] = g_{ij}^{l}$.

Далее обозначим $d_{ij}^{l,t}(i'', j'') = \rho(c[i, j, l][t], c[i'', j'', l+1][t])$, где $(i'', j'') \in father(i, j, l)$, а $\rho(x, y)$ - заданная мера близости.

На каждой итерации $t \ge 0$ необходимо произвести следующие вычисления.

$$p[i, j, l][t] = \underset{(i'', j'') \in father(i, j, l)}{\arg\min} d_{ij}^{l,t}(i'', j''), 0 \le l < l_0, \qquad (1)$$

$$a[i, j, l][t] = \sum a[i', j', l-1][t], \ 0 < l \le l_0, \qquad (2)$$

Если $a[i, j, l][t] > 0$, для $0 < l \le l_0$, пересчитываем значения $c[i, j, l][t]$ по следующей формуле:

$$c[i, j, l][t] = \sum (a[i', j', l-1][t] \cdot c[i', j', l-1][t]) / a[i, j, l][t]. \qquad (3)$$

Если $a[i,j,l][t]=0$, $c[i,j,l][t]=c[i',j',l-1][t]$, а (i',j') произвольный элемент, принадлежащий $son(i,j,l)$. Суммирование в формулах (2) и (3) берется по тем $(i',j')\in son(i,j,l)$, для которых $p[i',j',l][t]=(i,j)$. На этом работа одной итерации заканчивается. Затем вычисления по формулам (1) - (3) повторяются на следующей итерации и т.д.

Завершение работы описанной итерационной процедуры можно осуществлять исходя из следующих соображений. Если на некоторой итерации t не изменилось ни одного значения $p[i,j,l][t]$, и следовательно ни одного значения $a[i,j,l][t]$ и $c[i,j,l][t]$, вычисление признаков связных компонент заканчивается. Однако, как показали результаты экспериментов, уже на третьей-четвертой итерациях происходит очень мало изменений. Поэтому в программной реализации в целях увеличения вычислительной эффективности мы фиксировали число итераций (от 3 до 5). В этом случае часть вычислений мы можем отложить до последней итерации. На последней итерации мы вычисляем параметры $rect[i,j,l][t]$ для $0\le l\le l_0$. Пусть $rect[i,j,l][t]$ задается четверкой (p,q,w,h), где p,q - индексы левого верхнего угла прямоугольника, а w,h - соответственно его ширина и высота. Заметим, что все параметры задаются в пикселях исходного растра. Тогда для $l=0$ $rect[i,j,0][t]\equiv(i,j,1,1)$. Для $0<l\le l_0$

$$rect[i,j,l][t]=\bigcup rect[i',j',l-1][t],$$

где объединение берется по тем $(i',j')\in son(i,j,l)$, для которых $p[i',j',l][t]=(i,j)$.

На последней итерации определение связей между элементами соседних уровней в (1) осуществляется с контролем минимальной ошибки $d_{ij}^{l,t}(i'',j'')$. Если выполняется условие

$$\min_{(i'',j'')\in father(i,j,l)} d_{ij}^{l,t}(i'',j'')<D_1, \qquad (4)$$

как и на предыдущих итерациях указатель вычисляется по формуле (1). В противном случае указатель для данного элемента не определяется, и все его параметры записываются в специальный буфер связных компонент. Здесь D_1 - заданный порог точности. После завершения работы последней итерации в выходной буфер записываются также параметры тех элементов верхнего уровня пирамиды l_0, на которые есть по крайней мере один указатель с предыдущего уровня. На этом второй этап работы алгоритма завершается.

На третьем этапе происходит окончательная сегментация элементов в буфере связных компонент. Данная процедура представляет собой реализацию простейшего алгоритма кластеризации, в котором все элементы, значения признака $c[i,j,l][t]$ которых в смысле заданной меры близости $\rho(x,y)$ отличаются друг от друга не более чем на заданное число D_2, образуют один кластер (связную компоненту). Для каждого кластера k вычисляется среднее значение яркости $C(k)$, его площадь $A(k)$ и параметры описанного прямоугольника $RECT(k)$. Соответствующим образом корректируются и значения признаков $c[i,j,l][t]$, $a[i,j,l][t]$ и $rect[i,j,l][t]$ всех элементов попавших в один кластер.

На третьем этапе также происходит формирование результирующего растра, в котором каждому пикселю присваивается среднее значение связной компоненты, которой он принадлежит. Для этого для всех $0\le l<l_0$ выполняем

$$c[i,j,l][t]=c[i'',j'',l+1][t],$$

где $(i'',j'')\equiv p[i,j,l][t]$.

В заключении остановимся на выбранных нами мерах близости. Для сегментации полутоновых изображений мы выбирали $\rho_1(x,y)=|x-y|$. Для сегментации цветных изображений в цветовом пространстве RGB мы выбирали следующие две метрики:

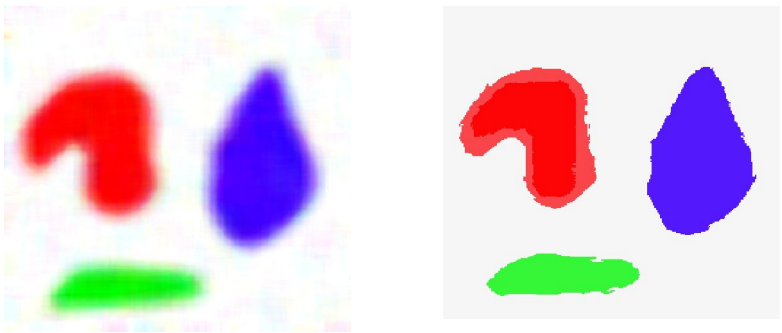
$$\rho_2(c_1,c_2)=\max\{|r(c_1)-r(c_2)|,|g(c_1)-g(c_2)|,|b(c_1)-b(c_2)|\},$$

$$\rho_3(c_1,c_2)=|r(c_1)-r(c_2)|+|g(c_1)-g(c_2)|+|b(c_1)-b(c_2)|.$$

ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

Описанный алгоритм был реализован на языке С и оптимизирован для процессора Pentium III. Проводилась серия экспериментов по сегментации полутоновых и цветных изображений. В качестве первых тестовых изображений выбирались изображения, содержащие несколько произвольных объектов разной яркости и цветности. На рис.1а) приводится цветное тестовое изображение размерностью 256*256 пикселей, а на рис.1б) сегментированный растр после работы алгоритма сегментации. Максимальное число уровней пирамиды выбиралось таким образом, что один элемент верхнего уровня включает в себя более одной связной компоненты, однако , подбирая порог D_1, удалось выделить все эти связные компоненты. Используя порог D_2 удалось объединить в один кластер несколько элементов пирамиды, которые соответствуют фону изображения. На рис.2 приводятся результаты сегментации реального цветного изображения. Эксперименты показали, что метрики $\rho_2(c_1,c_2)$ и $\rho_3(c_1,c_2)$ дают близкие результаты, однако использование метрики $\rho_3(c_1,c_2)$ позволяет получать большую вычислительную эффективность.

Результаты вычислительной эффективности работы алгоритма на процессоре Pentium III с тактовой частотой 500 мгц при различных размерах исходного изображения и числе итераций содержатся в табл. 1.



а) б)

Рис.1

а) б)

Рис.2

Таблица 1

Изображение	Число Итераций	Размер изображения					
		64*64		128*128		256*256	
		(мс)	Кадров/ сек.	(мс)	Кадров/ сек.	(мс)	Кадров/ сек.
Полутон	3	3	330	12	83	67	15
	5	4.4	227	17	59	95	10
Цветное	3	5.8	172	25	40	145	7
	5	8.2	121	34	29	205	5

ЗАКЛЮЧЕНИЕ

В данной работе описана эффективная модификация алгоритма сегментации полутоновых и цветных изображений на основе пирамиды Гаусса. Предложенный алгоритм был реализован на языке С и оптимизирован для архитектуры процессора Pentium III. Проведенные вычислительные эксперименты подтвердили его высокую вычислительную эффективность. Для дальнейшего увеличения его эффективности при сегментации больших изображений возможно применение описанной итерационной процедуры не с нулевого уровня а с некоторого начального уровня пирамиды $l_{\min}$. Таким образом можно уменьшить в несколько раз объем информации и сгладить входное изображение.

ЛИТЕРАТУРА

1. P.J.Burt, T.H.Hong, A.Rosenfeld, “Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation”, IEEE Tran. On SMC, Vol. 11, N.12, 1981, pp. 802-809.
2. T.H. Hong, A. Rosenfeld, “Compact region extraction using weighted pixel linking in a pyramid”, IEEE Transactions on PAMI, V.6, N.2, 1984, pp.222-229.
3. D.Prewer, L.Kitchen, “Weighted Linked Pyramids and Soft Segmentation of Colour Images”, ACCV2000, Taiwan, vol.2, 2000, pp.989-994.