РОССИЙСКАЯ АКАДЕМИЯ НАУК

СИБИРСКОЕ ОТДЕЛЕНИЕ

А В Т О М Е Т Р И Я

2005, том 41, № 2

УДК 004.932.2

А. П. Щербаков

(Томск)

БЫСТРОДЕЙСТВУЮЩИЙ АЛГОРИТМ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ

Рассматривается один из подходов к решению задачи разбиения цветного или полутонового изображения на участки одного цвета. Предлагаемый алгоритм основан на непосредственном поиске регионов с использованием методик работы со списками, применяемых в логическом программировании и системах искусственного интеллекта.

Введение. Как известно, методы, применяемые для сегментации изображений, определяются условиями и требованиями конкретной задачи и по способу функционирования в основном делятся на группы: методы границ, методы поиска регионов, гистограммные и кластерные методы. Методы границ основаны на поиске перепадов яркости и обычно используют оценки значений дифференциальных операторов от функции яркости [1], методы поиска регионов, наоборот, отмечают участки постоянного цвета. Гистограммные и кластерные методы базируются на определении кластеризации точек по их спектральным и пространственным характеристикам [2]. Метод «змеи» (подвижного контура) [3] сводит поиск сегментов к задаче поиска экстремума специально заданной функции.

Интерес специалистов к разработке новых алгоритмов сегментации изображений обусловлен тем, что перечисленные группы методов имеют свои характерные как достоинства, так и недостатки. Метод границ имеет ряд недостатков, отмеченных в работе [1]. Если порог для значений признака границы берется высоким, то снижается влияние шумов, но не улавливаются «слабые» границы. Это ведет к возникновению разрывов границ сегментов и соответственно к их слиянию или потере. Если порог слишком мал, то слабые границы обнаруживаются, но появляется много шумовых элементов, которые ведут к искажению формы сегментов после операции «чистки» (например, по алгоритму Зонга – Суня), а также к потере границ и слиянию сегментов. Как показано в [1], приходится выбирать некоторое компромиссное значение порога, но качество сегментации в таких процедурах далеко не всегда обеспечивает решение конкретной производственной задачи. Фильтрация с использованием вычислений свертки изображения с маской фильтрующего оператора или быстрого фурье-преобразования, или метода вейвлетов

и других методов полностью эту проблему не решает. Как было отмечено в работе [2], первые два типа процедур с трудом применимы к многоцветным изображениям, а гистограммные методы требуют огромных объемов памяти. Следовательно, методы границ применяются в случаях, когда требуется высокая скорость обработки, а гистограммные и кластерные методы – для получения высокого качества обработки, т. е. для точного определения границ и спектральных характеристик сегментов при невысоких требованиях к скорости. В то же время развитие кластерных методов приводит к повышению их быстродействия. Известная программа MultySpec [4] производит сегментацию трехцветного 24-битового рисунка формата 352 288 по алгоритму ISODATA примерно за 2 с, но без пространственной сегментации, края сегментов при этом получаются неровными (рваными) из-за неучета пространственной кластеризации. Кроме этого алгоритм требует предварительной оценки количества кластеров. Метод классификаторов [2] наиболее близок к кластерным методам и позволяет не только работать с большим числом цветов, но и производить полную спектральную и пространственную сегментацию 8-цветного рисунка размером 128 128 за 30 с на компьютере "Pentium-III" с частотой 866 МГц. Быстрый алгоритм, основанный на итерационном слиянии однородных по цвету участков по пирамидной схеме, предложен в [5] и является серьезным шагом вперед (обрабатывает трехцветный рисунок 128 128 за 165 мс), но анализ пространственной сегментации она не производит.

Метод, представленный в данной работе, предназначен в первую очередь для анализа изображений в системах реального времени. Алгоритм можно отнести к методам непосредственного поиска регионов, он позволяет проводить полную сегментацию трехцветного 24-битового рисунка размером 352 288 за 0,3 с (на компьютере "Pentium-IV" Celeron 1700 МГц) и менее, сохраняя высокое качество. Для его реализации применен механизм работы со списками, используемый в системах искусственного интеллекта (ИИ), как в задачах типа «поиск маршрутов» [6] (встречается под названием «задача о коммивояжере»), так и в логическом выводе по методу резолюций [6, 7]. Данный подход, с одной стороны, позволяет определять сегменты поочередно и таким образом уйти от поиска в базах данных по спектральной и пространственной кластеризации, на построении и использовании которых основаны кластерные методы. При поочередной обработке постоянно хранится информация только лишь об обрабатываемом в данный момент кластере. С другой стороны, списки дают возможность уйти от многочисленных рекурсивных вызовов, ведущих к падению скорости и переполнению стека при больших размерах изображения.

Процедура была реализована автором на языке программирования C/C++.

1. Описание алгоритма сегментации. Под списками будем понимать множество выстроенных в цепочку элементов – точек изображения. Описателем точки (ОТ) будем называть множество

$$s \quad \{\text{index}, \text{Class} \quad N_{[0,\ ]};\ P \quad R_{[0,\ ]};\ F \quad \{0,1,2,3\};\ s^{\text{next}};\ s^{\text{prev}}\},$$

где $N_{[0,\ ]}$, $R_{[0,\ ]}$ – множества натуральных и вещественных чисел соответственно в интервале [0,]. Элементы s^{next}, s^{prev} – ссылки на элементы такого же вида, как и s, на практике являются адресами следующего и преды-

дущего элементов списка. В алгоритме используется матрица ОТ $S \quad \{s_{ij};$ $i \quad 1, N_h;\ j \quad 1, N_w\}$ такого же размера, как и обрабатываемое изображение $N_h \quad N_w$, где каждой точке с координатами $[i, j]$ должен соответствовать свой ОТ с такими же индексами в матрице. Будем для простоты обозначать элементы index, Class, P, F, s^{next}, s^{prev} из s_{ij} как index_{ij}, Class_{ij}, P_{ij}, F_{ij}, s_{ij}^{next}, s_{ij}^{prev}.

Алгоритм использует два списка точек (изображения): «Открыт» и «Включен», которые строятся путем записи в поля s^{next} и s^{prev} адресов следующего и предыдущего элементов списка. Нулевой адрес элемента списка в переменной s^{next} означает конец, а нулевой адрес в s^{prev} – начало списка. Элемент F из s_{ij} используется для индикации вхождения в один из списков или обозначения того, что точка полностью обработана (например, 0 – точка еще не обрабатывалась, 1 – точка входит в список «Открыт», 2 – точка входит в список «Включен», 3 – точка полностью проанализирована). Включение элемента s_{ij} в список и удаление из него должны происходить с присвоением элементу F (флагу) соответствующего значения с одновременным удалением из другого списка, если этот элемент там есть. В поле Class_{ij} по окончании процедуры остается номер сегмента, которому принадлежит данная точка.

Этот метод также использует массив описателей кластеров

$$Z \quad \{z_k \quad \{ X_c^k \quad (c \quad 1,2,3);\ N_k \}\}, \qquad k \quad 1, K,$$

где X_c^k – действительные числа, которые являются компонентами усредненного цвета обрабатываемого сегмента с номером k (усреднение происходит в процессе работы процедуры); N_k – количество точек, вошедших в сегмент (кластер) z_k; K – количество зарезервированных описателей кластеров задается разработчиком, который должен быть уверен, что реально обнаруженное число кластеров не превысит K.

Процедура содержит три вложенных цикла, обозначенных номерами 1–3. Запишем соотношения, которые будут присутствовать в алгоритме. Отличие между цветом точки $[i, j]$ и усредненным цветом обрабатываемого сегмента с номером k вычисляется с помощью формулы

$$D(X^{ij}, \ X^k) \quad \sqrt{\sum_{c \quad \overline{1,3}} (X_c^{ij} \quad X_c^k)^2}, \tag{1}$$

где $X^{ij} \quad \{X_c^{ij};\ c \quad \overline{1,3}\}$ – вектор, компонентами которого являются уровни красного, зеленого и синего цветов для точки с координатами $[i, j]$; X_c^k – вектор цвета сегмента под номером k. Для повышения скорости вычислений вместо (1) лучше использовать выражение

$$D\ (X^{ij}, \ X^k) \quad \sum_{c \quad \overline{1,3}} \left| X_c^{ij} \quad X_c^k \right|. \tag{2}$$

Более полная обработка будет при использовании выражения, включающего отличие точки [,] от соседней с ней точки $[i, j]$, уже принадлежащей обрабатываемому в данный момент сегменту

$$D\ (X^{ij},\ X^{k}\ , X\ ) \quad {}_{c\ \overline{1,3}} \left| X_c^{ij} \quad X_c^{k} \right| \quad {}_{c\ \overline{1,3}} \left| X_c^{ij} \quad X_c \right|, \qquad (3)$$

где и – константы, выбранные пользователем. Первое слагаемое отвечает за сегментацию по значениям компонент цвета, второе реагирует на контрастные переходы. При этом скорость обработки падает почти в 2 раза, поэтому для приложений реального времени от одного из слагаемых в (3) лучше отказаться. По наблюдениям автора, зачастую можно обойтись лишь первым слагаемым, полагая 0.

Добавление цвета точки к текущему кластеру z_k производится по рекуррентной формуле

$$X_c^{k}\ [N_k \quad 1] \ \frac{X_c^{k}\ [N_k\ ]\ N_k}{N_k \quad 1} \quad \frac{X_c^{ij}}{N_k \quad 1}. \qquad (4)$$

Здесь $X_c^{k}\ [N_k\ ]$ и $X_c^{k}\ [N_k \quad 1]$ есть X_c^{k} на шаге N_k и $N_k \quad 1$ добавления точки соответственно.

Алгоритм сегментации выглядит следующим образом.

1. Подготовка. Очистить список «Открыт» и список «Включен». Установить счетчик сегментов k : 0. Установить все index$_{ij}$: 1, Class$_{ij}$: 1 и P_{ij} : 0. Занести в список «Открыт» одну из точек s_{ij} (например, в центре изображения).

2. Начало цикла 1. Удалить первую точку из списка «Открыт» и перенести ее в список «Включен». Обозначить ее координаты $[i, j]$. Обнулить средние значения цвета и значение N_k для сегмента z_k.

3. Начало цикла 2. К усредненным компонентам цвета текущего сегмента z_k добавить цвет текущей точки $[i, j]$ согласно правилу (4). Увеличить поле N_k : N_k 1.

4. Начало цикла 3. Проверить все точки [,], соседние с $[i, j]$, которые еще не приписаны никакому сегменту (F_{ij} 3), по следующим правилам:

4.1. Если $D\ (X^{ij},\ X^{k}\ , X\ )$ больше заданного порога T, занести ее в список «Открыт» и перейти к п. 4.2. Если $D\ (X^{ij},\ X^{k}\ , X\ )$ T, то занести точку [,] в список «Включен».

4.2. Если еще не все соседние точки проверены, то взять в качестве [,] следующую соседнюю точку и идти на п. 4.1, иначе п. 4.3. Конец цикла 3.

4.3. Из всех соседних точек выбрать точку $[i, j\ ]$, для которой

$$D\ (X^{ij},\ X^{k}\ , X^{ij})\quad D\ (X^{ij},\ X^{k}\ , X\ )\quad T, \qquad .$$

5. Текущую точку $[i, j]$ исключить из списка «Включен». Присвоить Class$_{ij}$: k и F_{ij} : 3. Взять за текущую точку $[i, j]$ точку $[i, j\ ]$, выбранную на этапе 4.3. Если такой точки не нашлось, взять первую точку из списка «Включен».

6. Если список «Включен» не пуст, то перейти на п. 3, иначе идти на п. 7. Конец цикла 2.

7. Увеличить счетчик сегментов k : k 1. Начало следующего сегмента.

8. Если список «Открыт» не пуст, то перейти на п. 2, иначе закончить процедуру. Конец цикла 1.

В этом алгоритме не используются члены P_{ij} и index_{ij} элементов s_{ij}, применяемые при учете пространственной сегментации, реализация которой описана далее.

Список «Открыт» содержит граничные точки, окружающие уже обработанную зону. Список «Включен» содержит лишь те точки, которые по цвету попали в обрабатываемый сегмент, но окружающие их точки еще не проверены. Оба списка служат для составления плана действий (проверки точек). При работе систем ИИ [6, 7] добавление новых найденных вершин дерева решений в начало списка соответствует поиску в глубину, а в конец списка – поиску в ширину. Метод резолюций использует поиск в глубину. Он начинается с поиска правила, левая часть которого унифицируется с анализируемым предложением. Подробную информацию о реализации метода можно найти в работе [6], где приведен пример упрощенной реализации интерпретатора языка Prolog на языке Lisp. Предлагаемый алгоритм использует список «Включен», не учитывая образованную им очередность, а выбирая вершины по правилу 4.3 и осуществляя другой собственный план поиска, отличный от поиска в ширину или глубину.

Система списков в данном случае способна не только устранить стек, используемый в рекурсивных алгоритмах, но и менять последовательность вычислений путем выбора точек в списке с учетом геометрических или других законов, между тем как стек задает четкую последовательность, которую уже нельзя нарушить. Это делает алгоритмы, работающие на списках, более гибкими для модернизации.

2. Учет пространственной кластеризации. Как известно, из-за шумовых и прочих факторов при сегментации возникают шумовые сегменты малого размера, не несущие информации и занимающие вычислительные ресурсы. Причем количество мелких шумовых сегментов составляет обычно около 2/3 общего числа сегментов и более. Кроме того, шумы приводят к искажениям границ остальных сегментов, неровные (рваные) края которых создают дополнительные трудности при анализе изображений. Вследствие этого требуется проводить дополнительную обработку, частным случаем которой является анализ кластеризации точек в пространстве. Привлечение информации о взаимном положении точек позволяет устранить перечисленные выше эффекты и сделать границы сегментов более оформленными. Так, представленный в [2] классификаторный метод проводит спектральную и пространственную сегментацию, используя кластерный анализ точек как по спектральным, так и по пространственным характеристикам; такие методы описывались в работах [8–10].

Для того чтобы добавить в представленный выше алгоритм наряду со спектральной еще и пространственную кластеризацию, введем оценки вероятности принадлежности точки с цветом X_{ij} текущему сегменту. В качестве модели распределения следует взять функцию в форме колокола, например распределение Гаусса, или более простую:

$$\hat{P}(X^{ij},\ X^{k}\ ,X\ \ )\ \ \frac{}{D\ \ (X^{ij},\ X^{k}\ ,X\ \ )}, \qquad (5)$$

где – ширина размытости «шапочки», которая выбирается пользователем в зависимости от того, насколько мелкое или крупное спектральное дробление кластеров требуется. Так, если значения компонент цвета колеблются от 0 до 256, а желаемое различие цвета кластеров около 2 %, то величина может быть взята около $256 \frac{2}{100}$ 5. Под D $(X^{ij},\ X^{k}\ , X\ )$ обозначена функция (3).

Решающее правило для отнесения точек к текущему сегменту z_k записывается в следующем виде:

$$u(\ ,\ )\quad P_s P\qquad \widehat{P}(X^{ij},\ X^{k}\ , X\ )\ \begin{matrix}[\ ,\ ]\ z_k\\ [\ ,\ ]\ z_k\end{matrix}\ T_p, \tag{6}$$

где T_p – пороговая вероятность;

$$\begin{matrix}P\\ a\quad 1, b\end{matrix}\qquad \begin{matrix}\widehat{P}\ (X^{ab},\ X^{k}\ , X\ ).\\ 1,\ \mathrm{point}[a, b]\ \ z_k\end{matrix} \tag{7}$$

Первое и второе слагаемые (6) отвечают за пространственное и спектральное распределение соответственно. Степень учета пространственной сегментации P_s 0 (s – spatial) есть вероятность (плотность вероятности) того, что точка [,] принадлежит сегменту, если соседняя точка тоже принадлежит этому же сегменту. Данная величина задается пользователем для конкретной задачи. Вероятность (7) для каждой точки [,] насчитывается в процессе прохода через точки, соседние с ней. Пороговая вероятность

$$T_p\quad \frac{}{T}\ (1\quad 4P_s), \tag{8}$$

где T – пороговое значение, которое использовалось в п. 4.1. Множитель 4 соответствует попаданию в текущий сегмент половины из восьми соседних точек. Выражение (7) насчитывается в процессе прохода через точки, соседние с [,]. Пункты 4–4.3 заменяются следующей последовательностью:

4. Начало цикла 3. Проверить все точки [,], соседние с [i, j], не принадлежащие никакому сегменту (т. е. F_{ij} 3), по следующим правилам:

4.1.1. Если номер сегмента, записанный в index (в s), не равен номеру текущего сегмента k, то обнулить P и записать в поле index номер текущего сегмента k.

4.1.2. Прибавить к P оценку вероятности, вычисленную по формуле (5).

4.1.3. Если выражение $u(\ ,\ )$ (6) больше заданного порога вероятности T_p, то внести точку в список «Включен». Иначе, если $u(\ ,\ )$ T_p, внести ее в список «Открыт» и перейти к п. 4.2.

4.2. Если еще не все соседние точки проверены, то взять следующую соседнюю точку и идти на п. 4.1.1, иначе п. 4.3. Конец цикла 3.

4.3. Из всех соседних точек выбрать точку [i, j], для которой

$$D\ \ (X^{ij},\ X^{k}\ , X^{ij})\quad D\ \ (X^{ij},\ X^{k}\ , X\ )\ \ T, \qquad .$$

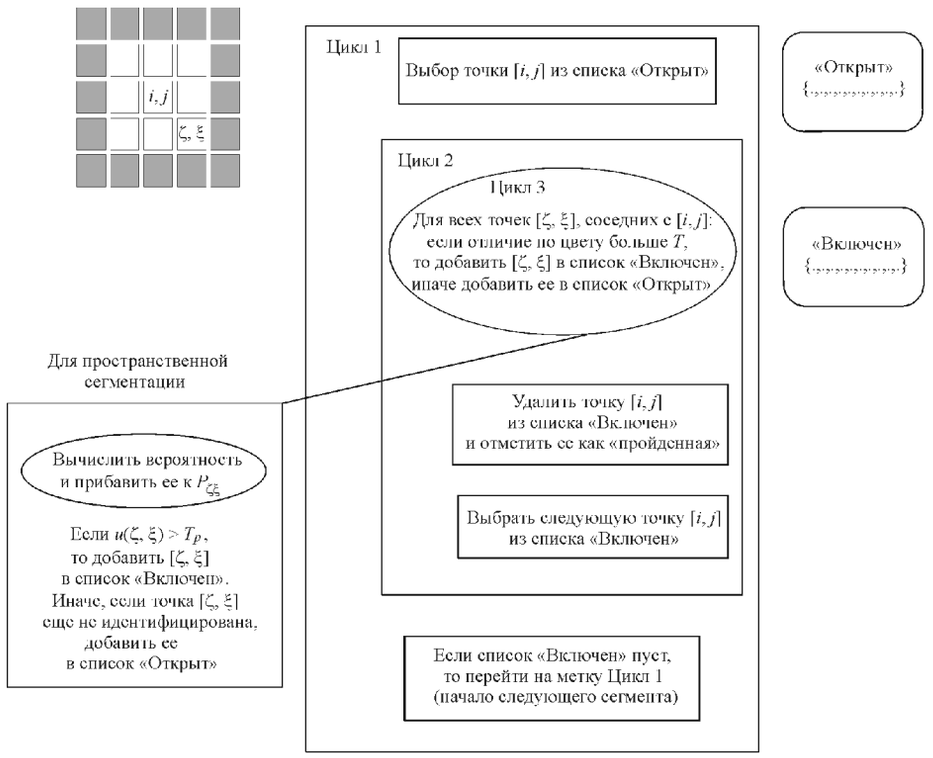

Рис. 1. Блок-диаграмма алгоритма сегментации

Если такой точки нет, то из всех соседних точек, для которых значение $u(\ ,\ )$ было больше порога T_p, выбрать точку $[i, j\ ]$ с максимальным значением $u(\ ,\ )$.

Пункт 4.1.2 содержит суммирование вероятностей для точек. Если такая точка окажется хотя бы частично окруженной точками текущего сегмента, то она попадается повторно при проходе соседних с ней точек. Значение P при этом увеличивается. Добавляемая к P_{ij} величина вероятности (5) зависит от того, насколько цвет точки далек от цвета текущего сегмента. При P_s 0алгоритм работает аналогично схеме, описанной в разд. 1. Полная блок-диаграмма алгоритма изображена на рис.1.

После отладки алгоритма вычисление оценок вероятности (5) и их сравнение с порогом лучше реализовать на более скоростной целочисленной арифметике, заменив в ОТ тип P_{ij} целочисленным, и использовать вместо значений выражений (5)–(8) целую часть результата их умножения, например, на 1000.

Заключение. Пример работы процедуры приведен на рис. 2. На рис. 2, d для сравнения показан результат обработки без учета пространственной кластеризации при P_s 0.

Алгоритм содержит один проход по всем точкам, что обеспечивает высокую скорость. Предложенный метод согласно теоретическим предпосылкам не может иметь более высокую точность и качество, чем у сложных и ресурсоемких кластерных алгоритмов, потому что является их скоростным вариантом и имеет свои ограничения. Количество одновременно учитываемых

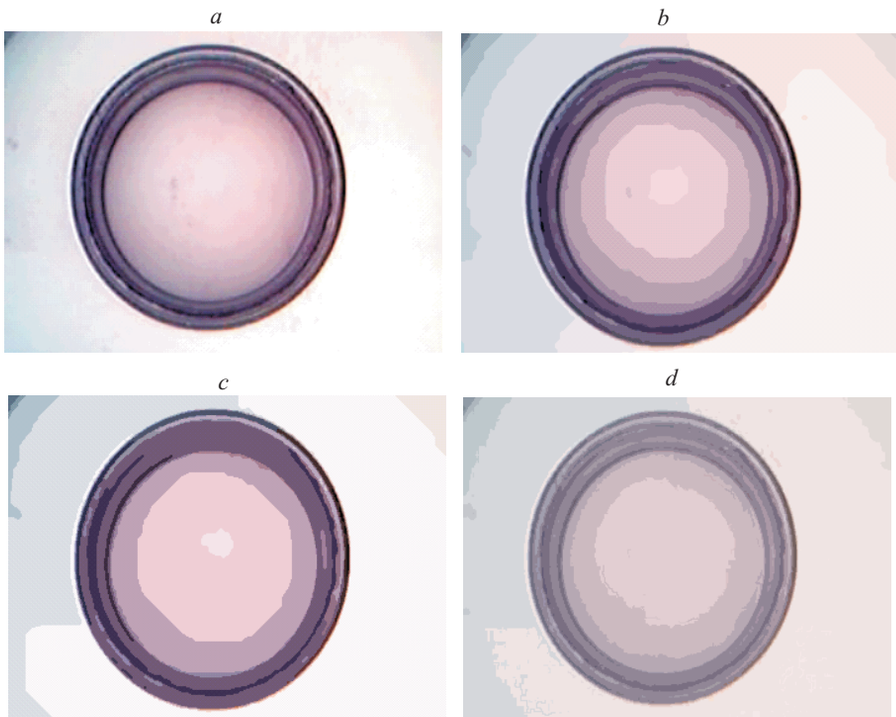

Рис. 2. Деталь подшипника: исходное изображение (*a*); обработка при T 20, 30, P_s 0,25 (*b*); обработка при T 40, 50, P_s 0,25 (*c*); обработка при T 40, 50, P_s 0 (*d*)

при анализе пространственной кластеризации соседних точек сведено к минимуму и ограничено квадратом 3 3. Из-за этого алгоритм при определенном сочетании параметров T, и P_s может допускать некоторую неточность определения положения неконтрастных границ (плавных, расплывчатых переходов). Это может затруднить применение его для исследований в области физики, если требуется более точная сегментация, например, при анализе запечатленных на снимке физических процессов, а также в задачах составления карт на основе аэрофотоснимков. В то же время представленный алгоритм вполне может использоваться в задачах идентификации объектов в системах реального времени.

Автор выражает благодарность д-ру физ.-мат. наук Л. Н. Синице и д-ру физ.-мат. наук А. М. Быкову за помощь в приобретении оборудования для проведения экспериментов.

СПИСОК ЛИТЕРАТУРЫ

1. Хорн Б. К. П. Зрение роботов. М.: Мир, 1989.
2. Paclic P., Duin R. P. W., Van Kempen G. M. P., Kohlus R. Segmentation of multi-spectral images using the combined classifier approach // Image and Vision Comput. 2003. 21. P. 473.
3. Ji L., Yan H. Loop-free snakes for highly irregular objectshapes // Pattern Recogn. Lett. 2002. 23, N 5. P. 79.
4. http://www.scanex.ru/rus/lakm/int_sem4/about_multispec.htm

5. Marfil R., Rodriguez J. A., Bandera A., Sandoval F. Bounded irregular pyramid: a new structure for color image segmentation // Pattern Recogn. 2004. 37. P. 623.
6. Хювенен Э., Сеппянен И. Мир Лиспа. М.: Мир, 1990. Т. 2.
7. Тей А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию /Под ред. Г. П. Гаврилова. М.: Мир, 1990.
8. Spann M., Wilson R. A quad-tree approach to image segmentation with combines statistical and spatial information // Pattern Recogn. 1985. 18, N 3/4. P. 257.
9. Haralic R., Shapiro L. Survey: image segmentation techniques // CVGIP. 1985. 29. P. 100.
10. Matas J., Kittler J. Spatial and feature based clustering: application in image analysis // Proc. CIAP95. Prague, 1995. P. 162.

Институт оптики атмосферы СО РАН,
E-mail: molnija2@inbox.ru.

Поступила в редакцию
5 апреля 2004 г.