ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

References

1. Gonsales, R. Tsifrovaya obrabotka izobrazheniy [Digital image processing] / R. Gonsales, R. Vuds. — Moscow, 2005. — 1072 pp.
2. Malla, S. Veyvlety v obrabotke signalov [A wavelet tour of signal processing] / S. Malla. — Moscow, 2005. — 671 pp.
3. D'yakonov, V. MATLAB. Obrabotka signalov i izobrazheniy. Spetsialnyy spravochnik [MATLAB. Signal and image processing. Special handbook] / V. D'yakonov, I. Abramenkova. — St.-Petersburg, 2002. — 608 p.
4. Stolnits, E. Veyvlety v komp'yuternoy grafike [Wavelets for computer graphics] / E. Stolnits, T. DeRouz, D. Salezin. — Moscow, Izhevsk [Res.Center "Regular and Chaotic Dynamics"], 2002. — 272 pp.

Рецензент д-р. техн. наук, проф. Одес. нац. політехн. ун-ту Крилов В.М.

Надійшла до редакції 17 листопада 2011 г.

УДК 517.5:004.932:004.421

Г.Н. Востров, канд. техн. наук, доц.,
М.Г. Годынский, магистр,
Одес. нац. политехн. ун-т

ПОСТРОЕНИЕ АДАПТИВНОЙ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЯ НА ОСНОВЕ БЛОКОВОЙ СТРУКТУРЫ

Г.М. Востров, М.Г. Годинський. **Побудова адаптивної сегментації зображення на основі блокової структури.** Проаналізовано проблеми побудови сегментацій зображень, досліджено особливості їх алгоритмів. Представлено алгоритми адаптивної сегментації зображень на основі блокової структури.
Ключові слова: методи обробки зображень, сегментація зображень, блокова структура.

Г.Н. Востров, М.Г. Годынский. **Построение адаптивной сегментации изображения на основе блоковой структуры.** Проанализированы проблемы построения сегментаций изображений, исследованы особенности их алгоритмов. Представлены алгоритмы сегментации изображений на основе блоковой структуры.
Ключевые слова: методы обработки изображений, сегментация изображений, блоковая структура.

G.N. Vostrov, M.G. Godynsky. **Construction of adaptive image segmentation based on the block structure.** The problems of constructing image segmentation are analyzed, the features of their algorithms are investigated. The algorithms of adaptive image segmentation based on the block structure are presented.
Keywords: image processing techniques, image segmentation, block structure.

В различных областях обработки изображений важно определение особенностей обрабатываемого изображения [1]. Например, при сжатии [2], классификации, интерполяции (как временной, так и пространственной) изображений часто очень важно иметь возможность выделять присутствующие в изображении объекты. Выделенные сегменты, обладающие специфическими для данной области свойствами частотоустойчивости, амплитудоустойчивости, дисперсионными свойствами, особенностями изменений автокорреляционной и корреляционной функций, могут быть использованы в тех или иных интернет-приложениях, программах связи, видеоконференций [3], системах слежения и других системах компьютерной обработки изображений. Особое значение такая процедура имеет в дискретном вейвлет-преобразовании, в

котором для каждого блока изображения, в зависимости от его особенностей и свойств, может выбираться собственный вейвлет-фильтр, что делает предварительное адекватное разбиение на блоки крайне важным этапом, влияющим на весь последующий процесс. Такой алгоритм позволяет формировать на каждом этапе пирамидального преобразования оптимальные пакеты фильтров с различными свойствами на разных уровнях пирамидального кодирования.

В результате возникает необходимость создания и использования методов, получающих на входе исходные изображения и дающих на выходе за приемлемое время их сегментированные варианты с набором результирующих характеристик. Предлагаемый метод предназначен для адаптивной сегментации изображения, позволяя эффективно определять особенности изображений, такие как наличие и форма объектов. Под объектом понимается совокупность пикселей, частично отражающая объект реального мира и обладающая схожими значениями компонентов. При стандартной сегментации объекты могут разделяться так, что в блоках, содержащих части объектов, эти части будут незначительными. В результате при последующем поиске они будут слабо влиять на результат, а объект будут иметь разрывы или вообще не будет найден.

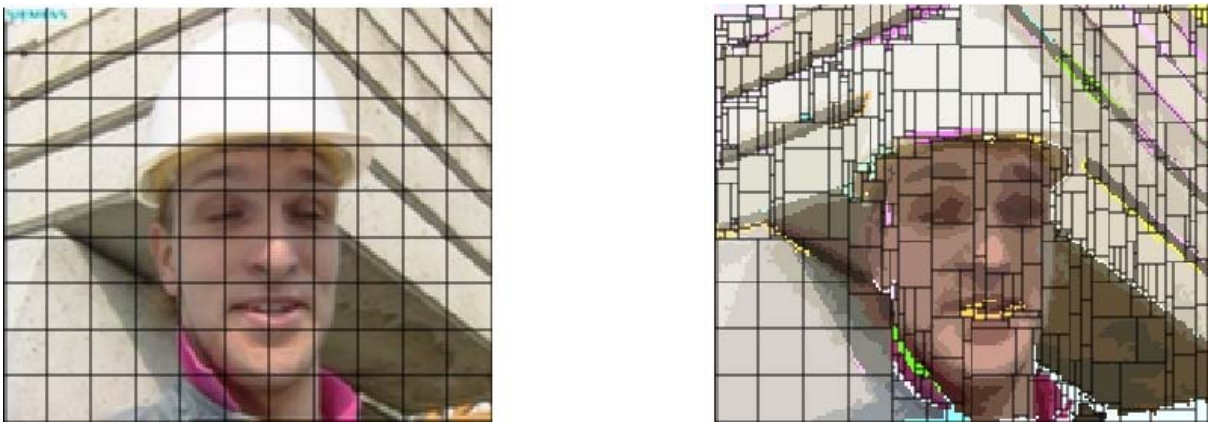
Предлагаемый метод сегментации, в отличие от стандартного, учитывает границы и целостность объектов и позволяет значительно сократить количество случаев, когда границы блоков разделяют видимый на изображении объект на отдельные части. Так, на рис. 1 приведено сравнение стандартного [2] и адаптивного методов сегментации.

Основными целями работы являются:

— выделение объектов в изображении для сохранения их целостности для дальнейшей обработки;

— разбиение изображения на блоки с учетом обнаруженных объектов;

— построение адаптивной сетки разбиения на прямоугольные блоки произвольных размеров в заданном диапазоне с учетом границ объектов.



а *б*

Рис. 1. Сетка разбиения: стандартная (а); адаптивная (б)

На первом этапе производится выделение объектов изображения, т.е. построение регионов, приближенно описывающих реальные объекты на изображении. Для этого применяется процедура создания регионов путем присоединения пикселей друг к другу при условии, что их значения близки по выбранному критерию. Затем осуществляется удаление шума и присоединение мелких регионов, не содержащих существенной информации, к более крупным регионам: такие блоки назовем присоединяемыми, при этом они содержат ссылки на соседние блоки. Под мелкими регионами в данном случае понимаются блоки, количество пикселей в которых меньше заданного порога, а под шумом — образованные после выделения очень мелкие объекты, являющиеся следствием выбросов функции распределения яркостей данного изображения.

На втором этапе производится разбиение изображения на блоки с учетом найденных объектов на картинке, т.е. построение адаптивной сетки, учитывающей границы между найденными регионами.

Итак, рассмотрим разработанный алгоритм.

I этап.

ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

Процедура 1. Объединение пикселей в приближенные регионы.

Входные параметры и переменные:

— *threshold* — порог, задающий максимальное расстояние между пиксельными компонентами сравниваемых элементов в YUV-представлении.

Разность между компонентами цветового пространства YUV вычисляется по формуле

$$dif = |Y_1 - Y_2| \cdot 0{,}5 + |U_1 - U_2| + |V_1 - V_2|, \quad (1)$$

где Y_1, U_1, V_1 — компоненты первого элемента,

Y_2, U_2, V_2 — компоненты второго элемента.

Средние значения цветовых компонент для регионов вычисляются по формуле

$$cAverageY = \frac{\sum_{i=1}^{N} Y_i}{N}, \quad (2)$$

где Y_i — значение *Y*-компоненты *i*-го пикселя,

N — количество пикселей.

— *aPixels* — массив структур, описывающих пиксели изображения и хранящих значения пикселей, номера регионов и блоков, к которым они относятся.

— *nMaxRegion* — номер текущего региона.

Шаг 1.1. Установка *nMaxRegion* в 1.

Шаг 1.2. Выбор начального пикселя для нового региона, который еще не был отнесен к какому-либо региону; сохранение значения его компонент (*cAverageY, cAverageU, cAverageV* — начальные средние значения для региона); занесение пикселя в стек пикселей и присвоение ему номера нового региона *nMaxRegion*.

Шаг 1.3. Извлечение верхнего элемента стека и запоминание значения его компонент (*cCurrY, cCurrU, cCurrV*) , если стек не пуст. Если он пуст, то переход к шагу 1.7.

Шаг 1.4. Выбор еще необработанного соседнего пикселя для пикселя, выбранного на шаге 1.3. Если необработанных соседних пикселей больше нет, то переход к шагу 1.3.

Шаг 1.5. Проверка для выбранного на шаге 1.4 пикселя разности по отношению к пикселю (*cCurrY, cCurrU, cCurrV*), выбранному на шаге 1.3. Также для выбранного на шаге 1.4 пикселя проверяется разность по отношению к средним значениям для региона (*cAverageY, cAverageU, cAverageV*). Если разности оказались больше установленого порога *threshold*, то осуществляется переход к шагу 1.4, иначе — к шагу 1.6.

Шаг 1.6. Присоединение выбранного соседнего пикселя к региону и обновление среднего значения для региона (*cAverageY, cAverageU, cAverageV*), занесение пикселя в стек пикселей и присвоение ему номера нового региона *nMaxRegion*. Переход к шагу 1.4.

Шаг 1.7. Присвоение *nMaxRegion* = *nMaxRegion* + 1. Если на изображении есть еще необработанные пиксели, то переход к шагу 1.2, иначе конец работы 1.1.

Перед процедурой 2 определяются и сохраняются границы всех регионов. Очень мелкие регионы обрабатываются особым способом — они объединяются с соседними такими же мелкими регионами. Это впоследствии позволяет избежать образования множества очень мелких блоков.

Процедура 2. Присоединение мелких регионов к более крупным регионам.

Входные параметры и переменные:

— *minpix* — порог, задающий минимальное количество пикселей в регионе, при котором регион не будет объединен с соседним более крупным регионом;

— *nMaxRegion* — максимальный номер региона, равный количеству регионов;

— *curRegion* — номер текущего выбранного региона;

— *small_regs* — массив маленьких регионов, которые будут объединены между собой, а не с большими регионами;

— *SMALL_MIN_PIX* — максимальный размер очень малых регионов.

Шаг 2.1. Присвоение *curRegion* = 1. Если *curRegion* > *nMaxRegion*, то переход к шагу 2.10.

Шаг 2.2. Если в текущем обрабатываемом регионе количество пикселей *npix* > *minpix* или *npix* < 0, то переход к шагу 2.9.

Шаг 2.3. Полагаем *z* = 0; *size* — количество граничных пикселей в текущем регионе. Если *z* > *size*, то переход к шагу 2.7.

Шаг 2.4. Рассмотрение всех пикселей, окружающих *z*-й пиксель контура региона, и регионов, которым они принадлежат.

Шаг 2.5. Нахождение номера региона, который ближе всех к текущему региону по средним значениям компонент YUV. Запоминание его в поле *near_region*.

Шаг 2.6. Приравнивание $z = z + 1$ и переход к шагу 2.4.

Шаг 2.7. Объединение регионов, если у текущего региона количество окружающих регионов меньше трех и в поле *near_region* есть номер региона, ближайшего к текущему.

Шаг 2.8. Иначе, если количество пикселей текущего региона меньше *SMALL_MIN_PIX*, то текущему региону выставляется флаг *small_region* и он заносится в *small_regs*.

Шаг 2.9. Присвоение *curRegion* = *curRegion* + 1 и переход к шагу 2.2.

Шаг 2.10. Объединение всех соседствующих регионов, занесенных в small_regs.

II этап. Адаптивное разбиение на блоки.

Схема 1.

Разбиение на блоки реализует полное покрытие изображения прямоугольными блоками с размерами в диапазоне до *max_dim*. Блоки строятся путем последовательного увеличения их размеров до максимально возможных. Увеличение происходит по двум направлениям: вправо и вверх. Процесс построения блоков происходит в два этапа: сначала строятся блоки, покрывающие мелкие регионы, объединенные между собой, затем покрывающие оставшееся пространство изображения. Это необходимо для того, чтобы более точно представить разбиение на блоки объектов небольших размеров, но все же заметных для зрительного восприятия, т.к. именно на таких объектах больше всего проявляются разрывы их структуры. После построения блоков происходит их связывание друг с другом, что необходимо при поиске блоков. Связываются друг с другом только соседние блоки.

Входные переменные:

— *aPixels* — массив, в котором хранятся данные о пикселях изображения;

— *right*, *up* — логические переменные, определяющие возможность увеличения размеров в соответствующих направлениях;

— *x*, *y* — начальные координаты блока;

— *dx*, *dy* — размеры блока;

— *block_cnt* — количество блоков в изображении;

— *segm_blocks* — массив, хранящий блоки изображения;

— *adjacent* — массив, хранящий блоки, соседствующие с данным блоком, — имеется в каждой структуре, описывающей блок.

Шаг 1.1. Присвоение *block_cnt* = 0.

Шаг 1.2. Выбор начального пикселя, который еще не был приписан к какому-либо блоку и принадлежит мелкому региону, для нового блока. Ему присваивается номер нового блока *block_cnt* = *block_cnt* + 1. Задание начальных координат блока и его размеров: *dx* = *dy* = 1.

Шаг 1.3. Присвоение *right* = *up* = true.

Шаг 1.4. Если размер блока по одной стороне превосходит размер блока по другой стороне, то переход к шагу 1.9.

Шаг 1.5. Проверка возможности дальнейшего увеличения блока вверх.

Шаг 1.6. Если up равно true, то увеличиваем размеры блока вверх на 1 пункт.

Шаг 1.7. Повторяем шаги 1.5 и 1.6, но уже для увеличения размеров блока вправо.

Шаг 1.8. Если *up* или *right* равно true, то переход к шагу 1.4, иначе — к шагу 1.9.

Шаг 1.9. Удаление блока и переход к шагу 1.2, если размер блока *dx* или *dy* равен 1. Иначе — к шагу 1.10.

Шаг 1.10. Выравнивание размеров блока *dx* и *dy* по размерам соседних блоков, что необходимо для устранения ступенчатости при заполнении блоками. Для этого находится ближайшая граница соседнего блока и по ней выравнивается граница обрабатываемого блока, если такое выравнивание не уменьшит значительно размеры текущего блока. Предел изменений размера блока определяется значением в *min_dim* пикселей.

Шаг 1.11. Занесение получившегося блока в массив *segm_blocks*. Переход к шагу 1.2.

Шаг 1.12. Повторение шагов 1.2 — 1.11 для оставшихся необработанными пикселей.

Шаг 1.13. Приравнивание *i* = 0.

Шаг 1.14. Выбор *i*-го пикселя изображения.

Шаг 1.15. Выбор еще необработанного соседнего пикселя для пикселя, выбранного на шаге 1.14. Если необработанных соседних пикселей больше нет, то переход к шагу 1.17.

ISSN 2076-2429 (print)
ISSN 2223-3814 (on line)

Шаг 1.16. Занесение номера блока, соответствующего пикселю, выбранному на шаге 1.15, в массив *adjacent* для блока, содержащего пиксель, выбранный на шаге 1.14. Переход к шагу 1.15.

Шаг 1.17. Приравнивание $i = i + 1$. Переход к шагу 1.14.

Шаг 1.18. Присоединение всех получившихся мелких блоков, у которых *dx* или *dy* равно 1, к соседним более крупным блокам.

Схема 2.

Альтернативная схема адаптивного разбиения обладает следующими специфическими особенностями:

— объекты разбиваются на блоки таким образом, чтобы впоследствии эти блоки можно было бы легко объединять в более крупные или разбивать на более мелкие;

— более точно учитываются граничные области объектов, т.к. при их представлении используются только блоки малых размеров.

Шаг 2.1. Цикл по i=1, N — числу объектов.

Шаг 2.2. Выбор i-го объекта.

Шаг 2.3. Нахождение для текущего объекта минимального прямоугольника, включающего все его пиксели. Координаты данного прямоугольника — *x*, *y*, размеры прямоугольника — *dx*, *dy*.

Шаг 2.4. Разбиение прямоугольника, полученного на шаге 2.3, на четыре подпрямоугольника.

Шаг 2.5. Проверка на наличее в каждом из полученных четырех прямоугольных блоков пикселей, принадлежащих другим объектам. Если имеются, то для таких блоков переходим к шагу 2.4. Такая проверка продолжается до тех пор, пока размеры блока не станут равны двум минимальным размерам.

Шаг 2.6. Если $i \le N$, тогда приравниваем i=i+1. Иначе переход к шагу 2.7.

Шаг 2.7. Приравнивание i=1.

Шаг 2.8. Выбор i-го объект.

Шаг 2.9. Занесение в переменные *x*, *y*, *dx*, *dy* координаты левого нижнего угла и размеров прямоугольника, найденного на шаге 2.3.

Шаг 2.10. Последовательный выбор пикселей из прямоугольника с координатами *x*, *y*.

Шаг 2.11. Проверка, включен ли выбранный пиксель в один из уже созданных блоков. Если да, то переход к шагу 2.10, иначе — к шагу 2.12.

Шаг 2.12. Построение на основании выбранного пикселя прямоугольного блока, используя шаги 2.2 — 2.11 из первой схемы алгоритма разбиения.

Шаг 2.13. Если в прямоугольнике с координатами *x*, *y* и размерами *dx*, *dy* есть еще необработанные пиксели, то переход к шагу 2.10. Иначе — переход к шагу 2.14.

Шаг 2.14. Если $i \le N$, тогда присваиваем i=i+1. Иначе конец алгоритма.

Все этапы работы алгоритма приведены на рис. 2.

Стоит отметить, что до начала выделения объектов необходимо определить значения входных параметров метода, в частности, пороги сегментации. Для оптимального определения данных порогов используется процедура адаптивного подбора. Ее смысл заключается в последовательном проведении алгоритма построения регионов с различными значениями параметров, изменяющимися с заданным шагом. При этом запоминаются параметры, дающие лучший результат. Критерием определения качества результата является получаемое после разбиения количество мелких регионов.

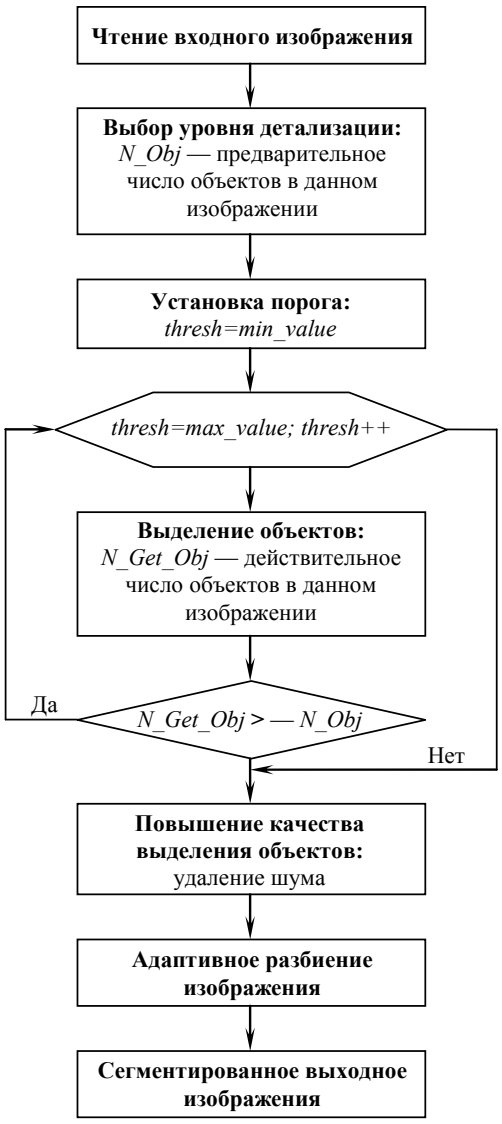


Рис. 2. Этапы работы алгоритма адаптивной сегментации

Пример работы алгоритма показан на рис. 3.

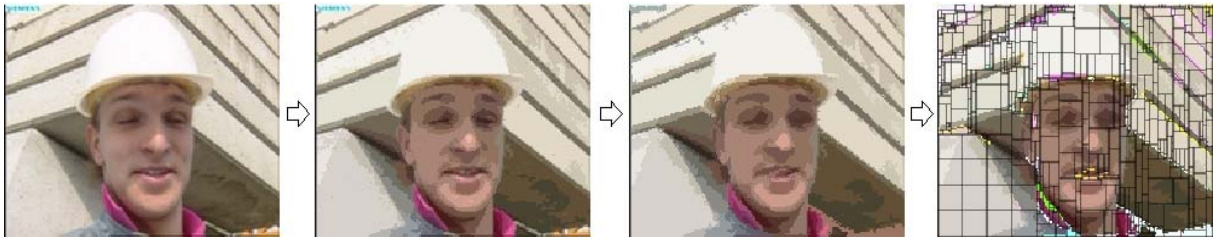


Рис. 3. Исходное изображение ⇒ Выделение объектов ⇒ Удаление шума ⇒ Адаптивное разбиение

Таким образом, построенный алгоритм может быть использован для работы с разными классами изображений: от реалистических, отображающих картины реального мира, до искусственно созданных, таких как схемы и диаграммы. При этом следует оговорить, что его вычислительная сложность (и соответственно время его работы) существенно зависит от особенностей входного изображения, например, от наличия в изображении большого количества мелких деталей. Поэтому его применение нецелесообразно для изображений с большым количеством перепадов яркости и мелких деталей, не представляющих собой текстуры. В последнем случае работа алгоритма сильно замедляется из-за большого количества выделяемых регионов, и отпадает смысл разбиения на блоки, поскольку получаются очень малых размеров, что недостаточно для последующего эффективного поиска необходимых блоков.

Литература (References)

1. Park, J. Hierarchical data structure for real-time background subtraction / J. Park, A. Tab, A. Kak // IEEE Intern. Conf. on Image Processing, Atlanta, USA, 2006, Oktober. — 2006. — P. 1849 — 1852.
2. Kim, J.-T. An image coding method by construction of variable block size and region compensation / J.-T. Kim, D.-W. Kim, J-S. Oh and others. — 6th WSEAS International Conference, Hangzhou, China. — 2006. — P. 102 — 104.
3. Ngan, K.N., Hongliang L. Video segmentation and its applications / K.N. Ngan, L. Hongliang // Springer Science+Business Media, New York, 2011. — P. 164.

Рецензент д-р техн. наук, проф. Одес. нац. политехн. ун-та Крышов В.Н.

Поступила в редакцию 16 сентября 2012 г.