5

A Parallel Framework for Image Segmentation Using Region Based Techniques

Juan C. Pichel, David E. Singh[1]
and Francisco F. Rivera[2]
[1]*Computer Science Dept., Universidad Carlos III de Madrid*
[2]*Electronic and Computer Science Dept., Universidade de Santiago de Compostela*
Spain

1. Introduction

Segmentation is the partitioning of an image into multiple regions (sets of pixels) according to a given criterion. The goal of segmentation is typically to locate objects of interest within the image. A wide variety of methods and algorithms are available to deal with the problem of the segmentation of images (Fu and Mui, 1981; Haralick and Shapiro, 1985; Pal and Pal, 1993). These methods can be broadly classified into four categories (Zhu and Yuille, 1996):

- Edge-based techniques.
- Region-based techniques.
- Deformable models.
- Global optimization approaches.

The edge-based techniques are based on information about the boundaries of the image. Therefore, they try to locate the points in which abrupt changes occur in the levels of some property of the image, typically brightness (Canny, 1986; Rosenfeld and Kak, 1982). On the other hand, those methods that use spatial information of the image (e.g. color or texture) to produce the segmented image fit into the region-based techniques (Chen et al., 1992; Sahoo et al., 1988). These methods depend on the consistency of some relevant property in the different regions of the image. The deformable models are based on curves or surfaces defined within an image that moves due to the influence of certain forces. They can be classified into various groups, principally snakes, deformable templates and active contours (Blake and Isard, 1998; Kass et al., 1988). All of these techniques avoid the use of a global criterion when segmenting the image, which is contrary to the global optimization approaches (Geman and Geman, 1984; Kanungo et al., 1994).

In this work a unified framework for image segmentation is proposed. The technique consists of two stages: a parallel seeded region growing algorithm (PSRG) and a region merging heuristic (RM). In Figure 1 the functional scheme of the proposed algorithm is shown. In the first step, different segmentations, performed in parallel, of the same input image are obtained. Each of these segmentations, which from now on will be called *partial segmentations*, are also generated in parallel using different number of processors. This way, the region growing algorithm uses a two level parallelism. Next, a region merging heuristic is applied to the oversegmented image created as result of combining the different initial

Source: Vision Systems: Segmentation and Pattern Recognition, ISBN 987-3-902613-05-9,
Edited by: Goro Obinata and Ashish Dutta, pp.546, I-Tech, Vienna, Austria, June 2007

segmentations. The merging process is guided using only information about the behavior of each pixel in the initial segmentations (without external parameters). In order to guide the merging stage we introduce a magnitude called repulsing force between neighboring regions that measures the tendency of them to remain separated in the oversegmented image. In order to stop the merging process an evaluation function of the segmented images was used. In addition the algorithm has been validated using several real images with different sizes and characteristics, and it has been tested on a HP Superdome cluster.

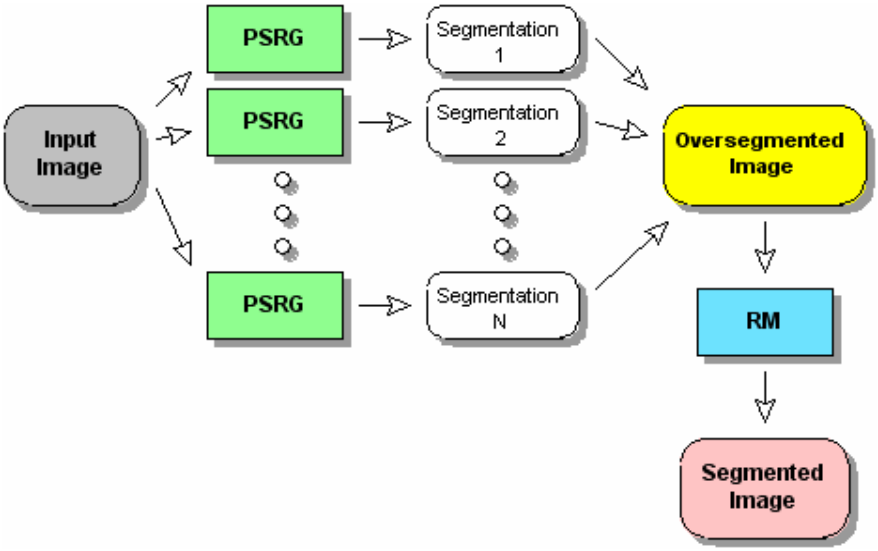


Fig. 1. Scheme of the proposed segmentation algorithm

2. Parallel Seeded Region Growing (PSRG) algorithm

Our proposal was inspired by the region growing algorithm introduced by Mehnert and Jackway (1997), referred as SRG (Seeded Region Growing algorithm) from now on. One of the main benefits of this algorithm is that it solves the dependencies imposed by the previous proposal by Adams and Bischof (1994) in the order to access the pixels in the image.

The SRG algorithm starts with a set of pixels in the image to be segmented, called seeds. These seeds are the starting point to determine the regions in the image. Pixels that are neighbor of the seeds are candidates to be included in the corresponding region. Some established similarity criterion is used to decide which of them are finally added to the corresponding region. This process is repeated sequentially in an incremental way, and the regions begin to "grow". In each iteration, the pixels that are candidates to be included in at least one region are stored in a Neighbour Holding Queue (NHQ). In this algorithm, the similarity δ between these pixels and its neighbour region is defined as the difference between the grey intensity of the pixel and the average value of intensities of the pixels that currently define the region.

In the SRG algorithm, the candidates are stored in a list of LIFO queues defined by consecutive and disjoint intervals of similarity, in such a way that pixels in the i-th queue

are those that present similarities in the interval $[\delta_i^{min},\delta_i^{max}]$. Queues are ordered according to δ_i^{min}, defining an ascending Priority Queue (PQ). In each iteration, only pixels in the First Queue (FQ), that is the queue with lowest δ_i^{min}, are considered, and in the SRG algorithm, all of them are immediately assigned to the corresponding region.
To avoid dependencies in the SRG algorithm, the average intensities of the regions are not updated in each assignment. Only after all the pixels in the FQ are assigned, this average is updated. Note that at this point new pixels can be included in the NHQ as well as in the PQ queues.

2.1. Implementation of the parallel algorithm

The proposal of a parallel algorithm based on the RGS is referred as PSRG (Parallel Seeded Region Growing algorithm). The main idea behind this proposal is to assign a different set of seeds or regions to each process, in such a way that the corresponding regions grow independently. Each process works with a subset of the regions. Regions assigned to a particular process are called local. The growing process in the regions of each process is not completely independent of the rest, but each process must take into account the state of the regions assigned to the remaining processes. Note that with this approach, the segmentation could not be the same as in the sequential case, and that it depends on the distribution of regions among processes.
In our approach, we consider second order neighbourhood for the pixels (Wang and Wang, 2004), however the algorithm can be adapted to any other case. The implementation of the parallel code uses the standard Message Passing Interface (MPI) library (Gropp et al., 1994). The main stages of the whole algorithm are described next.

2.1.1. Initial distribution of seeds

This stage can be considered as a preprocessing routine. Apart from initialling the parameters and variables, i.e. the position of the seeds, its main objective is to distribute the seeds (regions) among the available processes. As we will see in next stages, this distribution is important because the number of overlaps directly depends on it. *Overlaps* are the pixels that are simultaneously assigned to different regions. Note that regions in the same process can not produce overlaps. Therefore, in order to reduce overlaps, regions that are going to be neighbours must be assigned to the same process. This situation can not be foreseen, but in many cases a good approach can be obtained if we consider the distance among seeds, in such a way that seeds that are near to each other have a large probability of producing neighboring regions. We used a version of the Prim´s algorithm (Gibbons, 1984) on the position of the seeds in the image to reorder and distribute the regions. In addition, the regions are equally distributed among processes to achieve good load balance.

2.1.2. Parallel Region Growing

In this stage the SRG algorithm is applied to the set of regions in each process with the addition of two types of communications among processes: one to detect overlaps, and other to control the growing speed to avoid artificial growing of some regions. Two parameters T_1 and T_2 are introduced to determine the interval between pairs of communications of each type in terms of number of iterations of the SRG algorithm. This stage is referred as PRG algorithm. Next, both types of communications are introduced with detail.

2.1.2.1. Communications to detect and solve overlaps

After T_1 iterations a new communication to deal with overlaps is performed. For each process the objective is to know the pixels already assigned by other processes. Therefore, overlaps (pixels assigned to different regions in different processes) among regions are detected. These pixels are labelled as *borderline pixels*, and their neighbours are not considered as candidates in the following iterations.

This communication can be efficiently implemented by a reduction operation. Each processor labels the pixels already assigned to any local region as 1, and labels 0 the rest of them. After the reduction (summation) of the labels among processors for the whole image, the pixels with labels higher that 1 are considered overlaps. Note that only two bits are needed to label the pixels.

The number of iterations between communications is a parameter that affects the performance of the parallel code. We propose to consider as a initial value a estimation of the number of pixels to be processes before the first overlap:

$$T_1 = \frac{N}{R} \cdot D_{min} \tag{1}$$

Where R is the number of regions, P is the number of processors and D_{min} is the minimum euclidean distance in number of pixels among seeds assigned to different processes.

After this first value, T_1 changes dynamically according to the number of pixels detected as overlaps in the previous communication. We propose to use the values given by:

$$T_1 = \alpha \cdot \frac{\beta}{\Delta K} \tag{2}$$

Where α is a parameter that characterizes the cost of the communications in the particular system, β is the agreeable maximum number of pixels in the overlap areas between pairs of communications (this parameter can be tuned by the user), and ΔK is the number of new overlap pixels since the previous communication.

2.1.2.2. Communications to control the growing speed

In our parallel algorithm, each process has its own PQ and FQ, and the values of similarity they consider in a particular iteration can highly differ from one process to another. Therefore it is necessary to include some action to avoid unfair grows produced by local FQs with lower values of δ than in other FQs.

To deal with this problem, we propose to use a reduction operation to evaluate de maximum and minimum values of δ used in the FQs, δ^{max} and δ^{min} respectively. In such a way that a new parameter ϕ is defined to specify the agreeable interval of similarities to be processed in each iteration given by the size L:

$$L = (\delta^{max} - \delta^{min})\phi \qquad 0 \le \phi \le 1 \tag{3}$$

Therefore, if the similarity of a particular pixel in a local PQ is lesser than δ^{min} + L, then it is processed, otherwise it is not assigned to the associated region. The number of iterations between reductions is defined by T_2 (established by the user). Note that this communication

presents a lower cost that the previous one, because it involves just two values instead of information about all the pixels in the image.

2.1.3. Redistribution of seeds

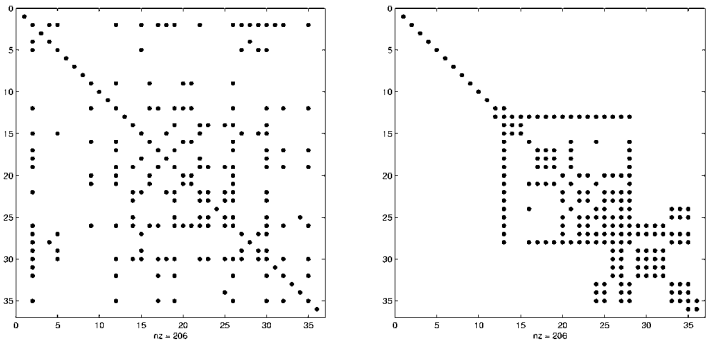
In this stage, a new distribution of regions among processes is performed in order to assign overlap pixels to regions. The objective is to minimize the number of overlap pixels after the execution of the PRG algorithm. This new distribution just involves the overlap area of the image. This stage consists of three steps:

1. Finding overlap pixels.
2. Obtaining the new redistribution of regions among processes.
3. Finding the optimum number of processes needed and establishing communications to perform the redistribution.

The idea behind this stage is to assign those regions that share overlap pixels to the same process, in such a way that a new execution of PRG can assign these pixels locally. Next we analyze the above three steps with more detail.

After the PRG algorithm was executed, each process has a number of regions defined by a set of pixels, and no other process has these pixels assigned to any of its regions. In fact, all these regions are limited by a border line defined by the overlap pixels or the frame of the image. So, a reduction operation involving all the processes is carried out. The objective of these communications is every process to know the overlap pixels and the regions that have them as part of its borderline.

Two regions are said to be close to each other if both have as neighbour at least one overlap area. In this step close regions are detected by a parallel flooding algorithm. As a result of this, a so called adjancency matrix M is obtained. This matrix is defines as: M[i][j]=0 if regions *i* and *j* are not close to each other, and M[i][j]=1 otherwise. Note that M is a symmetric matrix. Then, the Cuthill-McKee algorithm (Saad, 1996) is applied to reorder the matrix in such a way that the nonzero entries are moved near the diagonal. Figure 2 shows the effect of applying this algorithm. This reordered matrix can be partitioned into contiguous blocks that are distributed among the processes. The result of this distribution is that groups of close regions are asigned to the same process.



(a) (b)

Fig. 2. (a) Original adjacency matrix, (b) reordered matrix using the Cuthill-McKee algorithm.

2.1.3.3. Obtaining the number of processes and establishing communications

In this step, the most appropriate number of processes is obtained. Note that regions that are not close to any other are not relevant for this stage, because they can not grow, so they are not being condidered in this step. In order to obtain a good load balance, the same number of regions are assigned to the processes Figure 3 shows an example of this partition for 3 processes. The entries that are inside one of these partitions are solved in this step, however the others (marked as shared entries in the figure) will not be solved now (they represent the shared overlap areas). The time needed to finish this step is limited by the process that has more entries in the partition of M. This number is denoted as E_{max}. In addition, shared entries will be processed in the final stage, if their number is E_{shared}, then the cost of both processes can be modelled by the linear expression $K = A \cdot E_{max} + B \cdot E_{shared}$. Parameters A and B are used to weight the relative cost among of this step and the final stage. In our experiments, we found that adequate values for them are: $A=1.5$ and $B=1$. Therefore, to obtain the optimum number of processes, K should be minimized. Finally, the regions are distributed among the selected processes, and the PRG algorithm is executed.

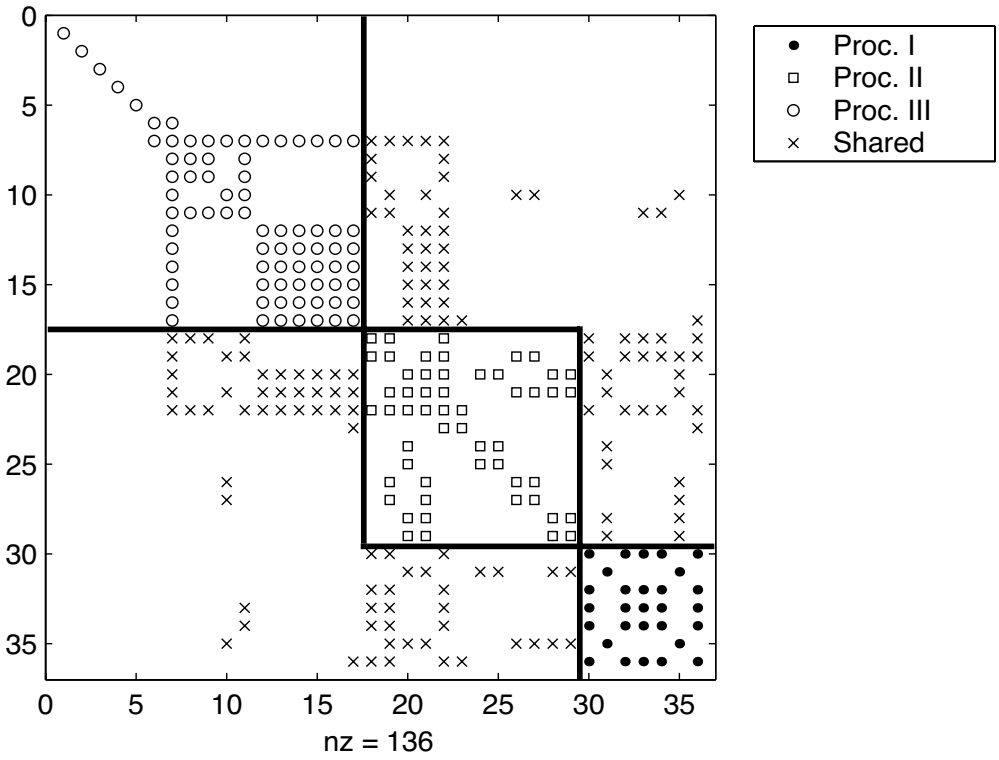


Fig. 3. Example of the partitioning of an adjacency matrix.

2.1.4. Final stage

In this stage, the shared overlap areas are solved sequentially by the RGS algorithm, and therefore the last overlap pixels are finally assigned to regions. Note that after this stage,

some groups of pixels, called *island*s, could be disconnected from the seed. These islands are easily detected by a flooding algorithm and finally added to their best neighbour region according to the similarity criterion.

3. Obtaining the oversegmented image

The PSRG algorithm presents two problems. On one hand, it has a great dependence with the initial position of the seeds. Moreover, the number of seeds is the number of regions of the final segmented image and therefore, we need *a priori* knowledge of the image to obtain a good segmentation. On the other hand, this type of segmentation algorithms (region based ones) force all the pixels of the image to belong to a region in the segmented image, so there will be pixels that belong to regions with low similarity levels.

We deal with these problems creating an oversegmented image from different executions of the PSRG algorithm with seeds placed randomly and introducing the concept of shadow zone. Later, this oversegmented image will be processed by a region merging method (detail in Section 4) to obtain the final segmented image. Note that this proposal presents a two-level parallelism: a coarse-grain one defined by the parallel execution of several PSRGs, and a fine-grain one defined by the parallel nature of the PSRG algorithm.
The MPI library was used to implement the parallel code of this proposal. This library allows the definition of groups of processes that fits with our two-level parallelism model, in such a way that the number of processes to solve each partial segmentation and the number of partial segmentations can be easily established. In other words, if N partial segmentations are executed with P processes each one, then the total number of processes is NxP.

3.1. Generation of seeds

Each partial segmentation is obtained from a different set of seeds. These seeds can be obtained randomly if there is no a priori information about the image. After that, each segmentation is obtained by executing the PSRG algorithm with P processors.

3.2. Shadow zones

As we have mentioned above, one of the drawbacks of the SRG and PSRG algorithms is that the number of regions is exactly determined by the number of seeds. Generally, the objective of the segmentation algorithms based on regions is the labeling of all the pixels of the image. In many cases, in the final stages of the process, this situation causes pixels to be included in regions from which they have very low similarity levels, thereby creating regions with low homogeneity. To avoid this effect we propose the inclusion of a specific threshold (ε) as a possible solution. This threshold would be such that those pixels with a low degree of similarity with respect to the target region are not included in it, remaining unlabeled. The set of unlabeled pixels will be called shadowed zones. Therefore, a pixel, after the partial segmentation, can be labeled and thus belongs to a region, or it can be included in a shadowed zone.
Figure 4 shows the effect of applying PSRG with different values of ε on the Lena image and using 30 seeds. In particular Figure 4(b) shows the segmentation produced by PSRG without threshold, and Figures 4(c), 4(d) and 4(e) show the result when ε is more and more restrictive. Note that when ε is low, the image present more details that when ε is high.

However, when ε is too low, the number of shadow zones can be so high that their execution in the following stages is less efficient.
It is important to note that the shadow pixels are not processed by PSRG algorithm. Any way, they will be taken into account in the creation of the oversegmented image and by the merging process.

(a) (b) (c)

(d) (e)

Fig. 4. Partial segmentations of the image Lena: (a) Original image, (b) ε = 255, (c) ε = 100, (d) ε = 50 and (e) ε = 35.

Figure 5 shows two partial segmentations produced by two different sets of seeds for the PSRG algorithm with no threshold ($\varepsilon = 255$) (Figures 5(a) and 5(b)) and the PSRG algorithm with $\varepsilon = 35$ (Figures 5(c) and 5(d)). As expected, note that the partial segmentations obtained by PSRG with $\varepsilon = 35$ are more similar to each other than the ones obtained by PSRG with $\varepsilon = 255$. We can conclude that using PSRG with shadow zones minimize the dependence of the final segmentation from the initial position of the seeds.
Our proposal generates an oversegmented image as a combination of various partial segmentations in which shadowed areas can exist. There are other segmentation techniques that create an oversegmented image such as watershed algorithms (Haris et al., 1998). As we detail later, we need to collect, from different partial segmentations, the information that will guide the merging process. In our algorithm an operation for intersecting all the partial segmentations is performed in such a way that those pixels that belong to the same region in all the partial segmentations remain united in one of the regions of the oversegmented image. Figure 6 shows a simple example of the creation of an oversegmented image formed, in this case, from three partial segmentations. The first partial segmentation presents a shadowed zone and two regions, whilst in each of the other two segmentations there are

three regions and no shadowed zones. In this example the oversegmented image consists of five regions.

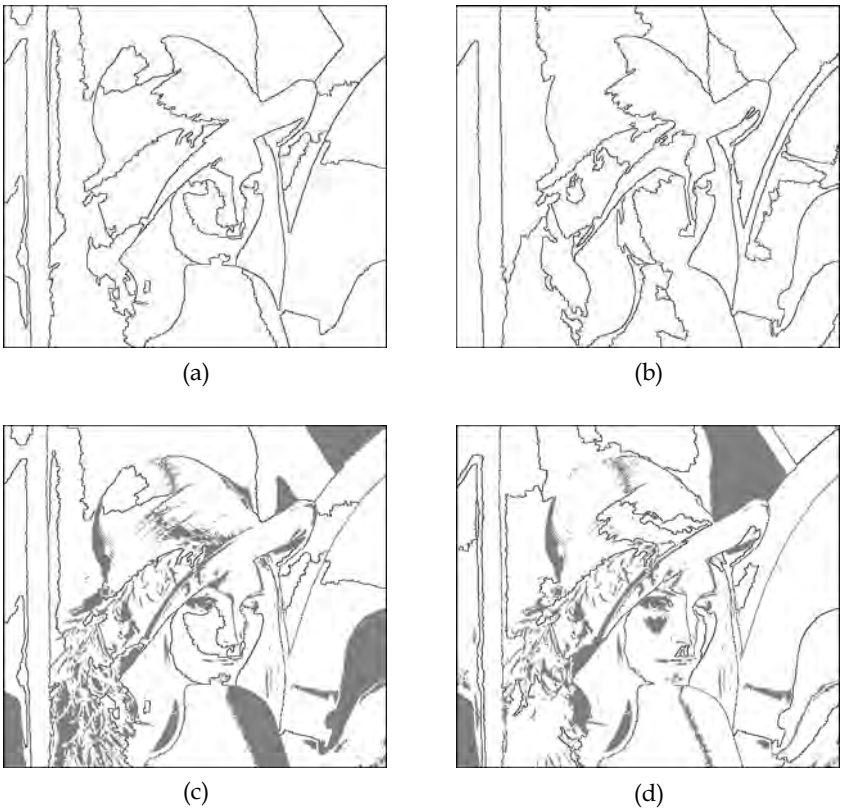


Fig. 5. Partial segmentations of the image Lena using different seeds: (a) and (b) ε = 255, (c) and (d) ε = 35.

In Figure 7 two oversegmented images obtained using four partial segmentations of Lena are shown. Note that when the threshold ε decreases, the number of regions of the oversegmented image grows. This behaviour is due to the existence of many shadowed zones in the partial segmentations obtained using the PSRG algorithm. This way, if shadowed zones exist in some of the partial segmentations, they are considered as any other region in the generation of the oversegmented image, although later, in the merging algorithm, they will be treated differently.

Finally, Figure 8 shows the evolution of the number of regions of the oversegmented images created from 2, 3 and 4 partial segmentations of the Lena image using different number of seeds. The results show that when using a higher number of seeds, the number of regions of the oversegmented images increases. This behavior is observed as well for an increasing number of partial segmentations.

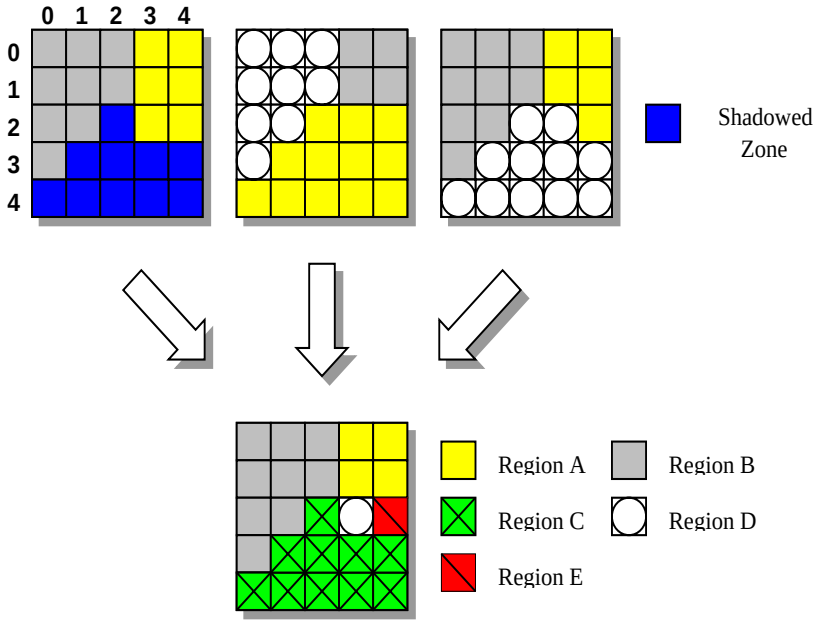
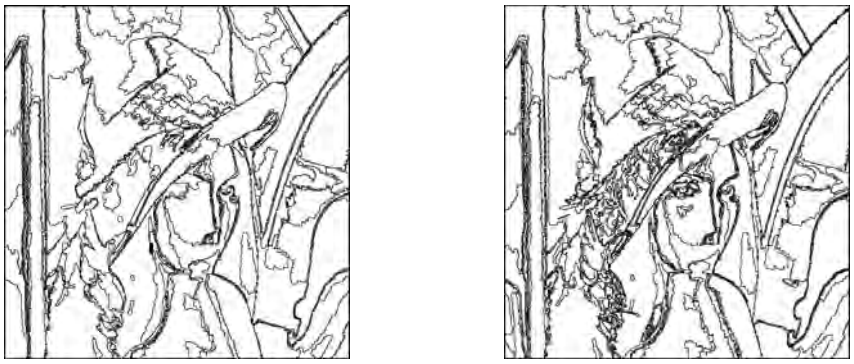


Fig. 6. Example of the generation of an oversegmented image from three partial segmentations of 5×5 pixels with three regions each one.



(a) (b)

Fig. 7: Oversegmented image created using four partial segmentations of the image Lena: (a) ε = 255 and (b) ε = 50.

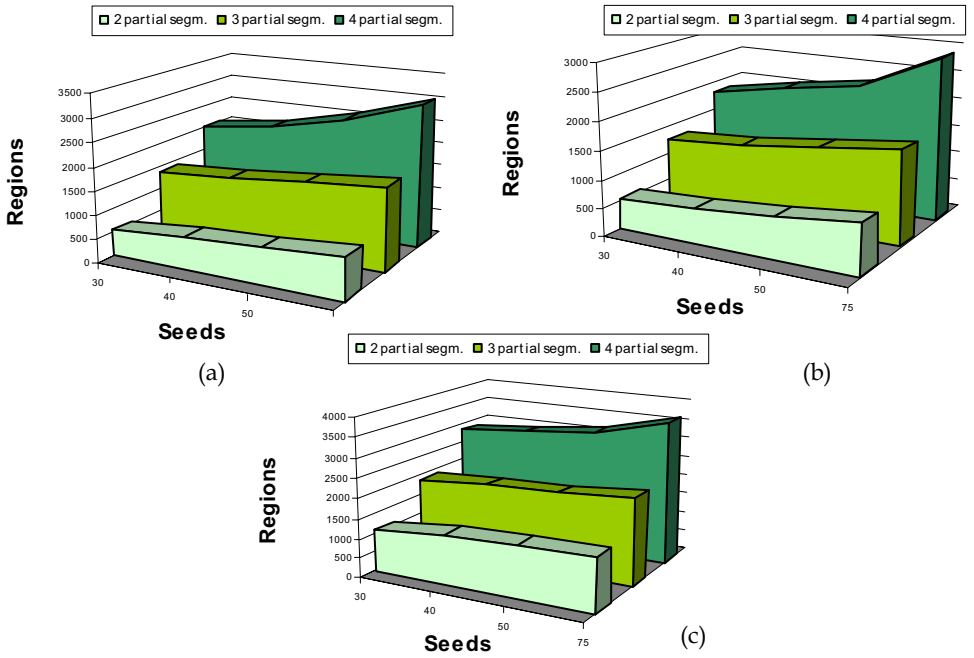


Fig. 8. Average number of regions of the oversegmented image using Lena as input image: (a) ε = 255, (b) ε = 100 and (c) ε = 50.

4. Region Merging (RM) algorithm

In a previous work (Pichel et al., 2006) a new region merging algorithm was introduced. The main contribution of this proposal is that all the relevant information to obtain the final segmented image is obtained exclusively from the different partial segmentations, both for creating an oversegmented image and for applying the subsequent region-merging algorithm. In this paper, the partial segmentations are performed using the PSRG algorithm, and later the RM algorithm is applied. This strategy does not take into account local characteristics such as size, shade of average grey intensities, etc. Based on the results obtained for all the pixels of the image in each of the partial segmentations, global conclusions are obtained with respect to them. Without losing generality, the merging algorithm uses a second-order neighborhood scheme, so that up to eight neighbors are defined for each pixel.

The RM algorithm uses a force of repulsion between two neighboring pixels i and j that measures the tendency of these pixels to be or not in the same region. This force is given through the following equation:

$$f_{ij} = -C_1 n_1(i,j) - C_2 n_2(i,j) + C_3 n_3(i,j) + C_4 n_4(i,j) \quad (4)$$

where C_K are parameters to weight each of the situations in which two neighboring pixels could be found, and n_K is the number of times this situation occurs in the initial segmentations. The four situations are the following: both belong to same labeled region (n_1), both belong to different labeled regions (n_2), both belong to shadowed zones (n_3), and one belongs to a region and the other to a shadowed zone (n_4).
Therefore, two different components can be identified in Equation 4: on one hand an attractive component given by term $-C_1 n_1(i,j) - C_2 n_2(i,j)$ that measures the tendency of these pixels to belong to the same region, and on the other, a repulsive component given by term $C_3 n_3(i,j) + C_4 n_4(i,j)$ that measures the opposite. According with this equation, the lesser the force f_{ij} the greater is the tendency for these pixels to belong to the same region. Additionally, we have that the pixels that belong to the same region always verify $f_{ij}<0$.
In (Pichel et al., 2006) an analysis to determine the way these parameters are related was performed. We can summarize the relationships between the parameters as: $C_4 > C_1 > C_3 > C_2$. Moreover, several definitions were introduced. Let R be a region in the oversegmented image, and let S be a neighboring or adjacent region to it. We define the set of pixels of R neighbors of S as:

$$V_{R,S} = \{ i \in R \mid \exists j \in S, \text{such that } i \text{ and } j \text{ are neighbors} \} \quad (5)$$

S and R are neighbors if $V_{R,S} \neq \varnothing$.
For each i in $V_{R,S}$, we define the associated set of its neighbors belonging to S, as:

$$U^i_{R,S} = \{ j \in S \mid i \text{ and } j \text{ are neighbors, and } i \in V_{R,S} \} \quad (6)$$

Then, the force of repulsion between neighboring regions is defined as the average of the forces of repulsion of all the neighboring pixels belonging to each one of those regions:

$$F_{R,S} = \frac{\sum_{i \in V_{R,S}} \left(\sum_{j \in U^i_{R,S}} f_{ij} \right)}{\sum_{i \in V_{R,S}} \| U^i_{R,S} \|} \quad (7)$$

where $\| \bullet \|$ is the cardinality operator.
The structure of the data used for representing the partitions of the oversegmented image is a region adjacency graph (RAG) (Haris et al., 1998). The RAG of a segmentation of K regions is defined as a weighted undirected graph, $G=(V,E)$, where $V=\{1,2,...,K\}$ is the set of nodes and $E \subset V \times V$ is the set of edges. Each region is represented by a node, and between two nodes $R,S \in V$ there is an edge (R,S) if the regions are neighbors. A weight is assigned to each edge of the RAG, so that those nodes joined by the lesser (or greater) weighted edge, depending on its definition, will be the regions that are candidates for merging. In our case, the function used to assign weights to the edges is the force of repulsion F_{RS} given by the Equation 7 and therefore, the regions that are candidates for merging are those joined by the edge with least weight.
Using the RAG as input, an iterative heuristic to deal with the problem of merging is proposed, so that one merging is performed in each of the iterations, based on the weight of

the edges. In each iteration of the RM algorithm, the pair of regions that have the smallest weight are merged. A data structure adequate for storing the weights is a queue, which can be implemented using a heap (Knuth, 1973). All the edges of the RAG are stored in the heap according to the weights, so that the first edge always has the smallest weight. Given the RAG of an initial partition of *K* regions denoted as (K-RAG), and a heap of its edges, the RAG of the partition *K-n* is obtained using the merging algorithm described in the following pseudo-code:

```
DO i = 0, n-1
    Find minimum cost edge in (K-i)-RAG
    Merge the selected pair of regions to get the (k-i-1)-RAG
    Update the heap
END DO
```

The *K-RAG* corresponds to the initial partition of the image, which in our case, is the oversegmented image. Subsequently, an iterative process is applied, in which, in the *i*-th iteration, the two regions with the smallest eight are merged. Once they have been merged, the list of edges is updated, and the *(K-i-1)-RAG* is obtained.

It is inferred that *n* iterations are needed to obtain the *(K-n)-RAG*. Therefore, one of the problems posed by this strategy is to establish the best value of *n*, i.e., the best number of regions of the final segmented image. Different alternatives can be used. For example, using the property of the growing value of the first term of the heap, a certain threshold can be selected to stop the iterations when the value of the edge at the top of the heap exceeds it. The main drawback of this approach is that it is not evident to determine a good threshold *a priori*. Another approach is the use of a method that allows a numerical evaluation of the segmentation for choosing the best threshold according to some selected validation criterion.

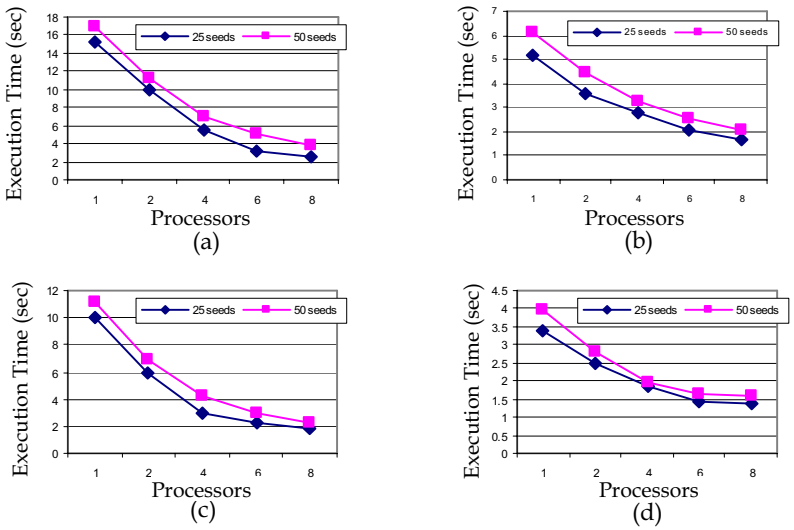


Fig. 9. Average execution times of the PSRG algorithm using different parameters for the image Lena (512×512 pixels): (a) $\phi = 0.01$, $\varepsilon = 255$, (b) $\phi = 0.05$, $\varepsilon = 255$, (c) $\phi = 0.01$, $\varepsilon = 25$ and (d) $\phi = 0.05$, $\varepsilon = 25$.

5. Results

In this section the results obtained by our proposal are shown. We have focused on two different aspects: the execution time required by the algorithm to obtain a final segmented image and the quality of the segmentation provided by the algorithm.

5.1. Execution Times

Our algorithm has been tested on a HP Superdome cluster with 128 1.5 GHz Itanium2 processors and 384 GBytes of memory. As example, in Figure 9 the execution times of the PSRG algorithm using an image of 512×512 pixels (Lena) are shown. The graphics point out that the code presents a good scalability, obtaining speedups up to 6.2 when using 8 processors per segmentation. Note that the execution times when using ε are lower than those obtained when the threshold is not applied (ε = 255). This behavior is due to, as we have commented before, the higher number of pixels to be added to the regions. In turn when ϕ increases, the execution time of the PSRG algorithm also increases.

Processors per partial segmentation

Execution Time (sec)	1	2	4	6	8
PSRG	9.9	5.8	3.1	2.3	1.9
RM	7.1	7.1	7.1	7.1	7.1
PSRG+RM	17.0	12.9	10.2	9.4	9.0

Table 1. Average execution times of the segmentation algorithm for the image Lena using a different number of processors per partial segmentation.

Finally, we have measured the execution times of the global segmentation system including the parallel algorithm (PSRG) and the sequential one (RM). The results are shown in Table 1. In the example, four partial segmentations were performed in order to create the oversegmented image, with ε = 100 and ϕ = 0.05.

5.2. Evaluation of the segmentation

As case of study we use function *Q* both to objectively evaluate the quality of the algorithm, as well as to adjust the value of the parameters of the weight function of the edges of the RAG. The evaluation function *Q* was proposed by Borsotti et al.(1998) for color images, which is a variant of that proposed by Liu and Yang (1994). One of its main advantages is that do not require any external parameter. Specifically, function *Q* is expressed by:

$$Q = \frac{1}{10000(N \times M)} \sqrt{R} \sum_{i=1}^{R} \left[\frac{e_i^2}{1 + \log A_i} + \left(\frac{R(A_i)}{A_i} \right)^2 \right] \tag{8}$$

where N×M is the size of the image, R is the number of regions, A_i and e_i are the area in number of pixels and the quadratic error of the color values of the i-th region, respectively. Also, $R(A_i)$ represents the number of regions that have an area equal to A_i. The smaller the value of Q, the better the segmentation of the image. For images in grey levels, we have

adapted the normalization term of the previous equation to obtain values within a similar range to those obtained by Q in color images, and the definition of e_i corresponds to the mean value of the grey level of the i-th region.

In (Pichel et al., 2006) an exhaustive study was performed to a broad set of test images in order to determine the values of the weight parameters of the force of repulsion. Finally, the following values were proposed: C_1=1, C_3 =0.2, C_3=0.8 and C_4=2.

In order to illustrate the behaviour of our proposal in a more precise way, in Figures 10, 11 and 12 the values of function Q compared to the number of regions of the oversegmented image using a different number of partial segmentations are shown. Shadowed zones, defined by different thresholds, have been used in all the tests. From the behavior Q, we can infer that when the number of regions is equal to the number that each image really has, Q presents its minimum value.

This behaviour is absolutely clear in the case of the image Test1 (Figure 10). This is a synthetic image and consists of 5 homogeneous regions. For this image, as we can observe in the figure, only two partial segmentations are needed to obtain a correct final segmented image. Nevertheless, the situation is different when the input image is a real one like Lena and Peppers (Figures 11 and 12). Note that, in these cases, a clear global minimum of Q does not exist, so we cannot decide on which is the best segmentation with this criterion. The information that we can extract is that an interval of values of Q exists, shown in the figures with an arrow. In this interval the most adequate segmentations can be found. The segmented images displayed correspond with a local minimum of function Q when using six partial segmentations.

Therefore, based on these results we conclude that for the segmentation of real images, Q is very useful for determining the set of the most adequate segmentations, but in most of the cases it is not going to be sufficiently discriminating to select just one. But note that using our proposal high quality segmentations are obtained.

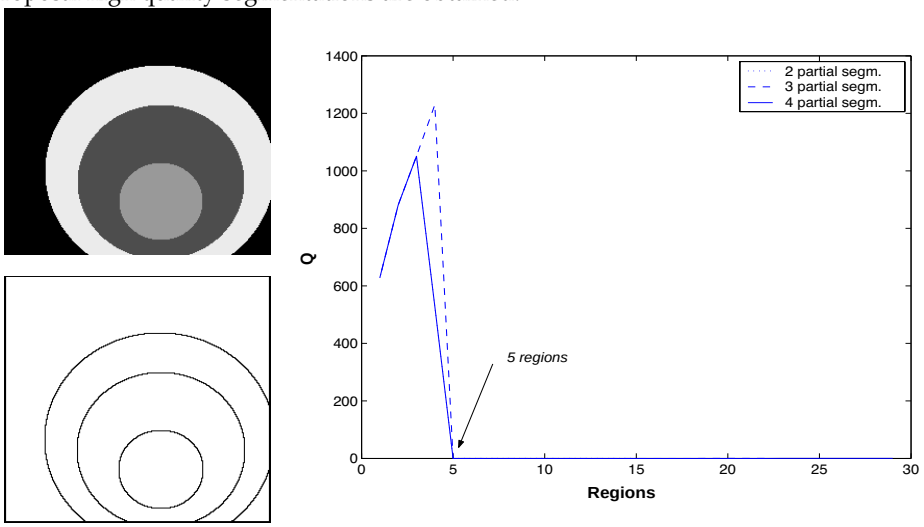


Fig. 10. Segmentation of the image Test1: original image, result of the segmentation and Q compared to the number of regions.

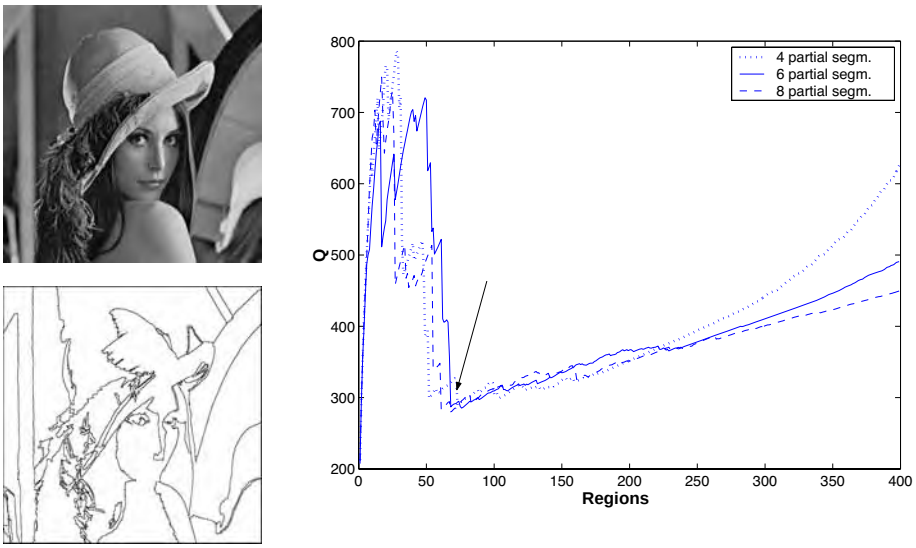


Fig. 11. Segmentation of the image Lena: original image, result of the segmentation and Q compared to the number of regions.

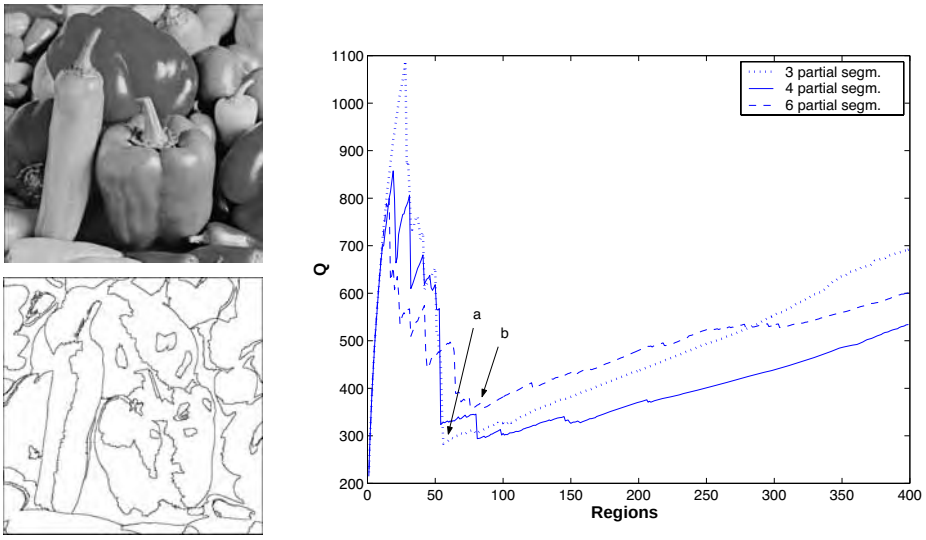


Fig. 12. Segmentation of the image Peppers: original image, result of the segmentation and Q compared to the number of regions.

6. Conclusions

In this work a parallel framework for image segmentation using region based techniques is presented. The algorithm is based on performing several segmentations of the same image using a parallel region-based algorithm. Moreover these segmentations are also obtained in parallel. This way, our proposal presents a two-level parallel layout. Next, an oversegmented image that collects all the information from the previous segmentations is created. A region-merging algorithm, developed previously by the authors, is then applied to this oversegmented image. A relevant aspect is that the information obtained from the partial segmentations will, in fact, guide the merging process, in such a way that the actual characteristics of each region or pixel are not taken into account.

The merging algorithm uses the concept of force of repulsion between neighboring pixels that indicates quantitatively their tendency to form part of different regions. The force of repulsion considers several situations in which any two neighboring pixels can be found in all the partial segmentations that are used to create the oversegmented image, including the shadowed zones. The shadowed zones are groups of pixels that differ in their intensity level a certain threshold from the region in which they could be included. Introducing this concept in the region-based algorithms, regions with low levels of homogeneity are avoided, improving the quality of the whole process. Note that, given that the shadowed zones are not treated by the algorithm, the information that can be extracted from these zones is minimum. As stopping criterion of the merging algorithm, we use a function to evaluate the quality of the segmentation.
The algorithm has been validated using several artificial and real images demonstrating the benefits of our proposal, and it was tested on a HP Superdome cluster.

7. Acknowledgements

This work was supported in part by the Ministry of Science of Spain through the TIN2004-07797-C02 project. The authors also thank the supercomputing facilities provided by CESGA.

8. References

Adams, R. and Bischof, L. (1994). Seeded region growing. *IEEE Transactions on Pattern Anal. Machine Intell.* Vol. 6, No. 16, pp. 641–647.

Blake, A. and Isard, M. (1998). *Active Contours*. Springer.

Borsotti, M.; Campadelli, P. and Schettini, R. (1998). Quantitative evaluation of color image segmentation results. *Pattern Recognition Letters*. Vol. 19, pp. 741–747.

Canny, J.F. (1986). A computational approach to edge-detection. *IEEE Trans. Pattern Anal. Machine Intell.* Vol. 8, pp. 679–698.

Chen, S.; Lin, W. and Chen, C. (1992). Split-and-merge image segmentation based on localized feature analysis and statistical tests. *CVGIP: Graph. Models Image Process*. Vol. 53, No. 5, pp. 457–475.

Fu, K. and Mui, J. (1981). A survey on image segmentation. *Pattern Recognition*, Vol. 13, No. 1, pp. 3–16.

Geman, D. and Geman, S. (1984). Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell*. Vol. 6, pp. 721–741.

Gibbons, A. (1984). *Algorithmic graph theory*. Cambridge University Press..

Gropp, W.; Lusk, E., Skjellum, A. (1994). *Using MPI Portable Parallel Programming with the Message Passing Interface*. The MIT Press.

Haralick, R. and Shapiro, L. (1985). Survey, image segmentation techniques. *Comput. Vision Graphics Image Process*. Vol. 29, No. 1, pp. 100–132.

Haris, K.; Efstratiadis, N.; Maglaveras, N. and Katsaggelos, A. K. (1998). Hybrid image segmentation using watersheds and fast region merging. *IEEE Trans. Image Process*. Vol. 7, No. 12, pp. 1684–1699.

Kanungo, T.; Dom, B.; Niblack, W. and Steele, D. (1994). A fast algorithm for MDL-based multi-band image segmentation. *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 609–616.

Kass, M.; Witkin, A. and Terzopoulos, D. (1988). Snakes: Active contour models. *International J. Computer Vision*, Vol. 1, No. 4, pp. 321–331.

Knuth, D. E. (1973). *The Art of Computer Programming: Sorting and Searching*. Addison-Wesley.

Liu, J. and Yang, Y. 1994. Multiresolution color image segmentation. *IEEE Transactions on Pattern Anal. Machine Intell*. Vol. 16, No. 7, pp. 689–700.

Mehnert, A. and Jackway, O. (1997). An improved seeded region growing algorithm. Pattern Recognition Letters . Vol. 27 , No. 10, pp. 1065–1071.

Pal, N. and Pal, S., (1993). A review on image segmentation techniques. *Pattern Recognition*, Vol. 26, No. 9, pp. 1277–1299.

Pichel, J. C.; Singh, D. E and Rivera, F. F. (2006). Image segmentation based on merging of sub-optimal segmentations. *Pattern Recognition Letters*, Vol. 27, No. 10, pp. 1105–1116.

Rosenfeld, A. and Kak, A. (1982). *Digital Picture Processing*. Academic Press, New York.

Saad, Y. (1996). *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company.

Sahoo, P.; Soltani, S.; Wong, A. and Chen, Y. (1988). A survey of thresholding techniques. *Comput. Vision Graphics Image Process*. Vol. 41, No. 1, pp. 233–260.

Wang, X. and Wang, H. (2004). Evolutionary Gibbs sampler for image segmentation. *Proceedings of International Conference on Image Processing*, pp. 3479-3482.

Zhu, S.C. and Yuille, A. (1996). Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Trans. Pattern Anal. Machine Intell*. Vol.18, No. 9, pp. 884–900.