UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament de Teoria del senyal i comunicacions

MULTIRESOLUTION IMAGE SEGMENTATION BASED ON COMPOUND RANDOM FIELDS: APPLICATION TO IMAGE CODING

Autor: Ferran Marqués
Director: Antoni Gasull

Barcelona, Diciembre de 1992

CHAPTER III

MONORESOLUTION SEGMENTATION

This chapter is devoted to the presentation of a new monoresolution technique for segmenting still, gray level images. This algorithm uses a compound random field as image model. The model is formed by a Strauss process at the lower level and by a set of white Gaussian Random Fields (GRFs) at the upper level. The chosen segmentation approach relies on Maximum a Posteriori (MAP) criteria and on deterministic relaxation techniques.

This segmentation technique sets the basis of a multiresolution approach which will be discussed in chapter IV. Therefore, it has to be seen as a first step towards a final algorithm rather than as a solution to the segmentation problem on itself. Besides, it has to be noticed that, within the algorithm, some assumptions are made having in mind its application for coding purposes. Nevertheless, when such assumptions are made, they are clearly underlined in order not to mix the general discussion with this special application.

III.1.- The image model

The chosen image model is a compound random field, whose upper level is a set of white GRFs whereas the lower level is a non-homogeneous Strauss process. Both models are discussed in the sequel.

III.1.1.- The upper level random field

The choice of white GRFs is owing to an intend of circumventing, as much as possible, the parameter estimation problem as well as that of reducing the computational load. A white GRF is totally characterised by its mean and variance (μ, σ). Therefore, only two parameters per region have to be computed. Note that, the minimum number of samples necessary to obtain non-zero estimates is two (of course, very poor estimates are obtained with such a little amount of samples). Furthermore, the fact of using simple white GRFs not only leads to more reliable estimates, but also speeds up the estimation procedure. The estimation of a white GRF can be performed by visiting only once each site in the random field, which is the optimum case.

The major problem when using white GRFs is that they cannot describe the spatial information of random fields. This shortcoming arises owing to the whiteness assumption (see Section II.1). However, when using white GRFs to characterise the upper level of a compound random field, this problem is partially overcome. Indeed, spatial information is introduced in the model by the underlying process. As it has been corroborated by several authors [56, 57, 61], this partial spatial information is enough in order to have fairly good image models.

Note that, when considering separately the upper level of the model, the realisation of the random field which maximises its probability function is the realisation that assigns to every pixel in the image a different label. That is, the probability is maximised when each single pixel is a region. In this case, the expression defining the probability of each GRF (see Section II.1) takes its maximum value. Given that the upper level of a compound random field is assumed to be formed by a set of independent random fields, its probability function is the product of the probability functions of its independent random fields. Therefore, in the above case, the global probability for the whole upper level is also maximum.

III.1.2.- The lower level random field

A Strauss process for the lower level of the model has been chosen for the following reason: as it has been highlighted in Section II.2.6, the "colour-blindness" characteristic of these processes makes them very suitable for modelling labelled images and partitions. Furthermore, studies carried out by other researchers [56, 57, 64] confirm this idea. These studies also report that a second-order neighbourhood system is large enough to characterise the behaviour of labelled images. Therefore, a non-homogeneous second-order Strauss process has been chosen for the lower level random field. Non-homogeneity has been introduced for avoiding boundary problems.

However, it should be emphasised that, in this work, the clique potentials of the Strauss process are not defined as usual; that is, β_1 is not taken to be equal to β_2. The parameters of the Strauss process have been chosen trying to control the behaviour of the region boundaries rather than the switching between regions. This idea comes from the work presented in [56]. There, the lower level random field is seen as a "line process" which can characterise contours of regions.

In order to visualise this concept, region boundaries are defined over a new lattice whose sites are situated midway between each vertical or horizontal pair of sites in the image lattice. Elements in this boundary lattice are set to zero if they are between two pixels with equal label (that is, if they are between two pixels belonging to the same region). In Figure III.1, the relation between these two lattices, as well as an example of realisation, is shown. Sites represented by circles belong to the image lattice, whereas sites represented by lines belong to the boundary one (darker boundary sites represent sites with non-zero value).

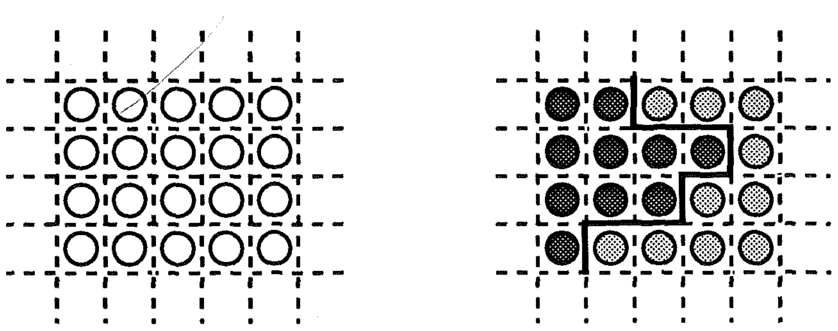


Fig. III.1.- Illustration of image and boundary lattices and example of realisation

Therefore, only cliques containing elements with different values (that is, belonging to different regions) are useful for characterising contours. As a result, instead of the usual definition given in Section II.2.6, zero potentials are assigned to cliques with equal value elements (homogeneous cliques). Note that single-site cliques are a special case of homogeneous cliques and, thus, have also zero potential ($\alpha = 0$). Non-homogeneous cliques have associated potentials whose values vary depending on the kind of clique c. In this way, (II.15) becomes

$$V_c(x) = \begin{cases} 0 & \text{if all } x_{ij} \text{ in c are equal} \\ \beta_c & \text{otherwise.} \end{cases} \tag{III.1}$$

Such a model has been used before in a contour relaxation technique [65].

In order to assign values to the clique potentials, the relation between cliques has to be studied. Since second-order neighbourhood systems η^2 are assumed, there are ten different clique configurations (see Figure II.4). From this set, zero potential has been already assigned to single-site cliques. Moreover, the potential definition should be invariant by rotation, if isotropy is assumed. Thus, the nine remaining configurations are classified into four types. This classification is shown in Figure III.2.

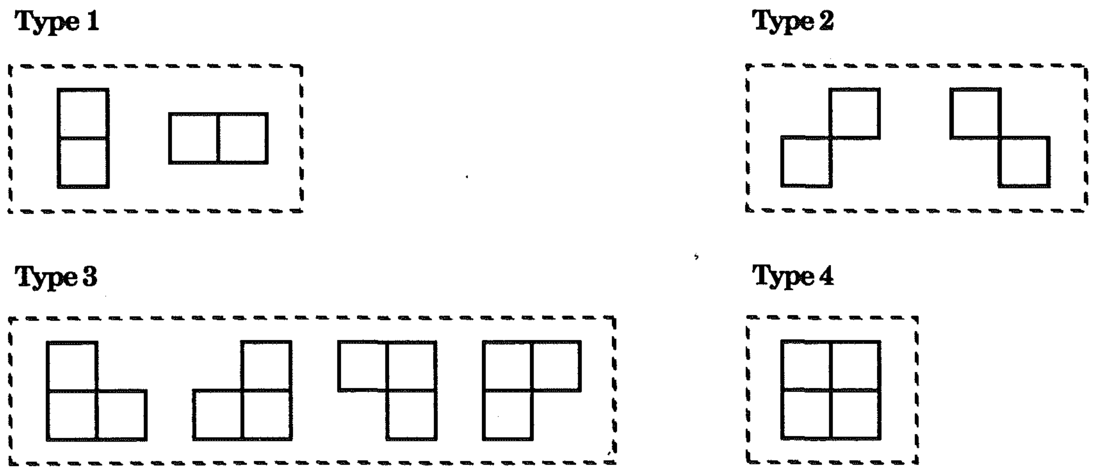
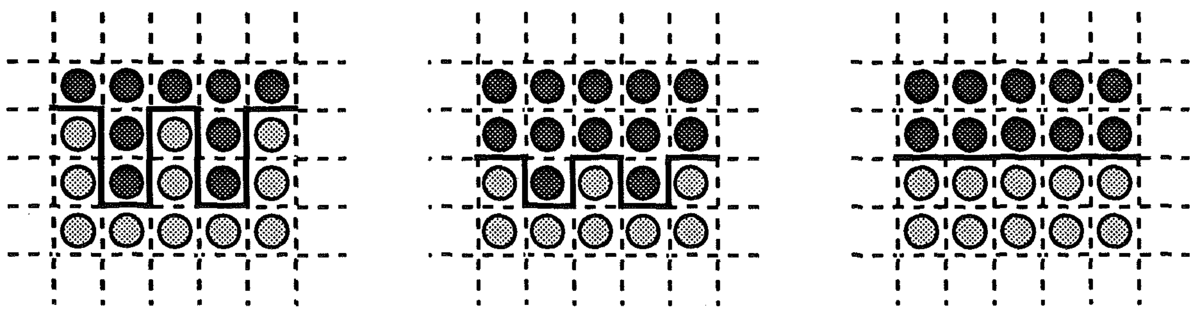


Fig. III.2.- Clique classification

Note that, although diagonal cliques may be understood as horizontal or vertical ones that have been rotated 45 degrees, they have been grouped into two different types. The reason for separating them is twofold. First, the equivalence between rotated cliques of type 1 and diagonal ones depends on the procedure of discretisation and, actually, the distance between pixels in diagonal cliques is larger than in cliques of type 1. Secondly, when dealing with contours, cliques of type 1 represent a different information than diagonal cliques.

The fact of having cliques of type 1 or 2 has a clear interpretation in the boundary lattice. In a clique of type 1, two elements from different regions are one in front of each other. This contact between regions produces a boundary element in the boundary lattice. On its turn, a clique of type 2 is produced by a diagonal difference. This kind of difference does not produce, by itself, any boundary element but marks the presence of a corner in some contour configurations. The combined behaviour of these two kinds of cliques is shown in the example of Figure III.3.



# Boundary el. : 13	# Boundary el. : 9	# Boundary el. : 5
# Cliques type 1: 13	# Cliques type 1: 9	# Cliques type 1: 5
# Cliques type 2: 12	# Cliques type 2: 8	# Cliques type 2: 8

Fig. III.3.- Behaviour of boundary elements and cliques of type 1 and 2

Three different realisations of a contour are shown in Figure III.3. As it can be seen, the smoother the contour, the lower the number of cliques of type 1 and the lower the number of boundary elements. Actually, when performing global analyses, the number of boundary elements is equal to the number of cliques of type 1. However, cliques of type 2 do not behave exactly the same, since the minimum amount of this kind of cliques is reached in a contour that is not the smoothest one (it does not present the lower number of boundary elements). Nevertheless, the number of cliques of type 2 does not increase when decreasing the number of boundary elements: it decreases or remains the same.

Comparing the behaviour of these two kind of cliques, diagonal cliques can appear to be useless for characterising contours, since the information provided by cliques of type 1 seems enough for such a task. Although this statement is true when dealing with the whole realisation of a boundary lattice (global analyses), this is not the case when handling only small zones of the lattice (local analyses). This is a relevant situation since, as it has been emphasised in chapter II, the MRF-GD equivalence allows performing global optimisation by carrying out local computations.

In Figure III.4, a local analysis of two similar situations is shown. In this figure, sites marked with a solid dot belong to the same region, while sites marked with a grey circle belong to other regions. In these cases, a local analysis relying on the neighbourhood of the central site based only on cliques of type 1 leads to the same result in both cases. However, a study taking into account diagonal cliques shows that both configurations are rather different. In this case, the more compact the region, the lower the number of boundary elements and the lower the number of diagonal cliques. Hence, the importance of keeping information about cliques of type 2.

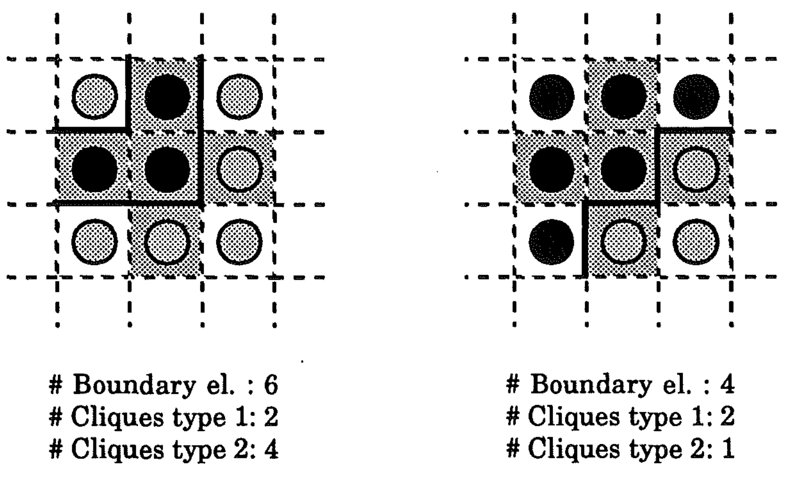
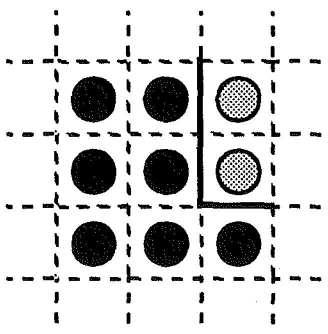


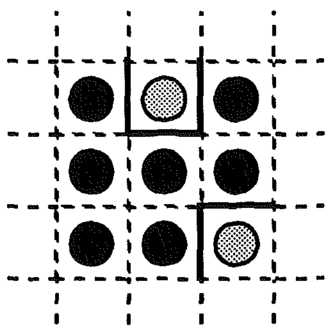
Fig. III.4.- Example of the usefulness of cliques of type 2

The usefulness of non-homogeneous cliques of type 3 and 4 for defining boundaries turns out to be not so clear. When dealing with a whole boundary lattice realisation, cliques of type 1 give enough information for characterising the boundaries and, therefore, cliques of type 3 and 4 are not necessary. Furthermore, when performing local analyses, their interpretation in terms of boundaries is not straightforward. These kinds of cliques may come from different configurations in the image lattice and, therefore, may describe different boundary behaviours. In order to give an interpretation, the information of these kinds of cliques has to be analysed jointly with the information provided by the presence of cliques of type 1 and 2. However, the way of mixing this information is not very intuitive and leads to cumbersome algorithms.

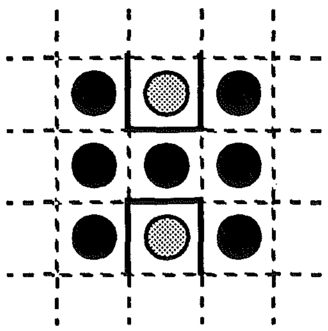
Problems arising from the analysis of the presence of cliques of type 3 and 4 in local configurations are shown in the example of Figure III.5. Two different cases are presented in this figure, both dealing with the local analysis of two configurations with equal number of cliques of type 1 and 2, but differing in the amount of cliques of type 3 and 4. In both cases, first configuration contours are smoother than those of the second configuration.



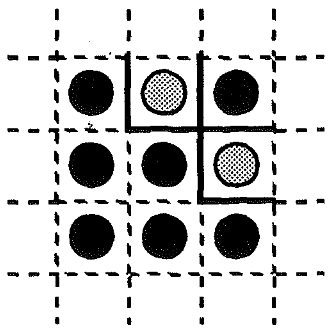
# Boundary el. : 3
# Cliques type 1: 1
# Cliques type 2: 1
# Cliques type 3: 5
# Cliques type 4: 2



# Boundary el. : 5
# Cliques type 1: 1
# Cliques type 2: 1
# Cliques type 3: 6
# Cliques type 4: 3



# Boundary el. : 6
# Cliques type 1: 2
# Cliques type 2: 0
# Cliques type 3: 8
# Cliques type 4: 4



# Boundary el. : 6
# Cliques type 1: 2
# Cliques type 2: 0
# Cliques type 3: 7
# Cliques type 4: 3

Fig. III.5.- Examples of the behaviour of cliques of type 3 and 4

Although, in the first case, the number of cliques of type 3 and 4 is lower in the first configuration, the situation is the inverse in the second case. In the second case, the number of boundary elements is equal in both configurations. However, the configuration with higher number of cliques of type 3 and 4 represents a contour simpler than that of less number of cliques. Therefore, a contour characterisation relying on the number of cliques of type 3 and 4 contained in each configuration does not provide any clear information. A method for characterising contours based on the relationship between the different amounts of cliques of each type present in the boundary may be possible. Nevertheless, the way of mixing this information is rather complicated and does not lead to any clear improvement.

It has been shown that a model only accounting for cliques of type 1 and 2 can provide with an intuitive and useful characterisation of contours. On the other hand, the use of cliques of type 3 and 4 does not seem to improve the performance of the model but, on the contrary, makes its computation much more complicated. Thus, cliques containing more than 2 sites (type 3 and 4) have been withdrawn (that is, their potentials have been set to zero). Further analyses on the performance of this image model will be performed in the following sections.

Note that, when taking separately the lower level of the model, the realisation of the random field which maximises the probability is the realisation which assigns a single region to the whole image. In this case, the whole image creates no cliques (the only boundary is the image itself). Therefore, independently of the potential values, the realisation has maximum probability.

Summarising, the chosen image model is a compound random field. The upper level of the model is composed of a set of white Gaussian random fields, while the lower level is a Strauss process. This MRF is defined on a second-order neighbourhood system and only non-homogeneous two-sites cliques have potentials different from zero. It is worth noticing that each random field maximises its own probability at opposite situations: the upper level when each pixel within the image is a region and the lower level when the whole image is a single region. However, the compound random field maximises its probability at a midway situation which yields the sought segmentation of the image.

III.2.- Segmentation algorithm

In order to segment images, the previous stochastic modelling can be used. That is, the image to be segmented can be assumed to be a realisation of the upper random field ($X = x$) and, the segmentation is performed by looking for the realisation of the lower random field ($Q = q$) which, more likely, has produced x. This idea can be expressed as a maximum a posteriori estimation $P(Q = q / X = x)$ that, by Bayes' rule, results in (II.17), which is reproduced here

$$P(Q = q / X = x) = \frac{P(Q = q)\, P(X = x / Q = q)}{P(X = x)} \quad . \qquad \text{(III.2)}$$

As it has been pointed out, maximising (III.2) leads to the same result as maximising the joint probability

$$P(Q = q, X = x) = P(Q = q)\, P(X = x / Q = q) \quad . \qquad \text{(III.3)}$$

Thus, (III.3) is maximised since the procedure is simpler. Given that Q is a Markov random field and that X is formed by a set of independent, white GRFs, (III.3) can be written as

$$P(Q = q, X = x) = \frac{1}{Z} \exp(-U(x)) \prod_{R_n} P(I_n / \mu_n, \sigma_n) , \qquad \text{(III.4)}$$

where R_n is the n-th region within the partition q, $\{ I_n \}$ is the set of elements in the image belonging to R_n, and μ_n and σ_n are the mean and the variance of the set $\{ I_n \}$, respectively. Assuming the potential definition given in Section III.1 and developing the expression for $P(X = x / Q = q)$, (III.4) results in

$$P(Q = q, X = x) = \frac{1}{Z} \exp(-\frac{1}{T} [k_1 V_1 + k_2 V_2]) \cdot$$

$$\cdot \prod_{R_n} \frac{1}{\sqrt{(2\Pi)^N \sigma_n^N}} \exp\left\{ - \frac{\sum (x - \mu_n)^2}{2 \sigma_n^2} \right\} , \qquad \text{(III.5)}$$

where k_1 and k_2 are the number of cliques of type 1 and type 2 (respectively) present in the partition q, N is the number of elements in the set $\{ I_n \}$, and the summation is performed over this set of pixels $\{ I_n \}$.

III.2.1.- Deterministic approach

In order to maximise (III.5) mainly two different techniques can be applied: stochastic and deterministic relaxation. When using stochastic relaxation approaches, the optimum solution is ensured to be reached. However, this kind of methods are computationally very demanding. On the other hand, although deterministic relaxation approaches may get trapped in local maxima or minima, they may achieve good quality results and in a feasible amount of time. The quality of the obtained results mainly depends on the initial conditions of the algorithm. Therefore, in this work, a deterministic approach has been chosen, having in mind the importance of setting the correct initial conditions.

The deterministic approach is a special case of the algorithm presented in [48] called iterated conditional modes (ICM). This algorithm seeks the optimum value of a function by introducing small changes in a given state, which is taken as starting solution. That is, at each step, a possible solution is devised by slightly changing some parameters in the former state. The new state is verified to be closer to the optimum than the former one. If this is the case, the new solution is assumed and the algorithm is re-applied. If not, other possible states are tried (that is, other small changes are performed in the former solution). This algorithm is repeated until no change improves the current solution and, thus, an optimum state is reached. Therefore, this algorithm performs the optimisation in a deterministic and iterative manner.

ICM can be seen as the last step in a simulated annealing (SA) process [62]. When finishing an annealing schedule (T = 0), SA carries out these changes in the solution which guarantee that the new state is closer to the optimum than the former one. In the SA case, the annealing schedule having been correctly performed, the global optimum is ensured to be reached. In the ICM case, the algorithm must be initialised with the right starting conditions in order not to get trapped in local optimum solutions.

It should be noticed that, in ICM, the way of selecting a new state from a given solution is very important: this selection marks the degree of optimality of the final solution. The final solution is said to be optimal when no other state, which can be reached from the current solution by means of small changes, improves the current result. Hence, the importance of correctly defining which "small changes" are allowed.

III.2.2.- Basic algorithm

In the case discussed here, the function to be maximised is the joint likelihood function of (III.5). In order to maximise this expression, the states that are checked correspond to different realisations of the lower level process ($Q = q$). The method for generating new states from a given partition must guarantee that the reached state is still a partition. This has been performed by means of the following procedure:

```
Given an initial partition ( $Q = q'$ )
Select an element of the image $(i, j)$
Change its label by that of a neighbour ($q_{ij} = q_{kl} : (k, l) \in \eta_{ij}$)
```

Note that by changing the label of an element by one of the labels of its neighbours, the basic constraints of a partition are still fulfilled, specially the connectivity of the regions. Therefore, the new state can be taken as a step towards the solution of the maximisation problem. It should be noticed that this method only changes labels of pixels lying on the boundary of regions (pixels belonging to, at least, one clique with potential different from zero). A pixel lying on the interior of a region has all its neighbours belonging to the same region and, therefore, to change its label leads to the original situation. Moreover, since a pixel can only get the label of one of its neighbours, new labels (that is, new regions) cannot be directly introduced by this procedure. However, the method is able to create new regions indirectly by splitting in two a former single region. This situation is illustrated in Figure III.6.

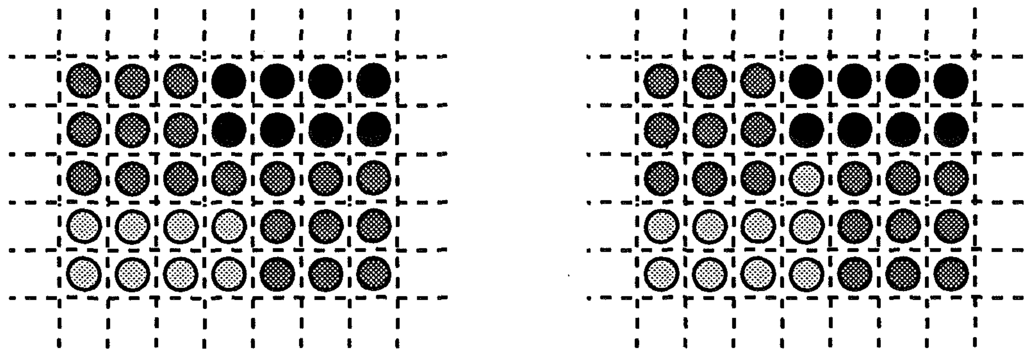


Fig. III.6.- Creation of a new region by splitting an initial one

This procedure for obtaining new possible states may seem very simple (it only checks pixels lying on the boundary of regions) and too constrained to the initial conditions (it is not able to generate directly new regions). However, as it will be shown latter, its iterative application allows reaching good local optimum states even when initialising the procedure with poor conditions. This is, in part, owing to the indirect creation of regions, which is, in most of the cases, good enough to overcome the possible drawbacks of the initial conditions. Furthermore, this procedure requires a low computational load, since only neighbour pixels are involved. Given that only neighbour pixels are used, only local computations are necessary for checking whether a new state must be either accepted or withdrawn.

Once an element of the image has been chosen to produce the new state, the procedure for selecting its new label is the following:

```
Given an initial partition ( $Q = q'$ )
Compute its joint likelihood function ( $P'$ )
Select an element of the image (i, j)
For all the neighbours of this pixel ($(k, l) \in \eta_{ij}$):
    Change the label of the selected point ( $q_{ij} = q_{kl}$ )
    Compute the new joint likelihood function ( $P''_{kl}$ )
If ( $\vee P''_{kl} > P'$ ) change the label ($q_{ij} = q_{kl}$) ( $\Rightarrow Q = q''$)
```

where $\vee$ stands for the max, $P' = P (Q = q' , X = x)$ and $P'' = P (Q = q'' , X = x)$.

The computation of the joint likelihood functions can be easily performed since it is not necessary to compute the whole expression, but only the parts that have changed from one situation to another. That is, by changing the label of a single pixel, only the clique configuration of its neighbourhood changes. Furthermore, in the product of (III.4) only two conditional probabilities have changed: those of the regions related to the former and to the current labels. Therefore, instead of calculating and comparing both probabilities (P' and P"), its ratio (P'/P") can be computed and compared to one. After removing the terms that do not change from one situation to another, the expression to check is:

$$\frac{P'}{P''} = \frac{\exp\left(-\frac{1}{T}[\, k_1' V_1 + k_2' V_2]\right) \cdot P(\, I_n' / \mu_n', \sigma_n' \,) \cdot P(\, I_m' / \mu_m', \sigma_m' \,)}{\exp\left(-\frac{1}{T}[k_1'' V_1 + k_2'' V_2]\right) \cdot P(\, I_n'' / \mu_n'', \sigma_n'' \,) \cdot P(\, I_m'' / \mu_m'', \sigma_m'' \,)} \qquad \text{(III.6)}$$

where k_1 and k_2 stand for the number of cliques of type 1 and 2, respectively, that there are in the neighbourhood of the selected pixel at each situation. Thus, only local computations are involved in the selection of new solutions for the optimisation problem.

In the above procedure for selecting a new solution there are still two steps which need further explanation. These steps are the computation of the joint likelihood of a given partition and the mechanism for selecting an element within the image so that the procedure is applied to it. This last step is very related to the way of performing the iteration of the algorithm.

III.2.3.- Computing the joint likelihood function

In Section III.2.2, it has been shown that rather than computing the whole joint likelihood function of (III.5), it is only necessary to calculate some of its terms, which are shown in (III.6). Terms dealing with the lower level of the model can be calculated directly since the number of cliques of type 1 and 2 (k_1 and k_2, respectively) can be exactly computed from the current realisation. The problem of giving values to the clique potentials and the temperature (V_1,V_2 and T) will be discussed latter in this chapter.

In order to calculate the conditional probabilities, the values of the mean and the variance of each region have to be known. As it has been emphasised in the previous chapters, assuming the a priori knowledge of the exact number of regions within the image is a pure academic case. This situation is even less realistic if the values of the parameters of the model characterising each one of the regions are assumed to be known as well. Therefore, they have to be estimated, which is one of the main problems of these segmentation techniques.

In this work, since each region is characterised by a white Gaussian random field, the problem of estimating the model parameters is not so difficult. White GRFs are totally defined by means of two parameters, which can be estimated even with two samples from the random field (of course, in this case, a poor estimate of the model parameters is achieved). Furthermore, the parameter estimation can be performed using only once the value of each sample, which speeds up the estimation procedure. A maximum likelihood criterion has been used for carrying out the estimations:

$$\hat{\mu}_n = \frac{1}{N}\sum_{R_n} x_{ij} \qquad (\hat{\sigma}_n)^2 = \frac{1}{N}\sum_{R_n} (x_{ij} - \hat{\mu}_n)^2 = \frac{1}{N}\sum_{R_n} x_{ij}^2 - (\hat{\mu}_n)^2, \qquad \text{(III.7)}$$

where N is the number of point of the region. Note that these estimations can be easily updated if the data change. Actually, very simple algorithms for updating the estimations based on the recursive introduction (elimination) of single data can be used.

Making use of the expression of the conditional probabilities of (III.5) and introducing the estimations of (III.7), these conditional probabilities can be written as:

$$P\,(\,I_n\,/\,\hat{\mu}_n,\,\hat{\sigma}_n) = \frac{1}{\sqrt{(2\Pi)^N\,(\hat{\sigma}_n)^2}}\;\exp\left\{-\frac{\sum(x-\hat{\mu}_n)^2}{2\,(\hat{\sigma}_n)^2}\right\}. \quad \text{(III.8)}$$

Taking into account the definition of $(\hat{\sigma}_n)^2$ given in (III.7), (III.8) results in:

$$P\,(\,I_n\,/\,\hat{\mu}_n,\,\hat{\sigma}_n) = \frac{1}{\sqrt{(2\Pi)^N\,(\hat{\sigma}_n)^N}}\;\exp\left\{-\frac{N}{2}\right\}. \quad \text{(III.9)}$$

Now, the conditional probabilities of (III.6) can be substituted by the expression given in (III.9). It should be noticed that, going from the situation (q') having (R_n', R_m') to (q") having (R_n'', R_m''), the total number of pixels has not changed. That is, N'+ M' = N"+ M", where M is the number of elements in R_m. Thus, (III.6) can be written as:

$$\frac{P'}{P''} = \exp\left\{-\frac{1}{T}\left[\,(k_1'-k_1'')V_1+(k_2'-k_2'')V_2\,\right]\right\}\cdot\frac{(\hat{\sigma}_n'')^{N''}\,(\hat{\sigma}_m'')^{M''}}{(\hat{\sigma}_n')^{N'}\,(\hat{\sigma}_m')^{M'}}. \quad \text{(III.10)}$$

Actually, instead of computing the ratio between probabilities, the logarithm of this ratio can be computed. The decision whether the new state should be accepted or not is performed by checking the sign of the logarithm. In this way, the expression to be calculated is:

$$\begin{aligned}\ln\left[\frac{P'}{P''}\right] = {} & -\frac{1}{T}\left[\,(k_1'-k_1'')V_1+(k_2'-k_2'')V_2\,\right] + N''\ln\left[\,\hat{\sigma}_n''\,\right] + M''\ln\left[\,\hat{\sigma}_m''\,\right] \\ & - N'\ln\left[\,\hat{\sigma}_n'\,\right] - M'\ln\left[\,\hat{\sigma}_m'\,\right]. \qquad \text{(III.11)}\end{aligned}$$

As mentioned above, the computation of terms related to the lower level of the model is very easy to carry out. In addition, the estimation of the variances can also be performed in a very fast way. Since from the initial state (q') to the final one (q") only one pixel has changed its label, the estimation of the parameters at the final state can be performed very quickly in an iterative way relying on the parameters of the initial state. If the new state is accepted (that is, it represents a realisation which increases the joint likelihood), the updating of the model parameter estimation does not imply any new computation. Indeed, parameters of state (q") have already been estimated when checking its validity.

Therefore, the proposed method leads to a fast algorithm for verifying the goodness of the new state. This characteristic of the algorithm not only comes from the fact that only local changes are allowed but also from the kind of model and from the parameter estimation criterion.

III.2.4.- Selection of elements from the image

From the basic algorithm presented in Section III.2.2, the step dealing with the way for selecting elements (that is, pixels) from the image has still to be discussed. This step is very related to the manner of performing the iteration of the algorithm, as well as to the precise moment in which the updating of the estimations is carried out. The selecting mechanism, in order to converge to a local optimum solution, has to ensure that all elements in the image will be checked at least once. Depending on the procedure for visiting each of the pixels, algorithms may present different characteristics. Here, three different procedures are discussed.

Best choice algorithm

Given an initial partition of the image, all the possible label changes for all the points are first checked. Records of the joint likelihood increments representing each one of these changes are kept. Once the whole image has been visited, the change of label which increases the most the joint likelihood function is performed. After carrying out the label change, the estimates of the model parameters are updated and the list of increments resorted. This procedure is iterated until convergence. The convergence of the method is guarantied, since only label changes increasing the joint likelihood function are performed.

Note that this algorithm seeks the label change that most increases the joint likelihood function in the whole image at each step. Thus, this method uses the procedure for checking if a new state should be accepted when selecting the point that has to be visited. Given that, no new checking step has to be performed after a point has been selected. However, since initially the whole image has to be tested and, at each step, a list of increments has to be sorted, the method is very slow.

Non-recursive (parallel) algorithm

For a given partition, all its points are selected at the same time. Thus, all label changes, initially increasing the joint likelihood function, are performed. After the label changing step, model parameters are updated. Since the convergence of this method is not guarantied, it must be iterated only a fixed number of times. Convergence is not guarantied given that label changes that do not increase the current likelihood may occur. Label changes decreasing the joint probability happen owing to the fact that model parameters are only updated after the whole set of changes is performed. Therefore, changes that isollately increase the joint probability function, may decrease it when performed together. This decrease can either be fixed in the following iteration or result in an oscillation.

This algorithm seeks the set of label changes that increases the joint likelihood in a given step. Therefore, its main advantage is that is highly parallel. Since a label change is carried out without taking into account the rest of the labels, different processors can be assigned to different zones within the image. Note that this method implemented in a sequential machine is rather slow, as, for approaching the local optimum state, several iterations have to be done.

Recursive (sequential) algorithm

In the recursive algorithm, pixels are selected by a scanning procedure. The image is scanned line by line, from top to bottom and from left to right (normal raster procedure). After each label change, model parameters are updated. The algorithm is iterated until no label change is performed in a whole scanning. Given that only label changes increasing the joint probability function are performed and that model parameters are updated after each label change, the recursive algorithm ensures convergence. In addition, since the selecting mechanism is performed by a simple scanning, the algorithm is very fast.

Due to the updating procedure, during the scanning, new labels are checked taking into account the label changes that have been performed until this moment. Thus, priority is given to changes performed in the direction of the scanning. This situation is illustrated in Figure III.7, where three partitions, representing three consecutive steps in the optimisation procedure, are shown. For this example, model parameters are assumed to allow label changes if the neighbourhood of the pixel under study contains five or more pixels with labels different from that of the central pixel. The initial state differs from the final one (which is supposed to be the partition with maximum likelihood) in only 5 pixels. Note that after the first scanning (second state), all label changes in the scanning direction have been performed. That is, all possible solid dots have been removed. On the contrary, in order to carry out the label changes related to gray dots, a new scanning is necessary. Note that, if the scanning is carried out in the opposite direction (from bottom to top and right to left) label changes related with gray dots are performed in the first scanning. However, in order to perform the label changes associated with solid dots, three different scannings are necessary.

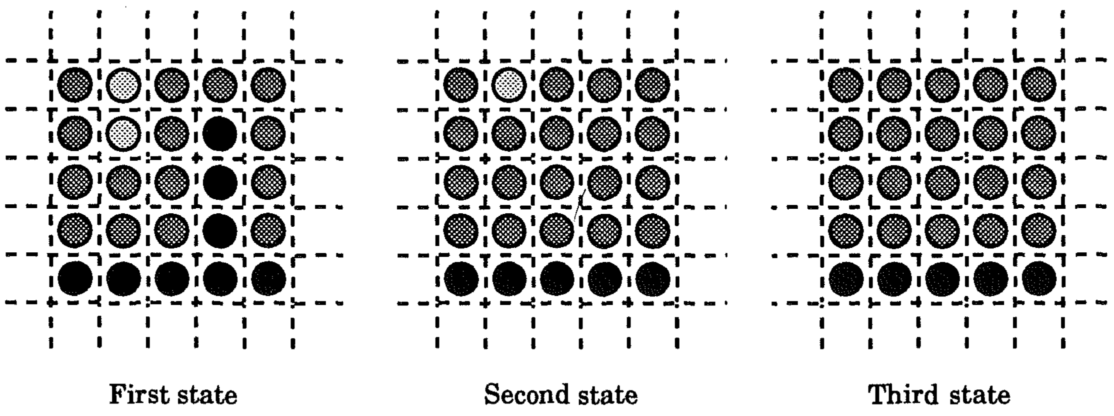


Fig. III.7.- Example of penalising of the scanning direction

To speed up the algorithm, instead of iterating the procedure after each scanning, four different scannings are used. Once the normal raster procedure is performed, the inverse scanning is carried out. That is, the image is scanned from bottom to top and from right to left. Analogous scannings are performed from top to bottom and right to left as well as from bottom to top and from left to right. In this way, the priority given to a direction is reduced. However priorities are not totally removed, since the direction of the scanning carried out in the last place is still penalised with respect to the others.

This four-scanning scheme requires an iterative procedure a little bit more complicated than the former ones. It would be useless to keep the procedure going on if, after one of the scannings, no label change had been performed. Therefore, after each single scanning, the procedure has to be allowed to stop. On the other hand, after completing the four scannings, the algorithm has to be able to repeat the whole procedure again. The final algorithm is depicted in Figure III.8. Throughout the sequel, when referring to the recursive algorithm, the term will refer to the whole system shown in this figure.

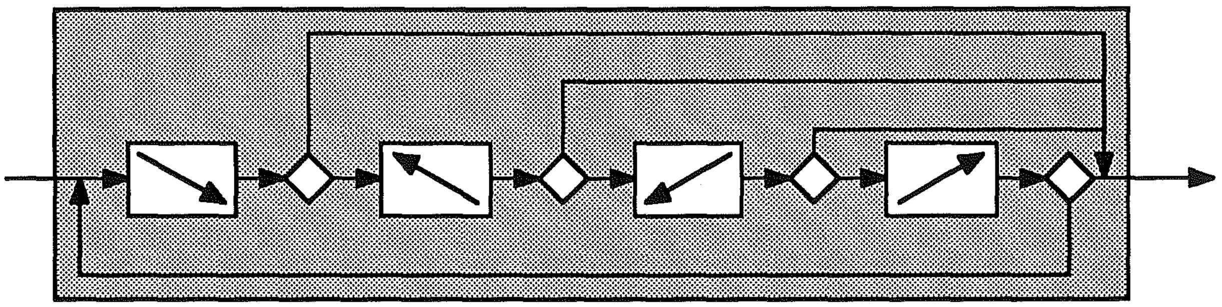


Fig. III.8.- Block diagram of the recursive algorithm

III.3.- Study of the basic segmentation algorithm

There are several features of the segmentation algorithm presented in Section III.2 which need further study. For instance, the best method for selecting pixels to create new states, the way of setting values to the lower level model parameters (V_1, V_2 and T) or the algorithm limitations (its robustness with respect to the initial segmentation or the range of images on which the algorithm can be applied).

To illustrate the results of this study, the image called Cameraman will mainly be used (see Figure III.9). This image has been chosen for pedagogical purposes but the results and conclusions are general. It contains very different kinds of regions in both shape and interior: textured areas (as the grass), soft gradients in large areas (as the sky), small but relevant gradients (as the pocket of the coat or the small building), small details (as the face or the camera), square regions (as the buildings), thin and large regions with different main directions (as the tubes or the legs of the tripod).

Regarding the initial segmentations that will be used throughout this study, it has to be emphasised that they have been obtained by using different segmentation algorithms. At each moment, the most adequate segmentation technique has been chosen in order to highlight some desired features of the algorithm. Nevertheless, a bottom-up algorithm for obtaining good initial segmentations has been proposed in [66].

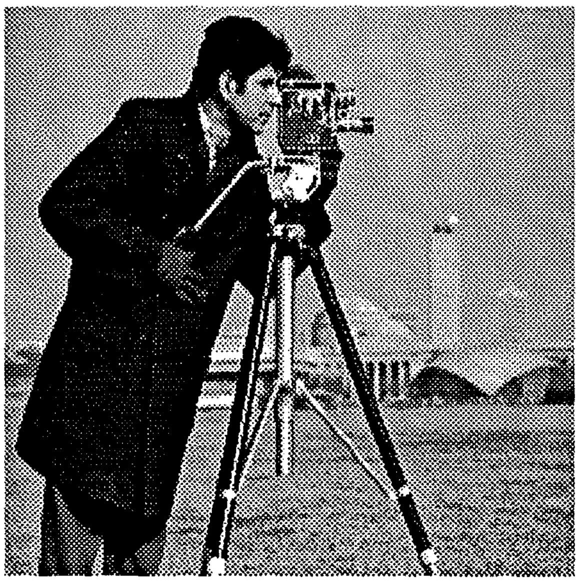


Fig. III.9.- The original test image Cameraman (256x256 pixels)

In the three following sections, the initial segmentation of Cameraman shown in Figure III.10 has been used. This segmentation contains 662 regions. The main characteristics of this segmentation is that it presents large overgrown regions (as the sky, which has been detected as a single region), as well as small ones (for instance, regions in the grass which have taken pixels from the tripod). In addition, there are oversegmented areas, as the grass, and some details have been blurred or overlooked, as in the face.

From this segmentation, as well as from the rest of the segmented images that are shown in this work, regions containing few pixels have been removed. This cleaning step has been done mainly for two reasons. Regions containing few pixels do not usually represent real objects or zones of the image but noisy areas. Moreover, the visual quality of the segmentation does not improve when these small regions are kept in the final result (they are unnoticeable).

When aiming at coding purposes, these regions increase the amount of bits necessary to code the image, while not increasing the quality of the final result. An example of the effect of small regions is illustrated in Figure III.11, where a segmentation of Cameraman is shown, before and after removing all the regions smaller than 10 pixels. As it can be seen in the mosaic images, their visual quality is almost the same.

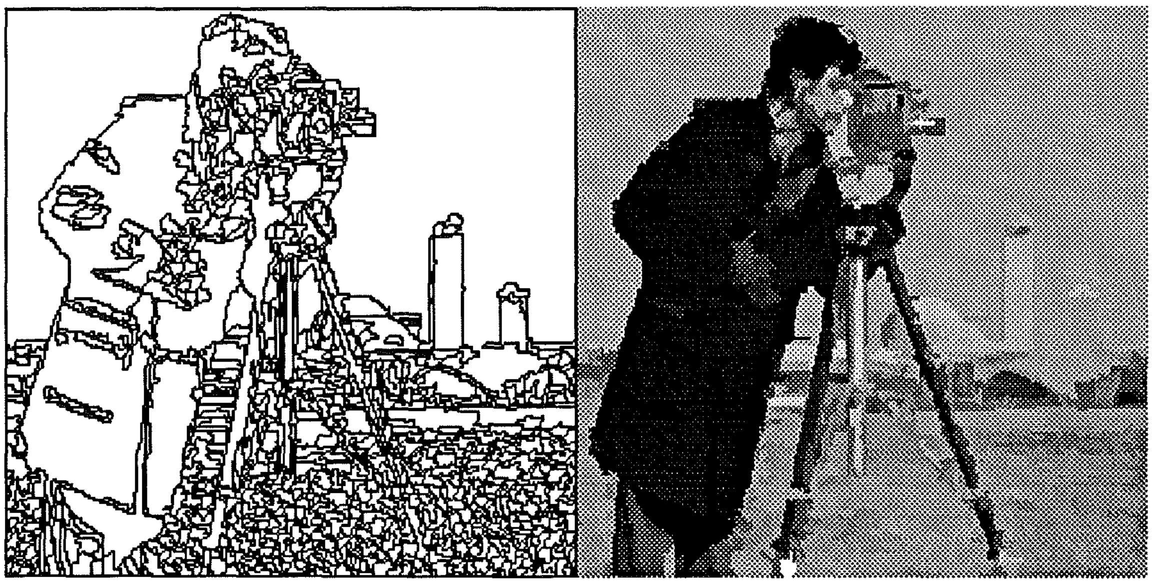


Fig. III.10.- Initial segmentation (contours and mosaic) of Cameraman: 662 regions

Note that in the example of Figure III.11, the number of final regions is less than half the number of regions before the cleaning step. This large amount of initial regions is mainly related to isolated, single pixel regions which are located beside real contours. The basic algorithm does not directly remove this kind of regions, since the computation of the parameters estimate in a single pixel region cannot be performed. That is, the variance estimation of (III.7) performed on a single element is always zero and, thus, the computation of the ratio between probabilities of (III.11) cannot be carried out. Therefore, the removal of these regions has to be performed in a posterior cleaning step.

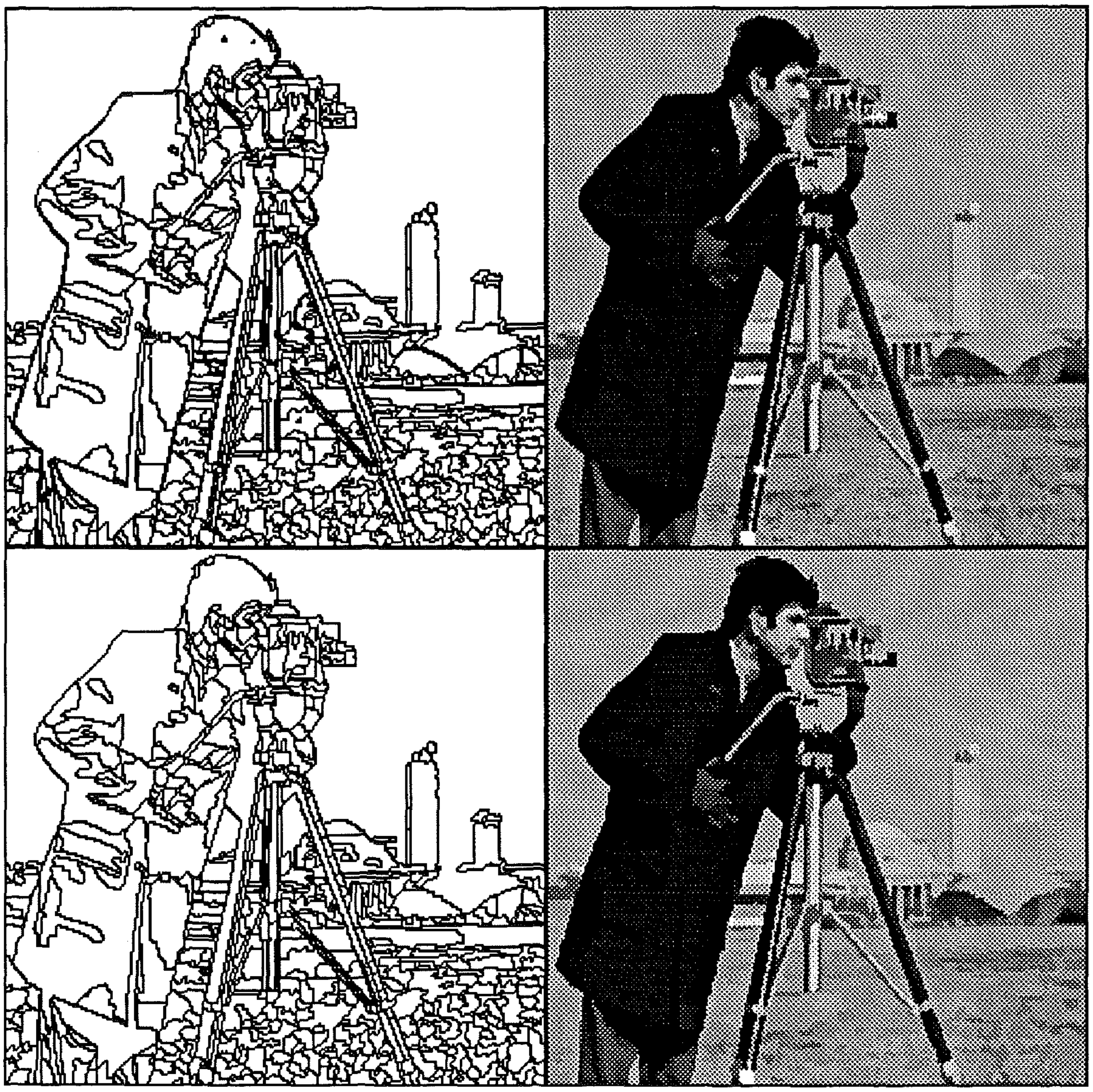


Fig. III.11.- Example of cleaning step: top 1356 regions, bottom 605 regions

III.3.1.- Comparison among the selecting element techniques

The performance of the techniques presented in Section III.2.4 (best change, parallel and sequential) will be now compared. Results obtained by means of these three techniques, when using the same parameters and the same initial segmentation, are very alike (see Figure III.12). However, the computational load of these algorithms in a sequential machine is rather different. Giving a unitary computational load to the sequential algorithm, the "best change" algorithm would have 100 as computational load, whereas the parallel algorithm would have 5. This feature discards the use of the "best change" algorithm.

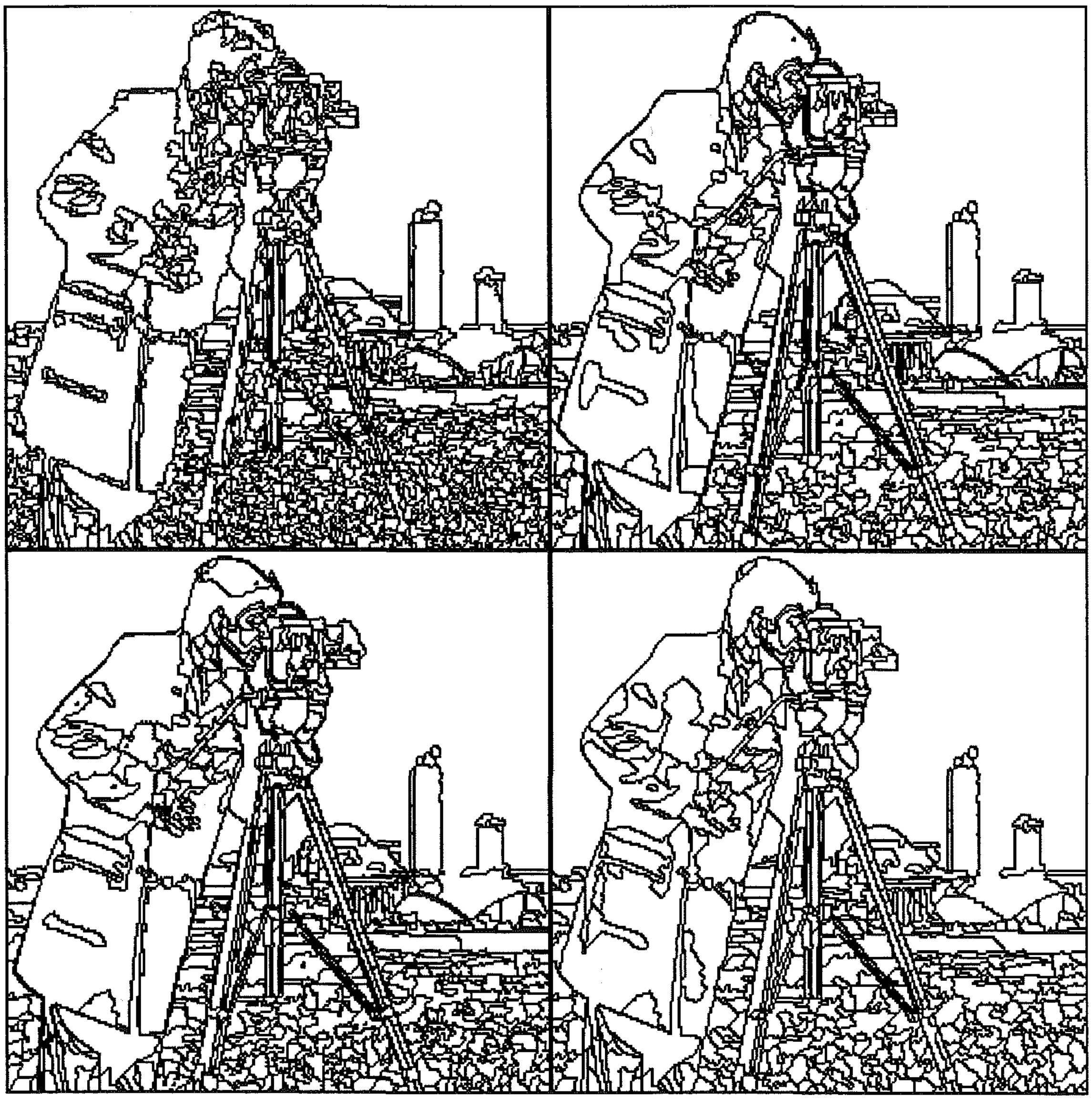


Fig. III.12.- Initial segmentation and results applying the three selecting techniques: best choice, parallel and sequential algorithms

On its turn, the parallel algorithm presents a convergence problem. Since label changes that do not increase the current likelihood can be performed, convergence is not guarantied. However, it has been observed that although single label changes may not increase the conditional likelihood function, the whole set of changes does increase it. Not having the convergence guarantied means, in this context, that the algorithm may oscillate. In the experiments performed, oscillations have only appeared in a few cases, being noticeable at the very end of the optimisation procedure. Therefore, when the likelihood values oscillate, the algorithm has already achieved a good segmentation.

The behaviour of the parallel algorithm for the result shown in Figure III.12 is illustrated in Figure III.13. A graphic showing, at each iteration, the number of label changes and the evolution of the conditional likelihood is presented. The vertical axis of the graphic is scaled with respect to the number of label changes. Oscillations can be better observed in the label changes curve than in the likelihood graphic, given that oscillations on the likelihood values are of very small amplitude. Oscillations can therefore be detected and the procedure stopped at this point.

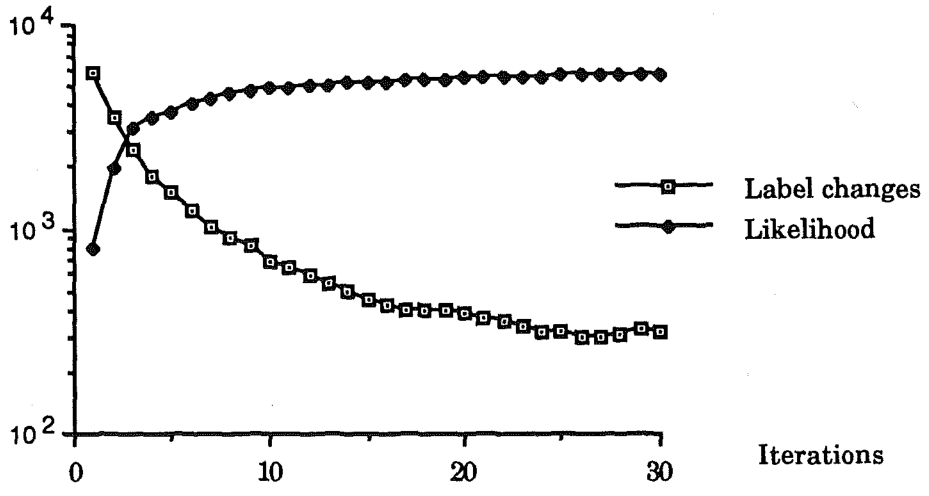


Fig. III.13.- Behaviour of the parallel algorithm

Given the above comments, the parallel algorithm can be a good choice if a parallel machine is available. However, the recursive algorithm has been always used in this work, given that both algorithms achieve similar results and that all the algorithms have been implemented in a serial machine.

III.3.2.- Study of the influence of the temperature

As it has been emphasised throughout this and the anterior chapter, the estimation of the model parameters is a difficult task that, when possible, should be circumvented. In the upper level case, this problem has been partially overcome by assuming white GRFs, whose parameters estimation can be easily performed. For the lower level case, some studies on the model behaviour characterisation when changing the parameter values have been performed. These studies are aimed at the possibility of not having to estimate the parameters while performing the segmentation but fixing them before hand for the whole procedure.

The three parameters describing the Strauss process defined for modelling the lower level (T, V_1, V_2) can be reduced to two (T*, V*), given that:

$$\frac{1}{T}[k_1V_1 + k_2V_2] = \frac{V_1}{T}[k_1 + k_2\frac{V_2}{V_1}] = \frac{1}{T^*}[k_1 + k_2V^*], \qquad \text{(III.12)}$$

where both parameters can take values from the [0, ∞) interval. In this section, the influence of T* is analysed and some experiments are carried out, using, for all of them, the same conditions. That is, the initial segmentation of Figure III.10, the recursive algorithm and potential ratio V* = 0.5. The choice of this value for V* will be justified in the following section.

The parameter T* controls the relative importance between both levels (upper and lower) within the whole compound random field. When T* increases (decreases), the term related to the lower level in the joint likelihood losses (gains) importance in front of the term related to the upper level. Thus, the maximum of the likelihood function moves towards the maximum of the upper (lower) level model. In this case, the compound random field takes more (less) into account gray level information than contour information and therefore, the final segmentation contains more (less) regions. However, there is a large range of values for this parameter yielding similar good results. This behaviour can be seen in Figure III.14 where the variation of the final number of regions with respect to different values of T* is plotted. It can be seen how, as the value of T* increases, so does the number of regions.

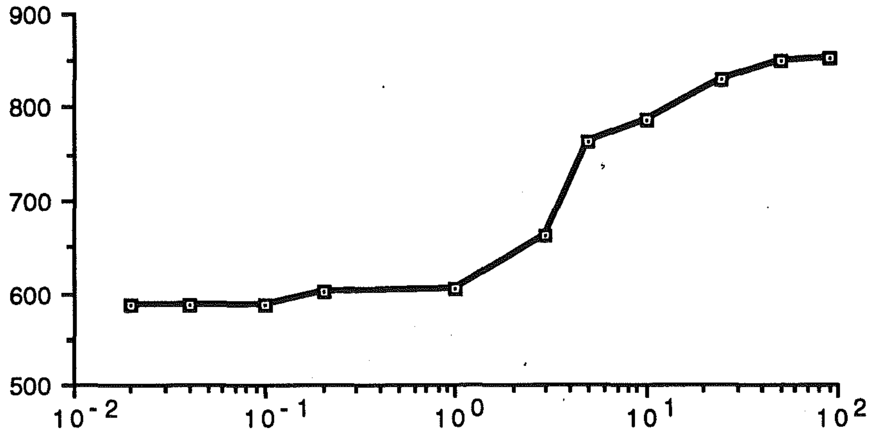


Fig. III.14.- Final number of regions versus T*

Figure III.15 shows some segmentation results achieved with the same algorithm using different values of T*. These values are presented in Table III.1, whose first row shows the location of the images within the figure. Note that contours become smoother and regions become more square-like as T* increases. For small values of T*, the importance of the texture model is so big that the algorithm even splits the sky into several regions.

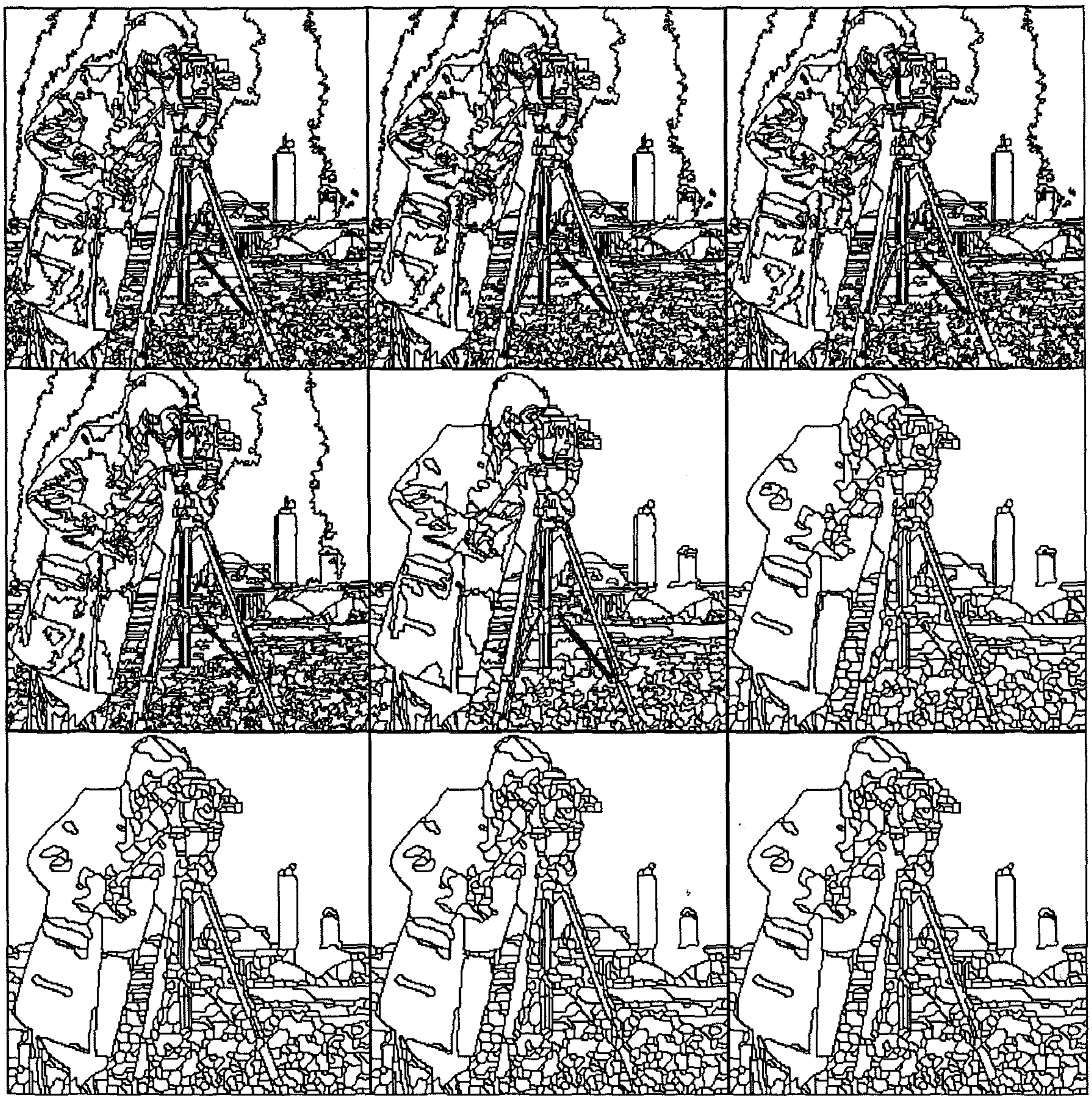


Fig. III.15.- Segmentations obtained by varying the value of T^*

On the other hand, when T* increases, the relative importance of the contour model makes the highest building appear with rectangular shape, losing the original oscillations of its contours. It has to be highlighted that, even for the smallest and the largest values of T* in this set, final segmentations improve with respect to the initial segmentation. In all of them, details in the man and in the camera, as well as the real shape of the tripod, are recovered.

(i, j)	(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)
T*	50	25	10	5	1	0.2	0.1	0.04	0.02
# Reg.	852	831	786	764	605	601	587	586	586

Table III.1.- Data from segmentations of Figure III.15

It has also to be noticed that the variation of the segmentation results is not linear with respect to T*. When $T^* \rightarrow 0$, the importance of the lower level with respect to the upper level saturates and segmentation results remain stable. A similar effect happens when $T^* \rightarrow \infty$, but the saturation is slower (note that in the data of Table I, saturation is not still reached). However, saturation states do not correspond with the expected optimum points (segmentations with only a region when $T^* \rightarrow 0$, or segmentations where each pixel is a single region when $T^* \rightarrow \infty$).

The reason for the algorithm not reaching these optima is different in each case. For small values of T*, the algorithm gets trapped in local optima, due to its deterministic basis. That is, the algorithm depends on the initial conditions. For large values of T*, the final state does not present single pixel regions given that the algorithm is not allowed to create new regions directly.

Similar experiments have been carried out with different images. Comparing these data, it has been concluded that best results are achieved when the value of the parameter T* is set around 1. Therefore, for the following experiments, $T^* = 1$ is used.

III.3.3.- Study of the influence of the potential ratio

The parameter V* of (III.12) controls the relation between potentials of cliques of type 1 and cliques of type 2 ($V^* = V_2 / V_1$). To change this relation results in giving priority or penalising some kind of contours with respect to others. $V^* \rightarrow 0$ is the case presented in Section III.1.2 (Figure III.4) in which cliques of type 2 are not taken into account. As it has been pointed out, in a global analysis this situation does not arise any problem. However, when performing local analyses, this choice of potentials leads to overlooking some contour configurations. That is, the local compactness of regions is not checked. Therefore, regions having zones of small width are more likely to appear.

The case of V* → ∞ translates into a reduction of the importance of cliques of type 1. This situation corresponds with the case discussed in Section III.1.2 (Figure III.3). There, it has been shown that to give priority to cliques of type 2, withdrawing the information of cliques of type 1, results in a model producing wringing contours. In order to build a model for smooth contours, both potentials should have similar values, and the main importance of cliques of type 1 has to be taken into account. That leads to $V^* \in [0.25, 1]$.

The variation of the final number of regions with respect to the value of V* is plotted in the graphic of Figure III.16. The range of variation of the vertical axis has been taken equal to that of Figure III.14. In this way, the small dependence of the final number of regions with respect to V* comparing with with respect to T* is illustrated. It can be seen how, as the value of V* changes, the number of regions does not vary very much, having, in this case, its minimum value around V* = 1.

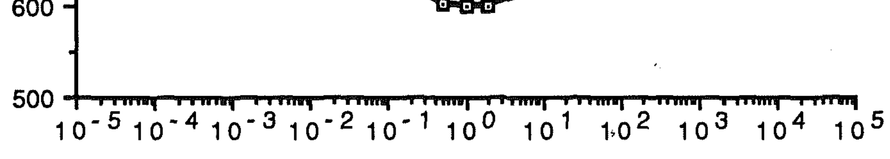


Fig. III.16.- Final number of regions versus V*

These different situations can be observed in Figure III.17, where the results yielded by the same algorithm when varying the value of V* are depicted. Data from these images can be seen in Table III.2. Note that for the whole range of values of V*, resulting segmentations improve very much the initial segmentation. As in the previous study, small details and complicated shapes that are not present in the initial segmentation are recovered.

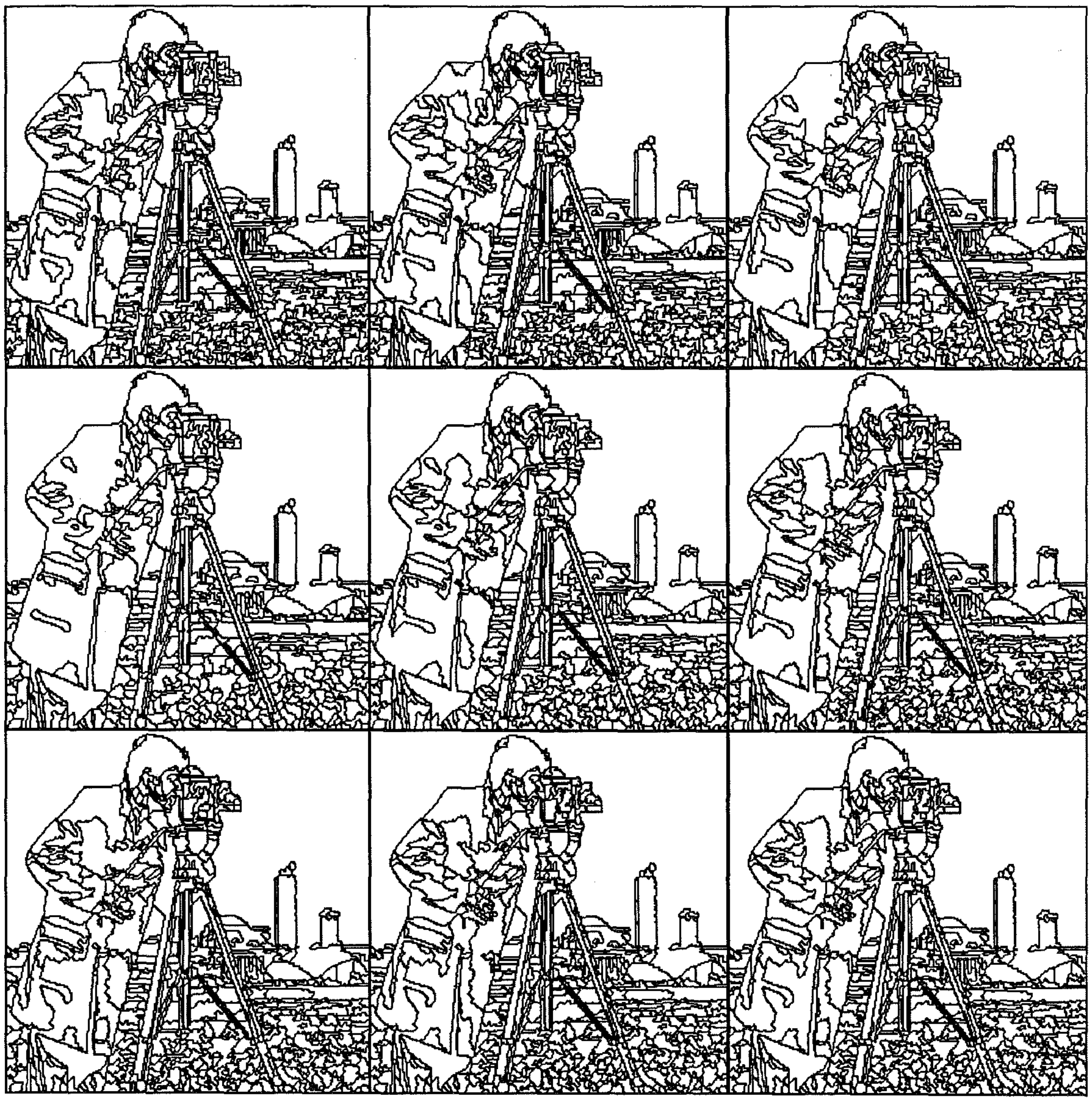


Fig. III.17.- Segmentations obtained by varying the value of V^*

(i, j)	(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)
V*	0	0.1	0.5	1	2	10	20	100	∞
# Reg.	653	636	605	601	601	621	620	619	629

Table III.2.- Data from segmentations of Figure III.17

However, the local form of contours varies from one segmentation to another. In the first results (small values of V*), several regions present elongations of one pixel width (see regions in the coat, the grass and the small building), whereas in the last ones (large values of V*), contours are less smooth and in some places rather wrung. It is worth noticing that this effect not only appears in regions that may not have a clear contour (regions within the coat or the grass) but also in regions that should have a very smooth shape (the clearest case is the right hand-side of the highest building).

To illustrate this behaviour, two parameters have been computed on the above set of segmentations. The first parameter measures the compactness of regions. This parameter is the amount of elongations of one pixel width appearing in the different regions. The second parameter measures the local smoothness of the boundaries. This feature has been quantified by computing the amount of three pixel length straight contours. Both parameters are plotted in the graphic of Figure III.18. Values plotted for the second parameter are the real values divided by ten, in order to make the dynamic of both parameters similar.

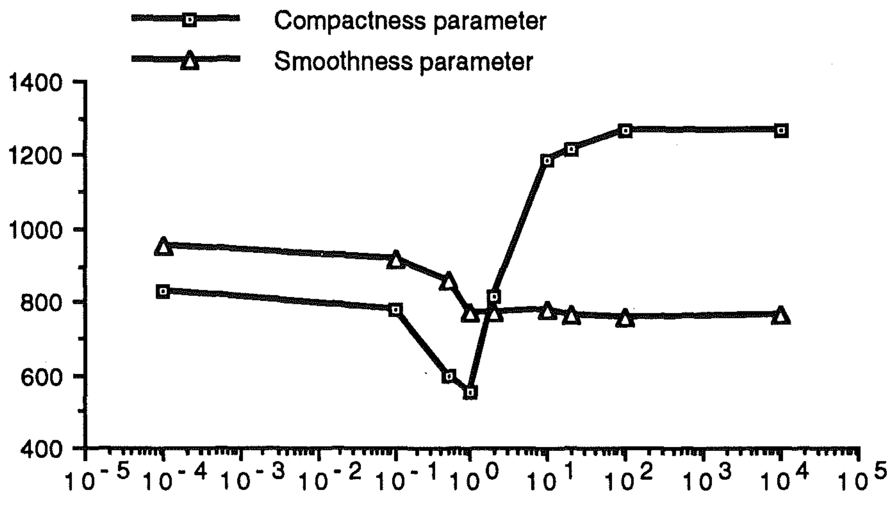


Fig. III.18.- Compactness and smoothness parameters versus V*

In order to fix the value of V*, two different concepts can be applied. First, boundaries of regions should conform to the boundaries of natural objects. Secondly, when using image segmentations for coding purposes, contours should be as easy as possible to code. Both concepts leads to a smooth contour solution without thin elongations. This kind of contours can be achieved by giving to the parameter V* a value into the above range. Several experiments performed with a large set of images show that a good choice for this parameter is V* = 0.5.

III.3.4.- Study of the influence of the initial segmentation

In the former sections (especially in Section III.3.2), the proposed image model and segmentation algorithm have been shown to achieve good results. Furthermore, the quality of the final results always improves that of the initial segmentations. However, it has been also highlighted that the quality of final segmentations is constrained by the initial segmentation. Here, different initial situations are studied so that their influence in the final result is analysed.

In the first row of Figure III.19, three different initial segmentations are shown. These segmentations have been carried out with an algorithm which has a tendency to produce vertical regions. Moreover, the algorithm uses a very simple region model, which results in segmentations with overgrown regions as well as oversegmented areas. The difference among the three examples is that, in each case, the algorithm has been tuned to yield a different number of regions (see Table III.3). In the second row of Figure III.19 the contour image of the final segmentations achieved with T* = 1 and V* = 0.5 can be observed. Note that, in all cases, final results improve the quality of initial segmentations. For instance, objects that are not present in initial segmentations, do appear in final segmentations (e. g.: the domes or the hand of the man).

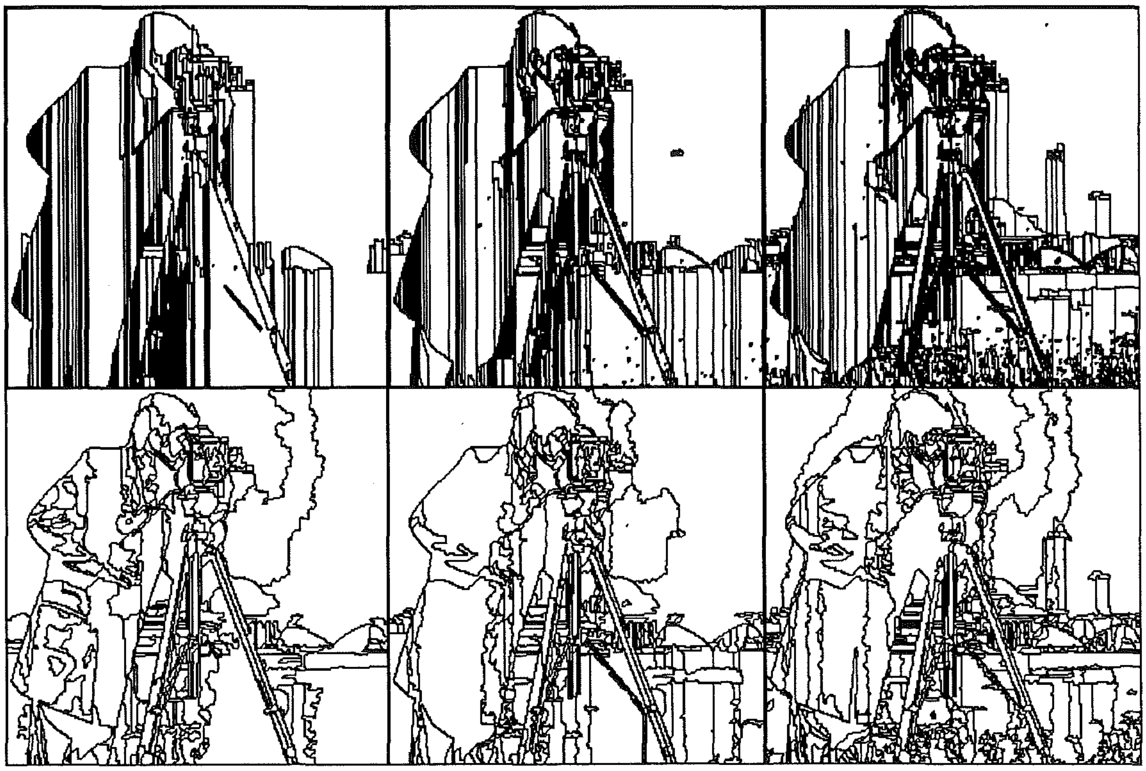


Figure III.19.- Examples of wrong initial segmentations

Column	1st	2nd	3rd
Init. # Reg.	502	1841	4069
Final # Reg.	315	424	631

Table III.3.- Data from segmentations of Figure III.19

Furthermore, zones wrongly oversegmented (that is, with too many regions and wrongly located) are recovered with their correct regions (e. g.: the coat). In this case, the importance of this result is even more noticeable. Initial regions have very smooth contours (almost rectangular), which are associated with very likely realisations of the lower level of the model. Moreover, their gray levels are very uniform, which leads to similar probabilities whichever the contour configuration is. Hence, the importance of performing correctly the initial segmentation of such zones.

On the other hand, it is important to emphasise the problems that a wrong initial segmentation arises. Note that the algorithm has not been able to recover some regions that do not appear in the initial segmentation or it has recovered them only partially (small and large building, respectively, in the two first examples). Since new states in the algorithm are obtained by checking contour points, recovering interior regions that the initial segmentation has totally overviewed is almost impossible. An interior region is a region such that its contour does not touch at any point the contour of another region. Thus, if an interior region has not been detected in the initial segmentation, the algorithm is almost unable to recover it.

The only mechanism for detecting interior regions is by expanding a spurious region that may happen to be close to the interior one. Spurious regions are regions appearing in homogeneous zones without corresponding with any obvious area of the image. Since they are located within homogeneous zones, their boundaries do not conform to any clear shape and their textures are very similar to those of neighbour regions. Therefore, their evolution when applying the segmentation algorithm is rather erratic. This is the case of regions that achieve to detect part of the large building in the two first examples.

The effect of having spurious regions in homogeneous zones within the initial segmentation is very difficult to control. As it can be seen in the first example of Figure III.19, a spurious region going from the camera down to the columns has allowed the algorithm to partially recover the large building, but it has not been able to totally recover it. Moreover, in the third example, a rectangular-like region above the shoulder of the man in the initial segmentation has expanded yielding a wrongly located region and with a very complicated shape.

Even if such spurious regions happen to expand correctly, their expansion slows down the algorithm. Actually, low quality initial segmentations result in a much slower algorithm convergence. In terms of computational load, results presented in Figure III.19 are, in average, four times more expensive than those shown in Figure III.12 or III.15.

A good summary of the main drawbacks arising when using a poor initial segmentation as well as a fair demonstration of the power of the joint use of the image model and the segmentation algorithm is the experiment illustrated in Figure III.20. In this experiment, the image Cars has been initially segmented (first image in Figure III.20 shows the original picture, while the second image is the contour image of its segmentation obtained with the algorithm presented in the previous sections). This result has been used as initial segmentation for the Cameraman image. The algorithm yields the result shown in the third image of Figure III.20 as a contour image and in the fourth one as a mosaic image.

As it can be seen, even if the segmentation uses such a simple algorithm with a white Gaussian random field based model, the main regions of Cameraman have been found. However, initially oversegmented homogeneous zones are finally obtained as a set of patches (for instance, the sky), and initially overgrown regions do not help for recovering details or hidden structures (for instance, the face of the man). However, the segmentation of the grass is performed better in this example than in the former ones. This improvement is owing to the fact that, here, the initial segmentation of this textured zone contains less regions than in the former examples. Since the algorithm lacks of a general view of the image (computations are only carried out in a local manner), when textured zones are initially oversegmented, the algorithm cannot detected the whole zone as a single region.

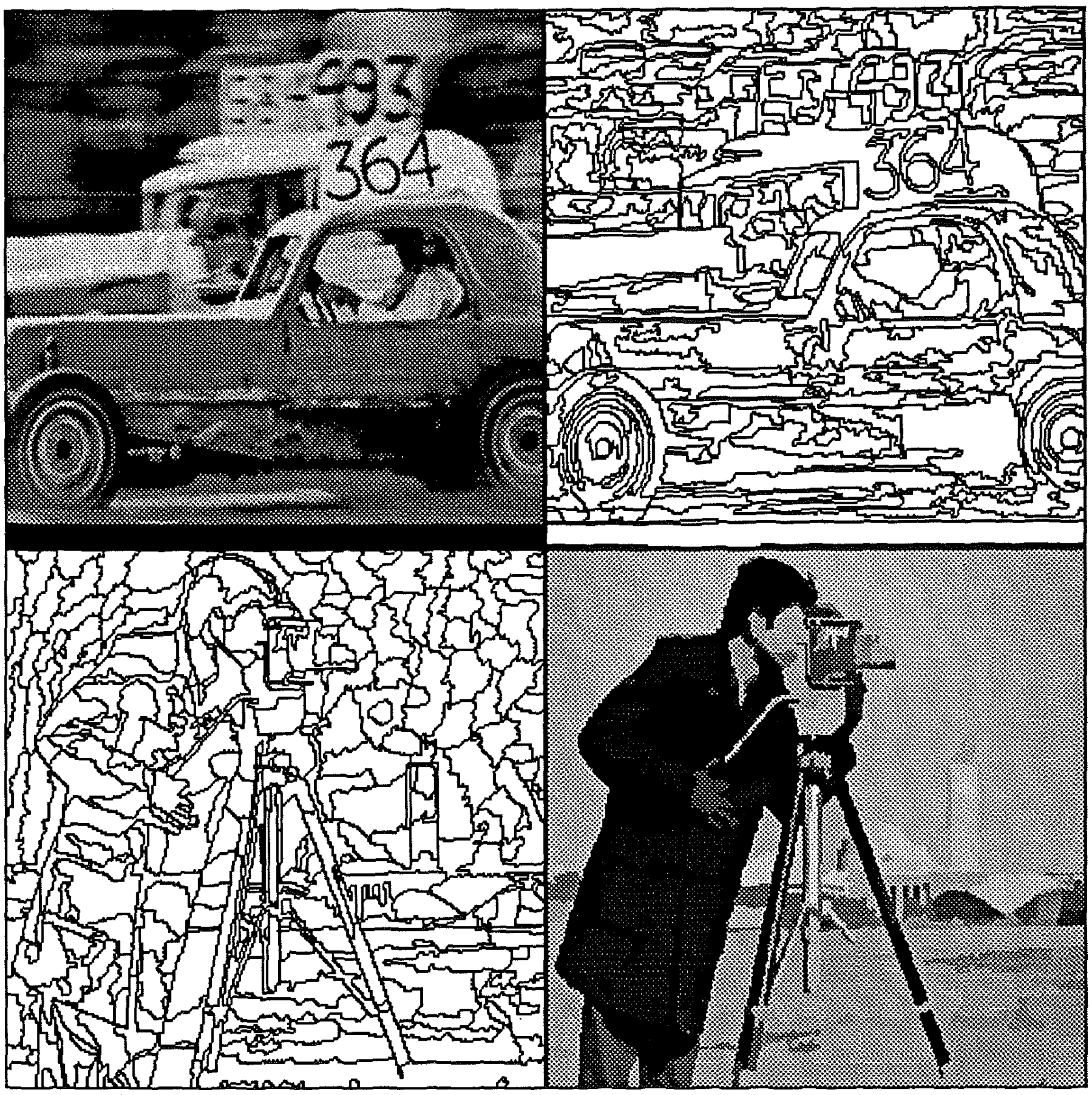


Fig. III.20.- Example of poor initial segmentation

Finally, in order to further highlight thè main characteristics of the algorithm, some examples of its performance on different images are shown in Figure III.21. At each row, the original image and its segmentation (contour image as well as mosaic image) are presented. The algorithm used for obtaining the initial segmentation is the bottom-up technique described in [66]. It has to be emphasised that this algorithm yields initial segmentations with fairly good quality but it is very computationaly demanding. The final segmentation has been performed by using the recursive algorithm. The value of the parameters (T*, V*) has been set to T* = 1 and V* = 0.5.

In the first row, the segmentation is performed on a frame of the so-called Miss America sequence. The quality of the result (292 regions) can be noticed in different zones. For instance, the hair has been segmented apart from the background whereas small regions have been detected (see the eyes or the teeth). Nevertheless, three spurious regions appearing in the initial segmentation have been expanded (left hand-side of the image). These regions prevent to have the whole background segmented as a single region.

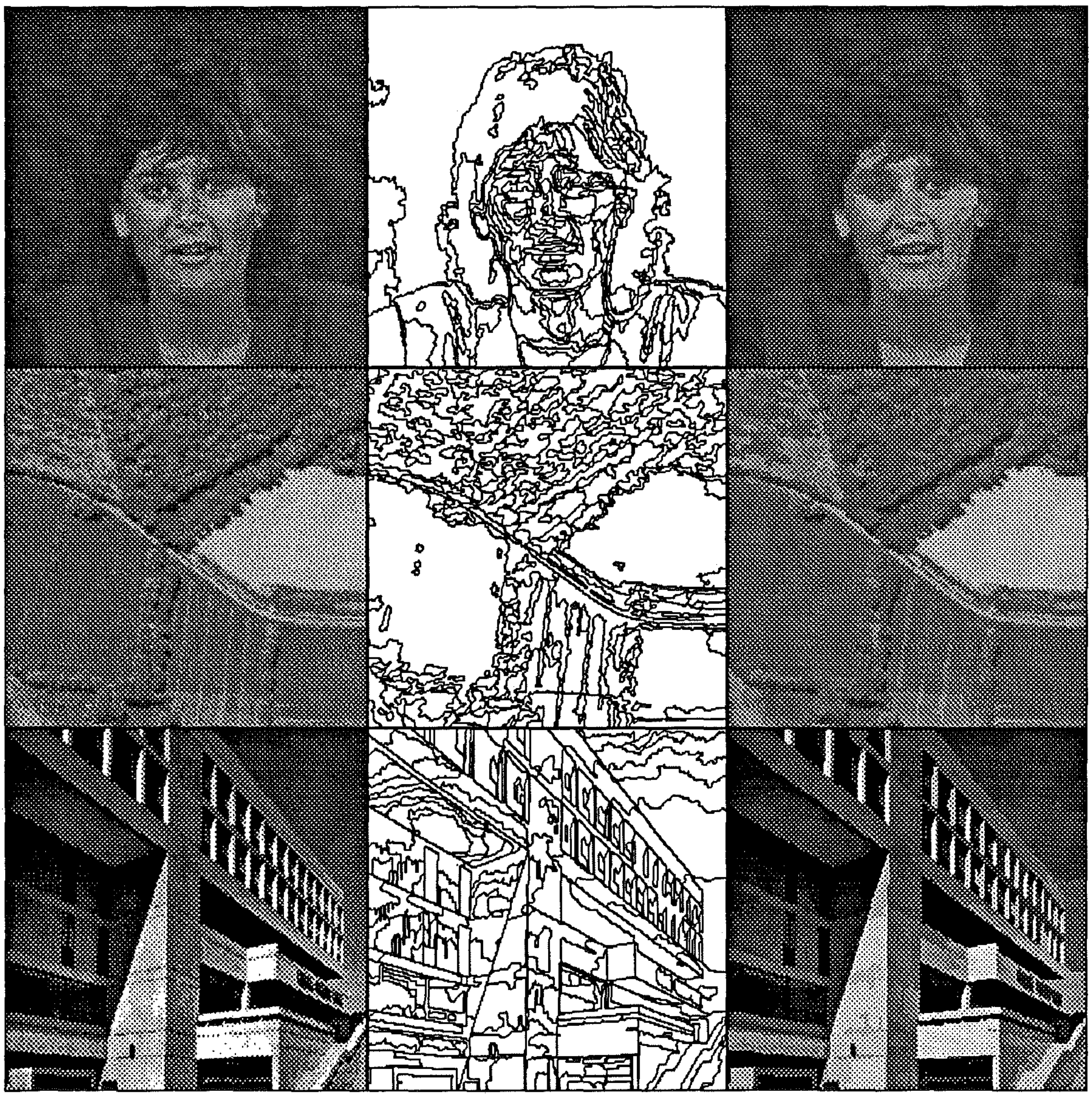


Fig. III.21.- Some examples of segmentations

An example using a textured image, which will be referred as Aerial, is presented in the second row. In this example the effect of using bottom-up techniques for carrying out the initial segmentation is very noticeable. This kind of techniques do not cope correctly with textured areas, and leads to oversegmented results. Moreover, given that the algorithm has a local viewpoint of scenes, it is not able to improve in this aspect the final segmentation. Therefore, although the mosaic image (317 regions) reproduces fairly well the original image, the final segmentation only accounts for gray level variations, withdrawing the texture concept.

Finally, a quite linear-like image, the so-called Building image, has been tested. In this example, the validity of the boundary model for handling both line-like and random contours is illustrated. The mosaic representation of the image (284 regions) shows that in the segmentation, all the important features of the image have been preserved.

III.5.- Summary

The image model used in this work has been presented in this chapter. This model is a compound random field formed by a set of white Gaussian random fields in its upper level and by a Strauss process in its lower level. The reason for choosing white GRFs for the upper level is that this kind of model, when used in a CRF, leads to a good compromise between capability of characterising regions and computational load. On its turn, a Strauss process is used for the lower level, given its good performance when modelling labelled images. Furthermore, Strauss processes can also be utilised for modelling the behaviour of contour images. The choice of a second-order neighbourhood system has been justified, as well as the fact of assigning zero potential to all cliques different from non-homogeneous, two-site cliques.

To perform a segmentation based on this stochastic image model, a deterministic algorithm has been chosen. This algorithm is a special case of the method called Iterated Conditional Modes (ICM). The main drawback of ICM methods is that they do not guarantee reaching global optimum solution. This kind of algorithms depend very much on its initial conditions and on the technique for selecting a new state of the system from a given solution. Here, new states are devised ensuring that they still represent correct partitions of the image. Although this method does not allow to create directly new regions, it can create them indirectly by splitting in two a previously found region.

The procedure for checking if a new state of the system can be accepted (that is, if it increases the joint probability function) has been shown to be simple and fast. This procedure relies on local computations since both the image model and the method for producing new states allow such kind of computations. The complexity of the algorithm has been fairly reduced by using maximum likelihood estimation criteria for estimating the parameters of white GRFs. Regarding the mechanism for selecting pixels to produce new states, three different methods have been presented. The so-called recursive algorithm has been chosen, given that its convergence is guarantied, its computational load in a sequential machine is very small and it yields similar results to the others.

For overcoming the problem of estimating the parameters of the Strauss process, studies of the influence of these parameters on the segmentation results have been carried out. The three parameters characterising the lower level model have been shown to be redundant and reduced to only two (T^*, V^*). The first parameter (T^*) controls the importance of the upper level model in front of the lower level model within the compound random field. That is, when T^* increases, contours are penalised with respect to textures (and vice versa). However, it has been shown that for a large range of values of T^*, resulting segmentations improve the performance of the initial ones. The best results for a large set of images have been obtained for a range of values around $T^* = 1$.

The second parameter (V^*) controls the relation between cliques of type 1 and type 2. It is worth noticing that, for any possible value of parameter V^*, the result of the algorithm improves the quality of the initial segmentation. Nevertheless, depending on its actual value, contours within the image present different characteristics. When $V^* \rightarrow \infty$, the model gives priority to cliques of type 2; that is, noisy contours (contours with fast oscillations) are highly likely. On the other hand, when $V^* \rightarrow 0$, the model does not account for region compactness, which mainly results in not penalising regions with zones of one pixel width. In order to obtain a model conforming as much as possible to natural objects, priority has been given to smooth contours. In the experiments performed on a large set of images, the best results in this sense have been achieved when using $V^* \in [0.25, 1]$.

The dependence of the algorithm with respect to the initial segmentation has also been studied. It has been shown that, given any possible initial segmentation (even random ones), the final segmentation always improves the visual quality of the initial one. However, although the algorithm improves the quality even when selecting a poor initial segmentation, results are seldom of high quality. This effect is mainly due to three reasons. There is no easy control on the algorithm behaviour when it deals with spurious regions which may appear in homogeneous zones: it randomly expands or shrinks them. Moreover, if the initial segmentation has oversegmented homogeneous zones, the algorithm is not able to remove spare regions. The main reason for this drawback is that the algorithm performs only local computations and, therefore, it lacks of a general view of the scene. Finally, interior regions that have not been detected by the initial segmentation cannot be directly recovered by the algorithm. This problem is owing to the fact that the algorithm builds new segmentation by acting in the boundaries of the current one.

Another important drawback is the computational load of the whole method. Using simple techniques for obtaining the initial segmentation usually leads to poor results. In these cases, the algorithm requires a large amount of computations in order to reach a local optimum. On the other hand, techniques providing with good initial segmentation are comparable or even more expensive in terms of computational load than the algorithm itself.