**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 VI

# APPLICATION TO IMAGE CODING

This chapter is devoted to the application of the segmentation technique presented in the previous chapters to the task of image coding. Segmentation based image coding schemes use regions as basic items in the coding task. Regions yield two different kinds of information to be coded: texture and boundary information. Assuming that the segmentation procedure has been correctly performed, textures (that is, region interiors) are associated to homogeneous data sets. This homogeneity translates into redundancy when aiming at coding. Therefore, texture coding can be performed without involving too many parameters. Note that the so-called mosaic images, which have been shown in the previous chapters, are reproductions of the original images using a single value (mean value) for characterising each texture.

On the other hand, boundary information coding requires much more effort, given that region boundaries lack of the high homogeneity of texture information. This difference between both kinds of information leans segmentation based coding schemes towards devoting the largest amount of bits to coding contour images. Usual figures are 7/8 of the total amount of bits dedicated to boundary coding and 1/8 to texture coding. Thus, the most of the effort has to be spent in trying to improve the boundary coding performance. Hence, this chapter mainly addresses the problem of coding contour images achieved by means of a previous segmentation of the scene.

## VI.1.- Problem statement

In order to efficiently code image data, the local redundancy of signals has to be exploited. Classical image coding techniques [82] rely on exploiting the local redundancy either among consecutive points in a given scan (e. g.: Differential Pulse Code Modulation or Deltamodulation) or among neighbour pixels grouped into separated sets forming blocks. This second group of techniques lead to higher performance than the first one. Thus, block based coding schemes will be used for comparing with segmented based coding techniques.

### VI.1.1.- Block based versus segmentation based coding schemes

Block based coding techniques gather the data to be jointly processed. However, these techniques carry out a fixed partition of images. That is, regardless the information contained within the image, all images are segmented following an a priori partition. This partition usually splits the image into several small square blocks, as in a chess board. Each of these blocks (usually containing from 4x4 to 8x8 pixels) is isolately processed in order to code its texture information. Among the different existing procedures for handling the information contained in each block, transforms techniques [82], as Discrete Cosine Transform (DCT) or Karhunen-Loewe (KL) transform, and Vector Quantisation (VQ) [83] are the most extended.

The use of a fixed partition for every image allows applying the same base of functions for all blocks when dealing with transform techniques or using the same vector dimension when coping with VQ approaches. Moreover, block shape and location within the image has not to be coded, as the partition is constant. On the other hand, a fixed partition arises some drawbacks. Since blocks are processed separately, small differences when coding adjacent blocks result in the so-called "block effect" in the decoded image. That is, the chess board structure of the partition clearly shows up in the decoded image, forming spurious contours. Problems related to this effect worsen when the coding scheme is aimed at producing images which are to be seen by human observers. Given that the Human Visual System (HVS) is very sensitive to the contour information, the "block effect" is even more noticeable.

Furthermore, although data have been grouped into several sets in order to exploit local redundancy, blocks have been formed regardless the actual data which they contain. In this way, data in a block may not share the same characteristics; that is, may not be homogeneous. For instance, a block may contain data from two objects in the scene with very different textures. This lack of homogeneity inside blocks leads to a low quality of the decoded images when aiming at high compression ratios (and vice versa). This degradation can be related to transitions between regions (contours) which are wrongly represented in the decoded image. As above commented, contour information is very important for the HVS. Therefore, small errors when representing contours translate into a huge degradation of the visual quality achieved by the coding scheme.

These effects are illustrated in the image of Figure VI.1, where the Cameraman image coded by means of DCT techniques with a compression ratio of 25 is shown. Note the presence of "block effect" in the background as well as the loss of details in the whole image.

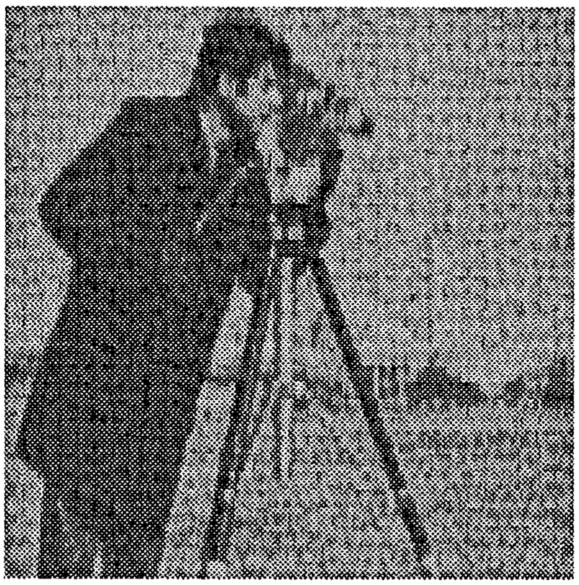

**Fig. VI.1.- Example of image coded by block based coding schemes**

The so-called Second Generation coding techniques [84] try to overcome the above problems by introducing results about the study of the properties of the HVS into the coding scheme. Under this scope, several methods have been proposed improving the previously existing approaches [85, 86, 87]. Object-based coding approaches form a group of the so-called Second Generation coding techniques. These approaches rely on a previous segmentation of the image and a posterior coding of the boundaries and textures of its regions. In this way, each image is coded by using its own partition, which should conform better to the data in the scene.

The advantages of a segmentation based coding scheme results from the fact that data to be jointly processed is guarantied to share the same features. This homogeneity leads to a simpler coding of the region interiors. Moreover, homogeneity prevents the appearing of transitions between different textures in a region. Therefore, the above image low quality, due to the wrong contour representation, is avoided.

In addition, the problems arisen when processing separately blocks in the partition do not appear when dealing with regions. The "block effect" translates, in this case, into a "region effect". That is, if any boundary is highlighted by the fact of processing regions separately, those are boundaries which have been supplied by the segmentation and which are assumed to be the actual boundaries present in the scene. Therefore, the "region effect" would emphasise the actual boundary information contained in the original image.

Note that all these advantages rely on the assumption of having fairly good segmentations of the images to be coded. If this is not the case, segmentation based coding schemes fail to avoid the problems of block based ones. As it has been seen in Chapter V, segmentations obtained by the method presented in the previous chapters largely fulfil these quality requirements.

The main disadvantage of segmentation based coding schemes is the fact that information about the actual region boundaries has to be coded. In the block based case, both the transmitter and the receiver know a priori the partition which is used and, thus, it does not have to be coded. However, in the segmentation based case, partitions vary from image to image and information about the current partition has to be coded. Within the whole coding scheme, this information demands the largest amount of bits.

Summarising, segmentation based coding schemes generate two different kinds of information: texture and boundary information. As regions in the segmentation are assumed to gather homogeneous data, texture information is very redundant. Hence, to code the texture information with an acceptable quality does not demand too many parameters (and, therefore, neither do too many bits). On the other hand, boundary information lacks of the high homogeneity of texture information. Therefore, the largest part of the total amount of bits to be sent in a segmentation based coding scheme is related to boundary information. Typical figures for the amount of bits generated by this two kinds of informations are 1/8 of the total amount devoted to texture coding and 7/8 devoted to boundary coding. Hence, the importance of improving the performance of the boundary information coding.

### VI.1.2.- On coding boundary information

As pointed out in Section VI.1.1, HVS is very sensitive to the boundary information. Decoded images presenting spurious or non-natural boundaries dramatically reduce the visual quality performance of the whole coding system. Therefore, a method for coding boundary information, which introduces as little distortion as possible, should be sought. That is, the boundary coding scheme should either be lossless or keep a strong control on the kind of losses introduced, ensuring that the visual quality does not decrease.

The boundary information produced by the segmentation of an image can be stored in a data structure only containing this information. This data structure is named contour image and the way to create it will be discussed in a latter section. Contour images contain two different kinds of information which can be analysed separately: region shape information and region location information.

Shape information refers to the actual form of each region. As stated above, the coding of this information is mandatory and the final visual quality of the whole coding scheme depends very much on the accuracy of the shape representation. Nevertheless, region boundaries present some redundancy which can be used in order to achieve accurate representations with a fair bit cost. Location information deals with the position of each region within the image. Coding the position by a brute force approach results in using 2 bytes per contour (images of size 256x256 pixels are assumed). However, this information can be coded in a more optimum way by taking advantage of its own redundancy. In addition, there exists some redundancy between shape and location information which can be used to further optimise the location coding.

There exist few works in the literature dealing with contour image coding. Most part of the related works handle the problem of coding or approximating continuous curves in a discrete grid. In this way, chain codes, which can be understood as a lossless contour coding technique, are presented in [88]. Other contour approximating techniques have been used for coding purposes, as the detection of dominant points [89], the use of Fourier descriptors [8] or contour polygonal approximations [26]. However, given their non-exact nature, they lead to coding schemes with losses. Thus, in order to reliably use them in a contour image coding application, measures on the kind of losses that they create and their visual importance should be introduced into the coding scheme.

Chain Codes (CCs) exploit the fact that two consecutive points of a contour in a discrete grid are connected. That is, in order to track a contour, only a few movements are possible. This concept is illustrated in Figure VI.2, where the set of possible movements for a contour defined on a 4-connected and on an 8-connected grid is shown. A contour is coded by chaining the movements needed to completely described its shape.

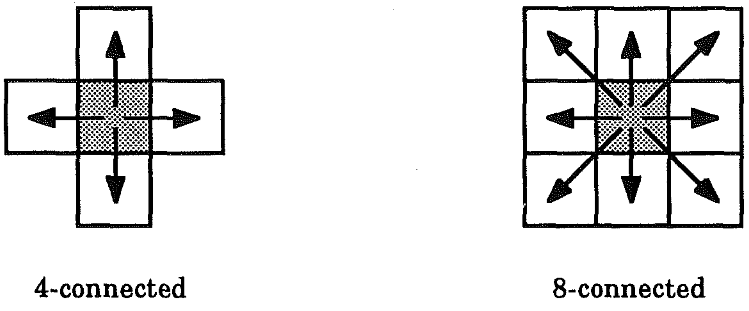

**Fig. VI.2.- Set of possible movements on a 4-connected and on an 8-connected grid**

The amount of different movements necessary to describe a contour can decrease if a Derivative Chain Code (DCC) is used. DCCs code the difference between two consecutive movements. In this way, the number of possible movements in a 4-connected grid is three, since a movement going back to the previous point is not possible. These three movements correspond to a turn to the right, keeping straight ahead and a turn to the left. They are usually denoted by $-\pi/2$, 0, $\pi/2$ or $r$, $s$, $l$, respectively. In a 8-connected grid, DCC results in using only five different movements, since a turn of $\pm 3\pi/4$ is neither allowed. Figure VI.3 illustrates the concept of derivative movements in both kind of grids.

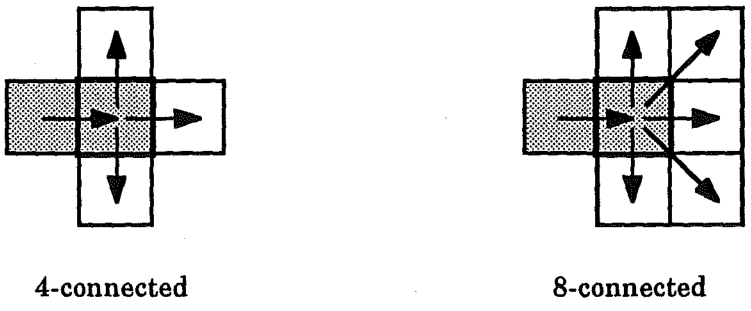

**Fig. VI.3.- Set of movements used in a DCC**

A method for coding contours obtained by a previous segmentation using DCC defined on a 4-connected grid is presented in [90]. Since segmentations are forced to present constrained shapes, all the possible chain configurations are not allowed. That is, regions cannot present elongations of one pixel width and, therefore, double turns to the right or to the left (*...rr...* or *...ll...*) cannot be contained in the contour chain. This feature is further exploited by grouping the symbols representing derivative movements and using Huffman coding techniques [91] to code them. This approach leads to a figure of 1.27 bits per pixel of contour (bppc), which is, up to the author knowledge, the lowest figure in a lossless shape coding scheme referred in the literature. However, even in such a work directly related to the task of contour image coding, the problem of coding the region locations is circumvented. That is, only the task of coding boundary segments is addressed.

A technique for coding region locations within the image is presented in [92] and is also used in [32]. In this technique, the absolute position of the initial point of the first chain of codes is coded by means of its coordinates within the image; that is, 2 bytes in the case of 256x256 pixel images. The remaining initial points are coded in a relative way with respect to their previous initial point. This approach results in a coding scheme whose performance depends on the contour image density of initial points. However, the mean value reported in [92] is of 9 bits per initial point (bpip).

A very different approach for coding contour images is presented in [87]. In this work, morphological skeletons [93] of contour images are used in order to code their information. The skeleton representation of contour images is coded by means of Elias runlength code [94] for coding the skeleton functions and Huffman code for the extinction functions [93]. As reported in [87], this skeleton based approach leads to compression rates lower than those achieved by coding boundaries.

In the present work, the basic shape coding scheme presented in [90] is extended to cover the problem of region location coding. In this way, the contour segment coding scheme which leads to the best performance will be extended to a contour image coding scheme.

## VI.2.- Contour image coding

In order to code the boundaries of a segmented image, and regardless the coding scheme to be used, boundary information has to be represented in a useful manner. A possibility is to transform first the partition into a binary image only containing information about boundaries. In order to obtain a binary contour image representing exactly the initial partition, contour images and label images (partitions) cannot be defined on the same lattice. If contour images are represented on the same lattice as label images, the relationship between contour and label representations is not one-to-one. That is, different label configurations lead to the same contour representation.

This situation is illustrated in Figure VI.4 where two examples are shown. In each example, two different partitions lead to the same contour image, with all representations defined on the same lattice. The contour image in this case has been obtained by assuming 8-connected regions and comparing two by two all the pixels in the partition. When two pixels have different labels, the pixel from the pair which is situated either at the top or at the left side is marked as being a contour pixel. Contour pixels are marked with a solid dot in the contour image of Figure VI.4.

Note that in the first case, the contour image cannot discriminate between vertical and horizontal regions of one pixel width. In the second case, a two-bulb region with both bulbs connected by a single pixel results in the same contour configuration as being two different regions. Therefore, the relationship between label and contour configuration is not one-to-one.

Although contour images related to these examples have been created following a specific procedure, any other procedure using the same lattice for label and contour images leads to analogous results. Hence, the necessity of using another kind of lattice on which contour images can be defined.

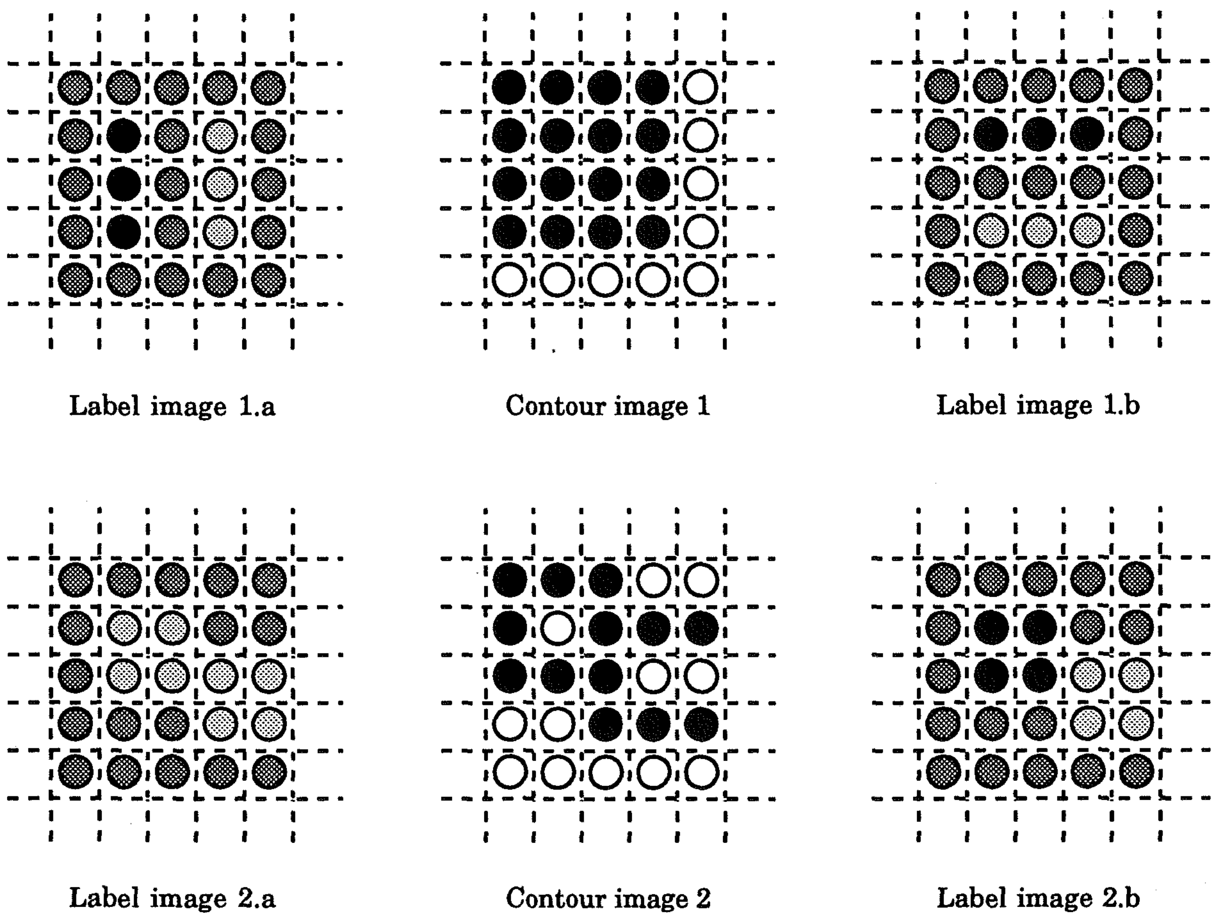

**Fig. VI.4.- Examples using the same lattice for contour and label images**

## VI.2.1.- Contour lattice

Given the above problem, a new lattice for representing contours has to be defined. Figure VI.5 illustrates the definition of a contour lattice solving the above problem, yielding a one-to-one relationship between label and contour representations. In the left part of this figure, circle elements represent pixel sites (elements of the label lattice), while line elements are contour sites. Square elements are not used in this discussion, but help to understand the relation between both lattices. In the contour lattice, a contour element is set to zero if its two closest pixel sites have the same label, and to one if not. Note that this representation is the same used in Section III.1.2 for illustrating the concept of "line process" involved in the definition of the lower level image model.

A second representation of both lattices is shown in the right part of Figure VI.5. This representation is more related to the actual implementation of the lattices, since a square matrix is used. Matrix elements marked with a circle belong to the label lattice whereas sites marked with a line are contour elements. Empty sites represent not used elements of the matrix (squares in the first representation).

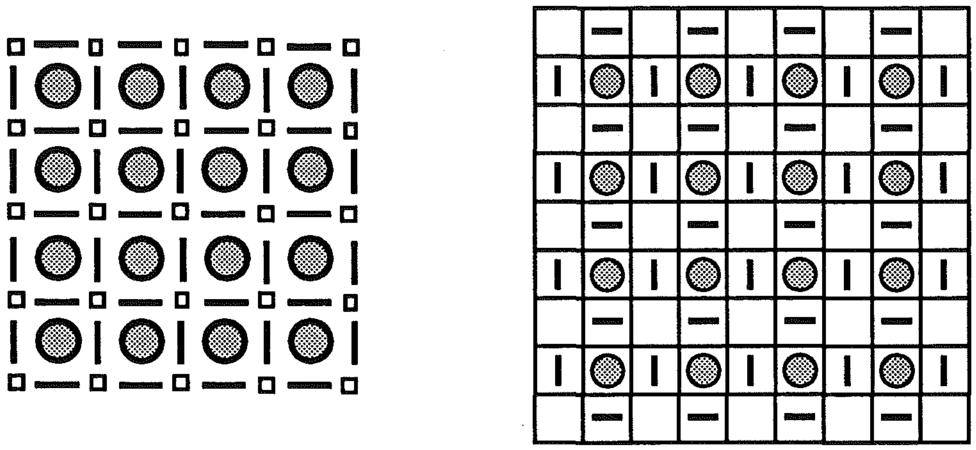

**Fig. VI.5.- Relationship between label and contour lattices**

Note that contour lattices contain more elements than their associated label lattices. Actually, if a label lattice has NxN sites, its contour lattice will have 2xNx(N+1) sites. To use a larger lattice allows contour images to have a one-to-one relationship with labelled images. This relationship is illustrated in Figure VI.6 where the contour images yielded by the four label configuration of Figure VI.4 are shown. Related label configurations lead here to different contour images, making possible recovering the original partitions from them.

It has to be highlighted that one pixel width elongations produce a large amount of contour elements with respect to the number of pixels laying on the boundary of the elongation. That is, if the elongation has a length of N pixels, 2xN+1 contour elements are necessary in order to represent it. Actually, the smoother the region, the closer the number of contour elements to that of pixels laying on its boundary.

Another characteristic of this lattice is that the neighbourhood system to be used is fixed. In pixel lattices, problems arise when defining lattices over a 4- or an 8-connected grid. The contour lattice defined above has a unique neighbourhood system which is 6-connected. Note that there are two different classes of sites in contour lattices: vertical and horizontal sites (see Figure VI.5). Each site is related to two sites of its same class and four of the opposite class. Therefore, the contour lattice is hexagonal.

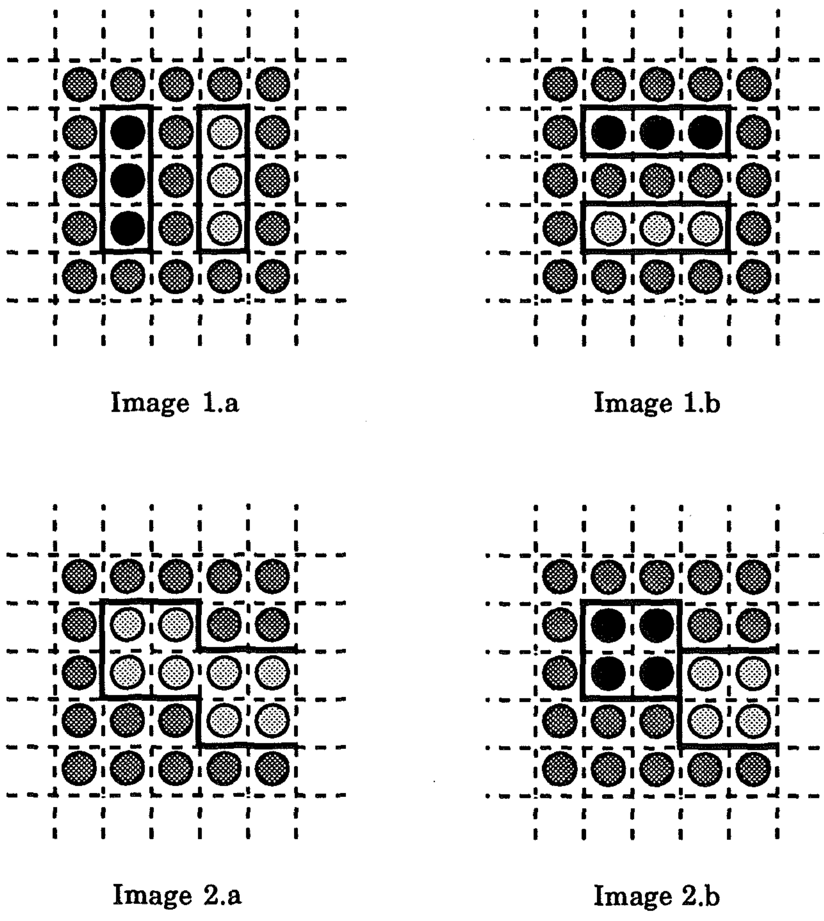

**Fig. VI.6.- Examples of the relationship between contour and label representations**

Chain codes defined on an hexagonal lattice deal with six different movements. Therefore, this kind of lattice worsens the situation with respect to 4-connected grids. However, when using DCC in 6-connected grids, there are only three possible movements. Therefore, the above defined contour images lead to a one-to-one relationship between contour and label images, while requiring the smallest amount of different movements in order to describe contours. The neighbourhood system on which contour images have been defined is shown in Figure VI.7, as well as the set of possible movements used in a CC and a DCC.

In order to use a set of code symbols having a general meaning for any possible situation in the contour lattice, each derivative movement has to be named regarding the class of contour pixels which associates. For instance, in the example of DCC shown in Figure VI.7, the central site is an horizontal one. Therefore, the movement relating it with a vertical site should be denoted by $-\pi/2$ or $\pi/2$ (*r* or *l*) depending on whether the turn is to the right or to the left, respectively. In this way, the derivative movement relating the central site to the other horizontal site is denoted by 0 or *s*.

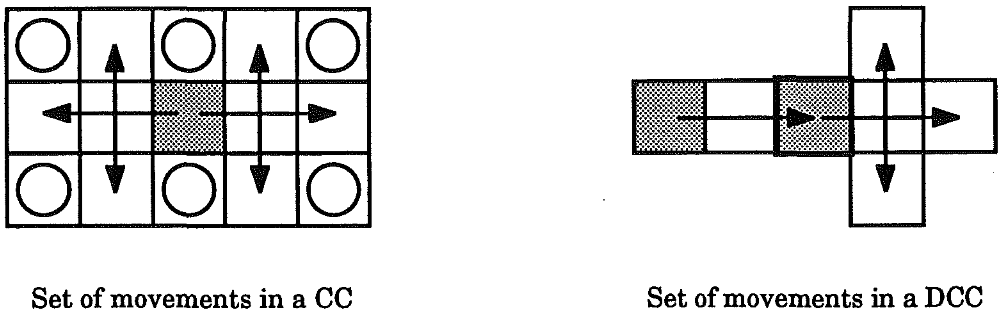

**Fig. VI.7.- Possible movements in a 6-connected grid for a CC and a DCC**

An example of representation of a shape defined on an hexagonal grid in terms of its DCC is shown in Figure VI.8. Note that, in spite of having two different classes of contour sites and of dealing with a 6-connected grid, only three different symbols (derivative movements) are necessary for representing any kind of shape.

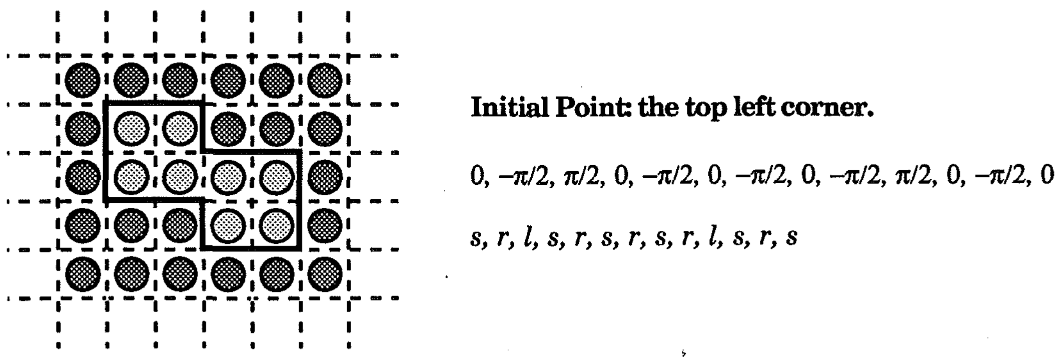
**Initial Point: the top left corner.**
0, −π/2, π/2, 0, −π/2, 0, −π/2, 0, −π/2, π/2, 0, −π/2, 0
*s, r, l, s, r, s, r, s, r, l, s, r, s*

**Fig. VI.8.- Example of DCC representation of a shape in an hexagonal grid**

## VI.2.2.- Shape coding

As it has been pointed out in the previous section, to utilise an hexagonal lattice for defining contour images does not prevent the use of only three movements for coding the shape of their regions. Therefore, the shape coding technique presented in [90] can still be applied. This technique relies on a DCC representation of shapes and a posterior grouping of codes in order to exploit the different probabilities of each group by means of Huffman coding. That is, small codewords are assigned to very likely groups, and vice versa.

However, this technique improves its coding performance when applied on constrained shapes. If shapes are constrained, some configurations may not appear when forming the groups of movements. Therefore, the remain configurations increase their likelihood and the Huffman coding achieves higher compression ratios. In [90], constraints forbid contours to have two consecutive turns to the same direction; that is, chains containing ...*rr*... or ...*ll*... are not allowed. These constrains translate into forbidding the existence of regions with one pixel width elongations. Note that such elongations are rather unnatural in real objects and can therefore be removed from the segmentation. Moreover, as stated in Section VI.2.1, one pixel width elongations are the region configurations which more contour elements yield with respect to the number of elements that they represent in the partition. Therefore, they are expensive to code while not containing very much information.

In the original paper [90], the segmentation algorithm was directly constrained to yield such contours. Nevertheless, any kind of labelled images can be filtered in order to fulfil the above constraint. This filter can be performed in several ways; for instance, by means of a morphological open_close ($\gamma\varphi$) or close_open ($\varphi\gamma$) with a flat, square structuring element of size 2 (see Chapter V). The application of these constraints results in changing the form of contours and, therefore, the shape coding scheme is no longer lossless. However, losses can be introduced if they do not distort very much the quality of the decoded image (specially from the human observer viewpoint).

The visual effect of applying this kind of filter to mosaic images can be appreciated in Figure VI.9, where the mosaic representation of the Cameraman and Building images are shown before and after filtering. In both cases the used filter is an open_close. The difference between both images are almost unnoticeable. However, some small details have been removed from the camera in the Cameraman image, as well as some of the horizontal dark lines on the columns of the Building image.

Once one pixel width elongations have been removed from the partition, shapes can be described in terms of DCC. In order to further exploit the constraints introduced in the shapes by the filtering, Huffman coding techniques are applied. As above outlined, in this kind of techniques, a codeword is assigned to a movement or a set of movements depending on their probability of occurrence. That is, the more likely the set of movements, the smaller the length of the codeword. It should be emphasised that codewords grouping large amount of movements result in more efficient coding schemes.

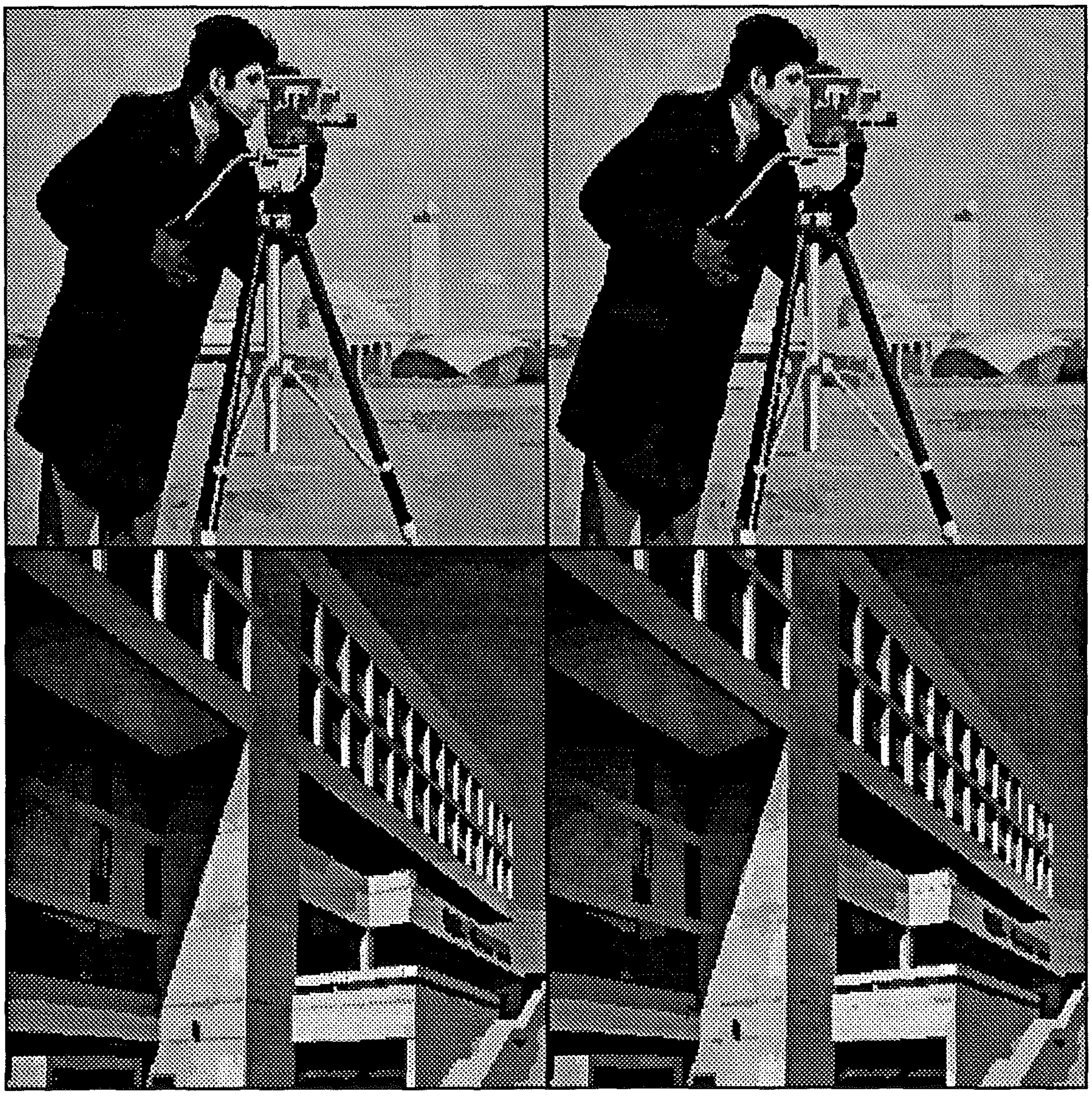

**Fig. VI.9.- Effect of the removal of one pixel width elongations**

The coding of the shapes in a contour image can be performed by fixing a global codebook for all the images or by using, for each one, its optimum codebook. When using for each image its optimum codebook, this codebook has also to be transmitted so that the receiver can decoded the data. This procedure leads to an unfeasible amount of overhead information. Therefore, a study on the size of the groups of movements leading to codebooks with the most similar likelihood distributions has been performed.

In this study, shapes obtained from a large set of contour images have been used for training the codebook. At each experience, a contour image has been removed from the training set and the codebook has been built with the shapes of the remaining contour images. This global codebook has been compared with the optimum codebook of the removed contour image. It has been observed that, using codewords grouping three movements, the codebook likelihood distributions are almost constant for any image and, therefore, a global codebook can be fixed. If codewords grouping four or more movements are used, likelihood distributions change so much that a fixed codebook cannot be implemented.

### VI.2.3.- From shape coding to contour image coding

Up to this point, regions shapes have been coded isolately, regardless the fact that they are contained in the same image. In order to code a whole contour image, each shape can be extracted from the contour image and be coded isolately. The extraction of a single region shape from a contour image can be performed by, given an initial point, tracking the contour elements in a clockwise direction when possible. Each time that, from the actual contour element, more than one contour element can be reached, priority should be given to $-\pi/2$ movements (a turn to the right) and 0 movements (to keep going straight ahead). Of course, an analogous extraction can be carried out by tracking the contour elements in counterclockwise direction and giving priority to $\pi/2$ movements (turning left). Following this method, a whole contour image can be coded by the concatenation of packages of location and shape information:

| | | | | | |
|---|---|---|---|---|---|
| ... | location #i | shape #i | location #i+1 | shape #i+1 | ... |

All location packages can have the same size (2 bytes in an absolute coding or 9 bites in a relative coding). On its turn, the end of a shape package is marked by the fact that its last movement leads to the initial point marked by the previous location package. Figure VI.10 illustrates this coding procedure. In this figure initial points are marked with a solid square.

However, this coding scheme exploits neither the redundancy of shapes nor that of locations. When shapes are located within a contour image, adjacent regions share parts of their boundaries. Therefore, a large amount of bits can be saved by coding only once these common boundary segments. Note that, in this case, the coding scheme deals with boundaries segments rather than with shapes. In order to achieve such a coding scheme, the procedure for finishing a chain should vary from that in the former coding scheme. That is, a boundary segment is assumed to be finished when, tracking it, a contour element already coded is reached.

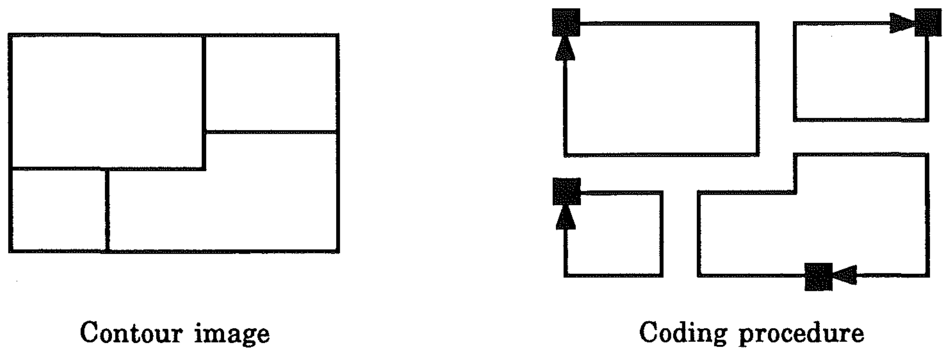

**Fig. VI.10.- Example of contour image coding by shape extraction**

Although this coding scheme reduces the amount of contour elements to be coded, it can produce more initial points than the actually necessary. Attention must be paid when choosing the initial points for each boundary segment in order not to produce unnecessary boundary segments. Two different codings of the contour image of Figure VI.10 are shown in Figure VI.11 to illustrated this effect.

Note that, in order to obtain a coding scheme involving the minimum possible number of initial points, those have to be selected in the extremes of the boundary segments. If initial points are selected in any other position, boundary segments are split and unnecessary initial points must therefore be introduced.

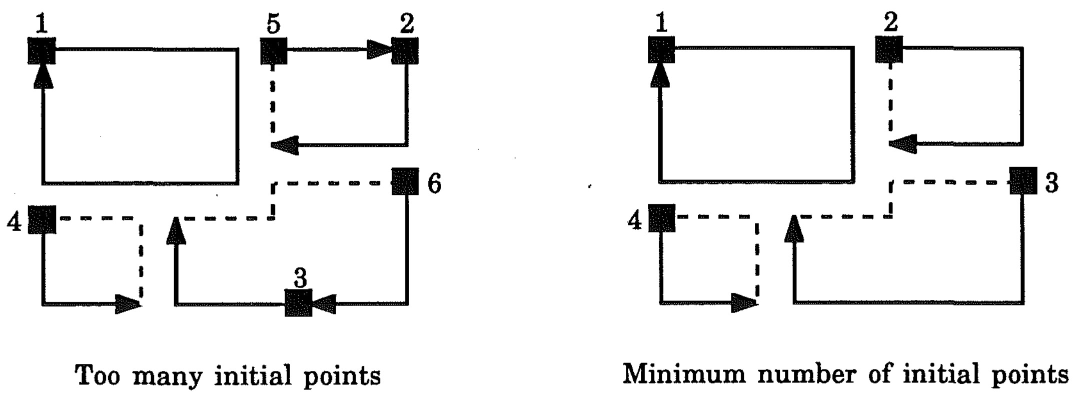

**Fig. VI.11.- Example of the importance of choosing correctly the initial points.**

### VI.2.4.- Location coding

The coding of both shape and location information has been assumed, up to this point, to be performed separately. However, a more efficient coding scheme can be achieved by exploiting the relation between both kinds of information. That is, a coding performance improvement can be obtained by introducing the information regarding region locations in the chain code itself. This improvement can be implemented by using the concept of triple point. A triple point in a contour lattice is a site having three active neighbour sites (actual contour elements).

Triple points are found in the intersection of two different contours and, therefore, they can be used in order to locate the initial point of a new boundary segment. The way to locate a new contour is by marking in the chain code that a triple point has been found. That is, each time that, when tracking a contour, two movements from the current contour site are possible, this site is marked as being a triple point by introducing a new symbol in the chain. This procedure is illustrated in Figure VI.12, where triple points are depicted by a solid dot.

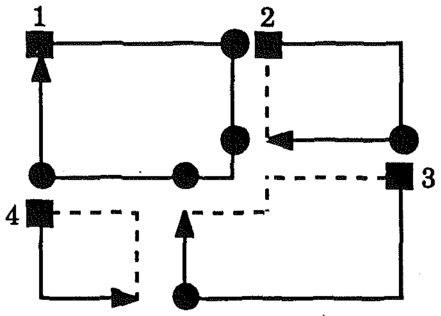

**Fig. VI.12.- Procedure for marking triple points**

Some important aspects regarding the use of triple points have to be emphasised. Triple points do not directly mark initial points, but they are related to them by means of a single derivative movement. Moreover, triple points composed a set of possible initial points. Note that in Figure VI.12 six different triple points have been detected while only four initial points are necessary (see Figure VI.11). That is, several triple points may be related to only one boundary segment. Therefore, a posterior selection has to be performed. This selection cannot be performed in terms of choosing those triple points yielding initial points in the extremes of boundary segments. By construction, all triple points are related to boundary segment extremes (see

Figure VI.12). However, this characteristic ensures that a minimum set of initial points can be obtained relying on the triple points.

A procedure for detecting the unnecessary triple points is to remove them whereas computing the chain code. That is, triple points are marked in the chain and their related possible initial points are stored in a first-in-first-out (FIFO) stack. When a boundary segment has been totally coded (that is, the chain code has reached an already coded contour element), a new initial point is popped from the stack and its related boundary segment is coded. If a point which has been previously marked as possible initial point is reached when coding the new boundary segment, this point is removed from the stack and so is the mark of its related triple point from the chain code. In this way, only the set of necessary triple points is coded. Figure VI.13 illustrates this procedure for selecting triple points. Rejected triple points have been marked with a white dot.

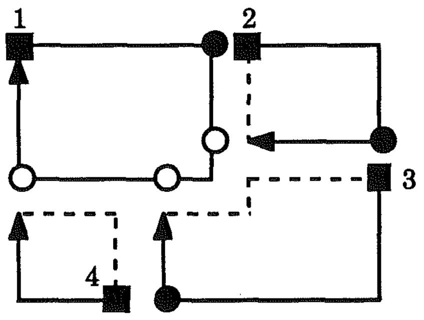

**Fig. VI.13.- Selection of triple points**

It has to be noticed that not all the initial points of boundary segments can be coded relying on triple points. If there is a region in the partition (or a cluster of regions) not having any contact point with other regions, its contour representation does not produce any triple point. That is, clusters of interior regions cannot be located by means of triple points. However, the number of interior regions in segmented images is very small (less than 8% of the total number of regions). Given that their density within the image is very low, location of clusters of interior regions have been coded using their absolute coordinates. Thus, the above coding scheme yields packages of information with the following structure:

| | | | | | |
|---|---|---|---|---|---|
| ... | loc. #i | shape cluster #i | loc. #i+1 | shape cluster #i+1 | ... |

where all location packages have the same size.

Packages containing the coding of the shapes of the region clusters are formed by the concatenation of the DCCs of all the boundary segments in the cluster. The end of a boundary segment is obtained when a previously coded contour element is reached. When decoding a shape cluster package, triple points have to be stored in a stack, in an analogous way as in the coding procedure. The end of a shape cluster package is marked by the fact that, when reaching a previously coded contour element, there are no items in the triple point stack.

In order to code the presence of a triple point, a new symbol, different from the anterior symbols referred to movements, has to be introduced in the chain. In the case of dealing with constrained contours, a new symbol is not strictly necessary since a triple point can be denoted by one of the forbidden chain configurations (*...rr...* or *...ll...*). Actually, a configuration marking a double turn to the right (*...rr...*) cannot be used, given that two different chain configurations lead to the same representation:

$$...dt... \Rightarrow ...ddd... \qquad \text{and} \qquad ...td... \Rightarrow ...ddd...$$

where the symbol $t$ denotes triple point. This is not the case when using a double turn to the left configuration (*...ll...*) since chains containing a turn to the left after a triple point are not possible (*...tl...*). Note that after a triple point mark, at least two different possible movements can be performed. Since shapes are tracked following a clockwise direction, a turn to the left will never be chosen.

The introduction of *...ll...* configurations within the chain code changes the likelihood distributions of the codewords in the Huffman coding. However, the amount of triple points to be coded is so small with respect to the total number of movements that the variation in the distribution are almost unnoticeable.

## VI.3.- Coding results

A large set of images has been coded by using the region based coding scheme presented above. For pedagogic reasons, only four different images are used in this section in order to illustrate the performance of the coding scheme. The chosen images are Cameraman, Miss America, Cars and Building. Nevertheless, mean values of all the coding parameters obtained with the whole set of images are also given.

Images containing large textured areas (e. g.: Table-Tennis or Synthetic) have been removed from this set given that they present very different characteristics from the rest. That is, they lead to higher compression ratios than lower textured images (values around 100 are normal in these cases) since a few regions are detected. On the other hand, the representation of textured areas by means of their mean value results in a lower quality reproduction than that achieved for lower textured images

Four different figures are given in order to characterise the performance of the coding scheme: the number of bits per pixel of contour necessary for coding the shape information (bppcs), the total number of bits per pixel of contour (bppc) necessary for coding the whole contour image –that is, when coding both shape and location information–, the average number of bits introduced for coding each initial point (bpip) and the compression ratio achieved when representing each region by its mean value (cr). Values of these figures for the above commented four images as well as their average values are shown in Table VI.1.

| Image | bppcs | bppc | bpip | cr |
|---|---|---|---|---|
| Cameraman | 1.20 | 1.38 | 7.1 | 25.4 |
| Miss America | 1.21 | 1.35 | 7.6 | 40.3 |
| Cars | 1.12 | 1.35 | 7.2 | 22.8 |
| Building | 1.17 | 1.39 | 7.8 | 26.6 |
| Average | 1.20 | 1.36 | 7.3 | 27.5 |

**Table VI.1.- Results of the proposed coding scheme**

The bpip figure has been computed as the difference between bppc and bppcs divided by the number of regions. Note that in this coding scheme, in order to code a whole contour image, three different kinds of information have to be added to the shape information: marks of triple points, the first movement of each chain relating the triple point with the actual initial point and the direct coding of the initial points of the interior clusters of regions.

It should be highlighted that this coding scheme leads to compression ratios ranging from 20 to 40 when using a byte to characterise the interior of regions (regions characterised by their mean value). Moreover, the amount of bits necessary to code the location information has been reduced with respect to absolute or relative coding. In absolute coding 16 bits per region are necessary while in relative coding, the average value is 9 bits. When using triple points, this figure reduces to 7.3 bits per region (in average). The quality of the decoded images can be observed in Figure VI.14 for the Cameraman and the Miss America images and in Figure VI.15 for the Cars and Building images.

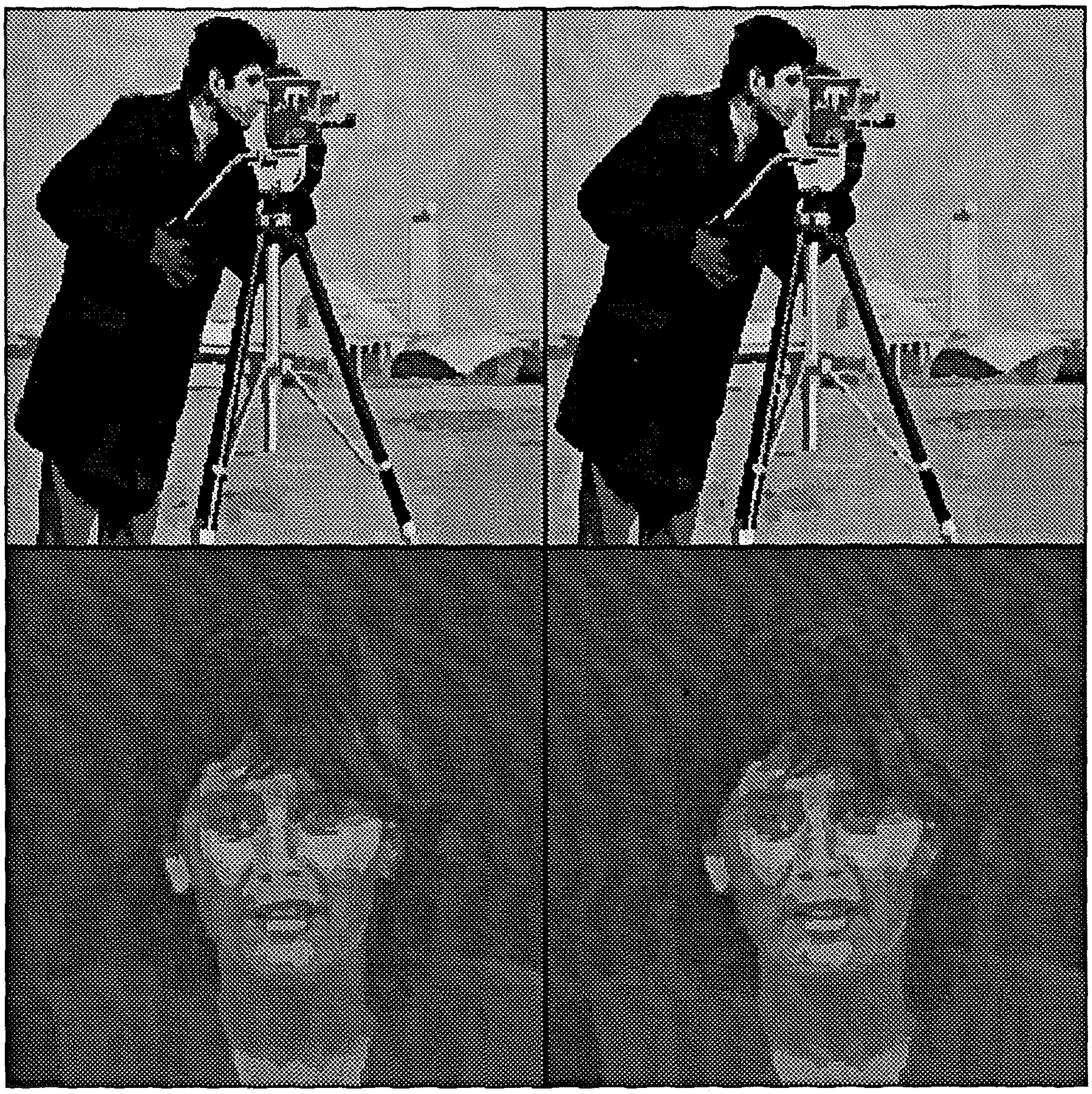

**Fig. VI.14.- Results of the proposed region based coding scheme (I)**

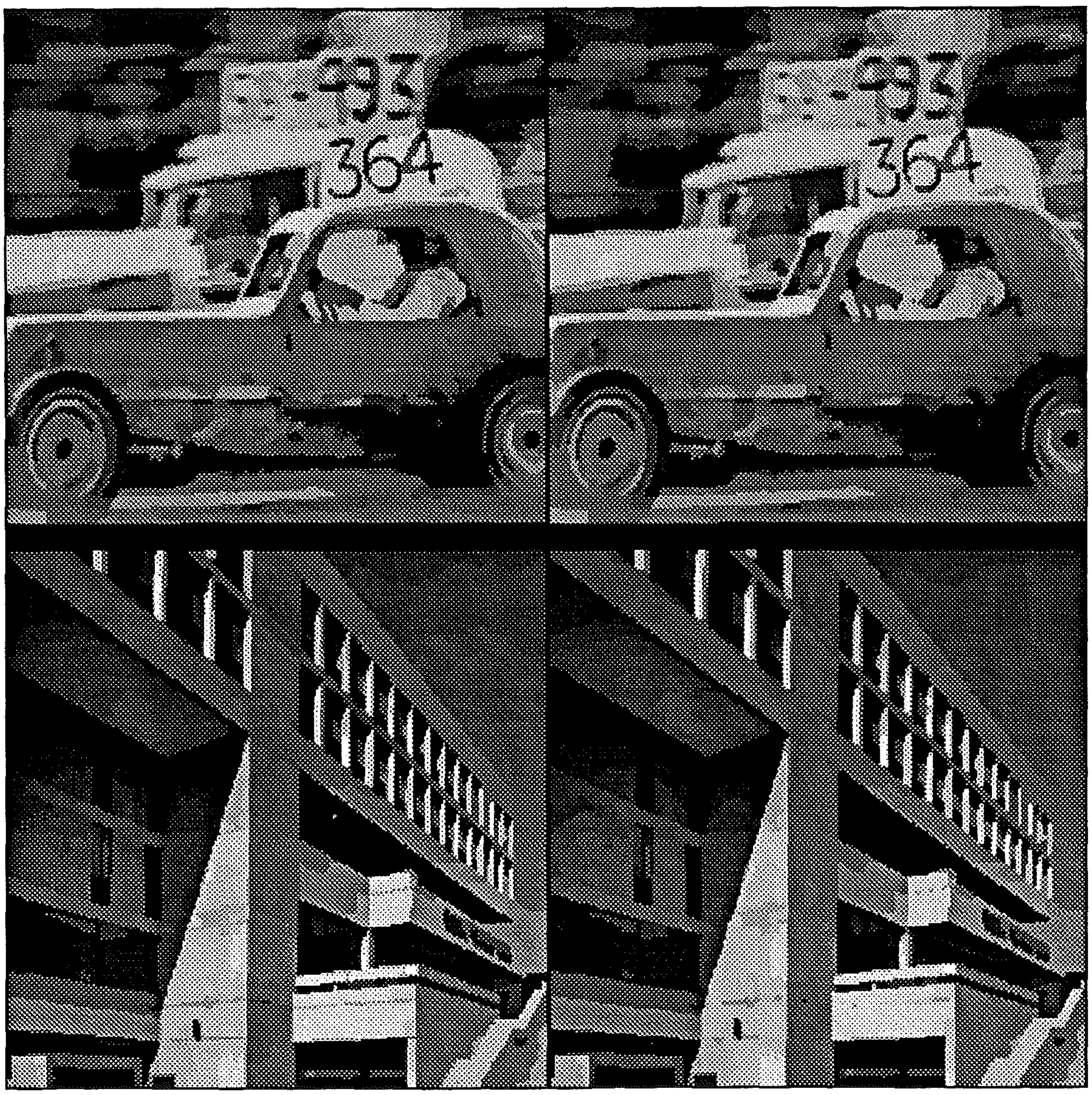

**Fig. VI.15.- Results of the proposed region based coding scheme (II)**

Finally, in order to compare the performance of this coding scheme with a block based coding scheme, Figure VI.16 shows the results of coding the Cameraman and Miss America images by a transform technique (DCT) and by the proposed method. In both examples, images have been coded with same compression ratio for both approaches. Note that the quality of segmentation based results clearly improve those of block based coding schemes. However, it has to be emphasised that classical transform techniques yield better quality results when dealing with low compression ratios than the proposed method.

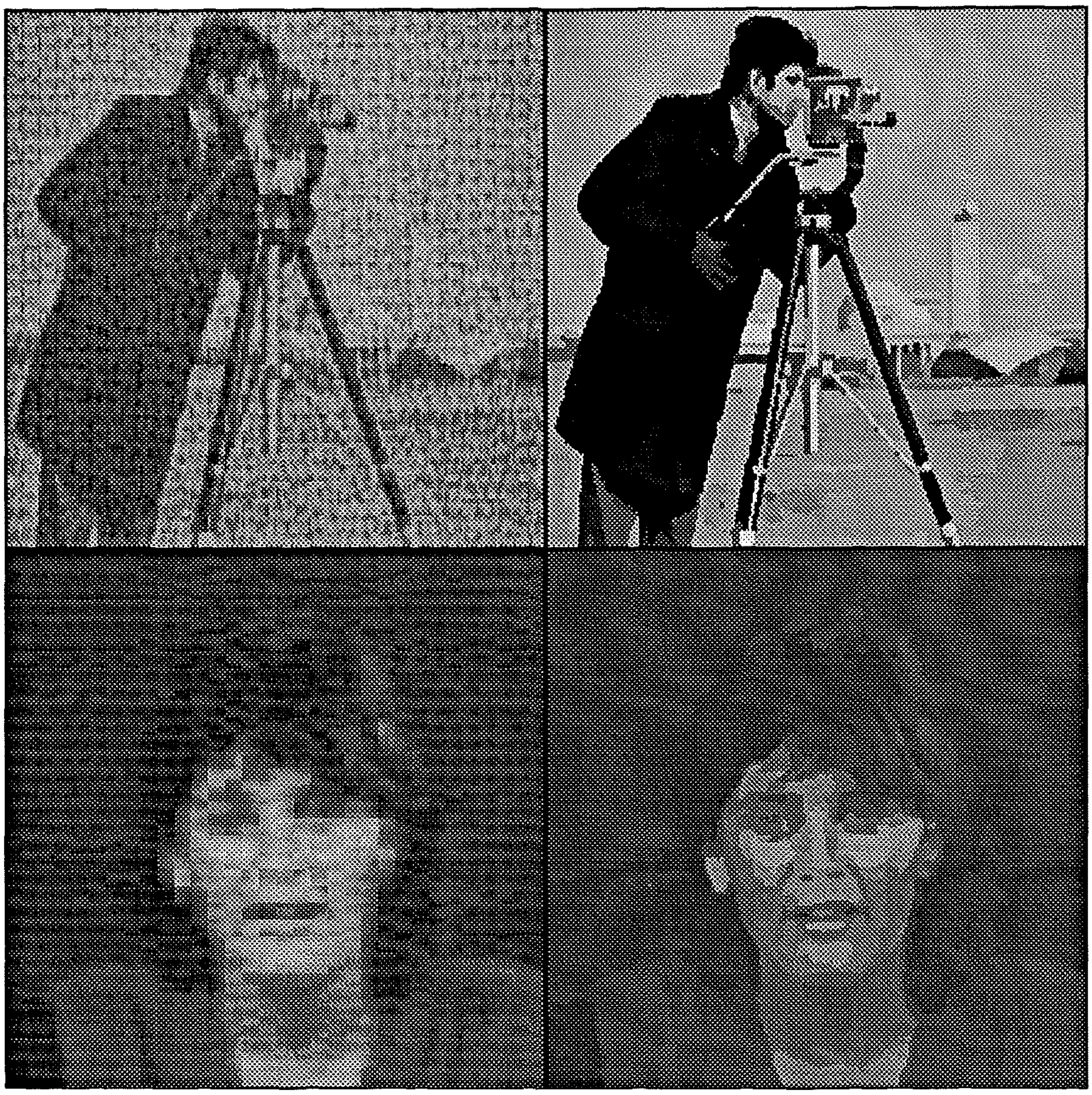

**Fig. VI.16.- Comparison between block based and region based coding schemes**

## VI.4.- Summary

The application of the segmentation technique presented in the previous chapters to the task of image coding has been discussed in this chapter. Segmentation based and block based image coding schemes have been compared. It has been seen that, segmentation based coding schemes can improve the visual quality of block based ones: they can remove the so-called "block effect" and improve the contour representation quality in the decoded image. On the other hand, these coding schemes generate two different kinds of information: texture and boundary information. Texture information, as being very redundant, can be correctly characterised by few parameters.

However, since boundary information is not so redundant, it requires a large amount of bits in order to be coded. Therefore, the greatest effort has to be devoted to improve the boundary information coding procedure. In this way, the chapter has mainly dealt with the problem of coding the boundary information yielded by an image segmentation.

Boundary information in a segmentation can be split into two parts: information about the shape of regions and about their location within the image. Distortion introduced when coding shape information should be strongly controlled, given the importance that the human visual system gives to this kind of information. Therefore, a lossless coding scheme is to be used or, in case of introducing losses, their influence in the visual quality of the decoded image should be very small. The best known scheme for coding shape information relies on the use of derivative chain code techniques over constrained boundaries defined on a 4-connected grid. By this approach, figures of 1.27 bppc are achieved. However, this method does not take into account the coding of the region location. Regarding the region location coding, some works have used a scheme based on coding the initial point of a boundary relatively to the last coded initial point. In this way, figures of 9 bpip in average are obtained.

The above shape coding scheme has been extended to cover the problem of region location coding. In order to gather the boundary information in a simple data structure, contour images have been introduced. Contour images are defined on an hexagonal lattice for allowing one-to-one relationship between boundary representations and partitions. Although being 6-connected, shapes defined on this lattice can be described only using three different movements when using derivative chain code. Therefore, the above shape coding technique can still be used.

In the original work, this technique deals with constrained shapes in order to improve its performance. These constraints prevent the appearing of one pixel width elongations on regions. Given that the removal of such kind of elongations does not degrade the visual quality of the decoded images, these constraints are introduced by means of a filter. Constrained shapes are described in terms of derivative chain codes. Introducing the above constraints results in having some forbidden chain configurations. In order to further exploit the chain code redundancy, Huffman coding techniques are applied to the chain codes. Codewords have been chosen to group three derivative movements since this number leads to the most efficient Huffman coding scheme.

In order to code the location information in an efficient way, this information has been introduced in the chain code itself. This task has been carried out by using the concept of triple point. A triple point of the contour lattice is a site that has three active neighbour sites (actual contour elements). Therefore, triple points mark the presence of a new shape and can be used to locate its starting point. The way to locate a new shape is by introducing a mark in the chain code denoting that a triple point has been found when tracking a boundary segment. However, after finishing to track a boundary segment, remaining triple point marks have to be checked. Some marks may have to be removed from the chain since they may not give information about new boundary segments (more than one triple points may be related to only one boundary segment). The mark of triple point can be coded by using one of the forbidden chain configurations yielded by the shape constraints.

It has been highlighted that not all the starting points of boundary segments can be coded in this way. Interior regions do not produce triple points. Therefore, interior regions still require 2 bytes in order to be located. However, the number of interior regions that appear in segmented images is very small (less than 8% of the total number of regions). Thus, to use the concept of triple point still leads to a great reduction in number of bits.

Results yielded by the above presented region based coding scheme have been shown. Average values of the different parameters that describe the coding scheme have been given, as well as specific values for a small set of images. Good quality images are obtained with 1.36 bits per pixel of contour and a compression ratio of 27.5 (average values). Finally, results achieved by block based (DCT) and region based coding schemes have been compared. The quality of the results yielded by the proposed coding scheme is clearly superior to that of DCT, when aiming at high compression ratios.