Available online at www.sciencedirect.com
SCIENCE DIRECT®
Omega 35 (2007) 173–183
omega
The International Journal of Management Science
www.elsevier.com/locate/omega

A graph coloring approach for image segmentation

D. Gómez*, J. Montero, J. Yáñez, C. Poidomani

Complutense University, Escuela de Estadistica, 28040 Madrid, Spain

Received 23 July 2004; accepted 12 May 2005
Available online 7 July 2005

Abstract

In this paper we develop a segmentation scheme for digital images based upon an iterative binary coloring technique that takes into account changing behavior of adjacent pixels. The output is a hierarchical structure of images which allows a better understanding of complex images. In particular, we propose two algorithms that should be considered as image preprocessing techniques.

Keywords: Segmentation techniques; Graph theory; Decision support systems

1. Introduction

Classical segmentation of digital images is quite often assimilated to a search for *objects* whenever clear borders exist, i.e., when sudden changes are observed when we move from one pixel to a next one. But natural entities may not show either a clear border or any standard shape, but a smooth gradation between different regions (see, e.g., [1–5]).

As shown in [6,7], some classification and optimization problems can be modelled as graph coloring problems. In this paper we present a coloring algorithm for segmentation, susceptible of being considered as a first stage for a posterior classification analysis, specially in the presence of transition zones between classes. A main argument, at least in the land cover problems we have some experience with (see [8–10]), is that our regions have a core of connected pixels, and surrounding pixels may show some relatively smooth gradation towards a different region. Hence, the behavior of surrounding pixels should have a strong influence at every stage.

Our approach will be based upon the successive application of a basic binary procedure which will produce a nested structure of regions, with a precision to be fixed by decision makers according to their abilities and final objectives (a small family of classes explaining the image, for example). The output of this iterative binary coloring (IBC) technique is a crisp partition of all image pixels, each pixel belonging to exactly one *color class*.

A change in color may only suggest a different region. Each one of these regions may suggest a class meanwhile it is considered *homogeneous*, but a color itself should not be associated to a class: a detailed comparison between different regions should be developed in order to get some useful classification (determining possible classes and their transition zones). Such a classification problem will not be addressed in this paper, but only a hierarchical segmentation procedure. Improvements leading to more sophisticated segmentation algorithms can be also tried depending on the final objective of our image analysis (for example, introducing main arguments about fuzzy classification systems, as considered in [11,12]). Decision makers should take into account such an information in a later supervised analysis, where additional information may exist.

* Corresponding author.
E-mail address: dagomez@estad.ucm.es (D. Gómez).

0305-0483/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.omega.2005.05.003

Hence, in this paper we propose two new methods for image segmentation, based on a grid graph representation where the vertices are the pixels and the edges have weights corresponding to distances or dissimilarities between adjacent pixels. The first method aims to produce a very precise coloring of the graph, taking into account all variations in distances. Such an algorithm is based upon an IBC procedure which can be applied if the pixels network verifies some consistence conditions. However, such an algorithm has no polynomial complexity, so a second relaxed method is proposed.

The paper is organized as follows: in Section 2 we introduce the pixels network as a valued planar graph summarizing all information of the image, and in Section 3 a crude coloring algorithm is detailed. A relaxed version of the previous algorithm is then introduced in Section 4, providing some results. Application to some particular images is given in Section 5, followed by a final comments section.

2. The image as a pixels network

Our main objective in this paper is to produce a coloring algorithm, so a summarizing picture of the original image can be offered to decision makers in order to search for possible regions.

The image is therefore understood as a bidimensional structure of pixels (the *minimal information unit*), each one being connected to its natural neighborhood on the image. Of course, each pixel will be characterized by a fixed number of measurable attributes. These attributes can be, for example, the values of the three bands of the visible spectrum (red, green and blue), the whole family of spectrum band intensities, or any other family of physical measures. The information about our image is in this way summarized as

$$I = \{(x^1_{i,j}, \ldots, x^b_{i,j})/(i, j) \in P\},$$

in case each pixel (i, j) is being characterized by b numerical measures ($b = 3$ when dealing with the visible spectrum only), where

$$P = \{(i, j)/i = 1, \ldots, r;\ \ j = 1, \ldots, s\}$$

is the set of pixels, described by means of their cartesian coordinates (i, j), meaning that we are dealing with an image of $r \times s$ size.

Two pixels $p = (i, j) \in P$ and $p' = (i', j') \in P$ are *adjacent* when they share one coordinate, being the other one contiguous. In this way we define the edges set E of the planar graph $G(I) = (P, E)$ as

$$E = \left\{((i, j), (i', j')) \in P \times P / \vee \begin{bmatrix} i = i', & |j - j'| = 1 \\ j = j', & |i - i'| = 1 \end{bmatrix}\right\}.$$

Given such an image I, a standard segmentation problem pursues a partition, each one being a subset of pixels:

$$P = \bigcup_{h=1}^{H} R_h / R_h \cap R_{h'} = \emptyset \quad \forall h \neq h'.$$

In order to visualize segmentation, each element of such a partition will be associated here to a color class, so borders are associated to a change in color. This partition will be based upon a distance

$$d : E \longrightarrow [0, \infty)$$

on the measured properties of pixels. Such a distance can be, for example, the Euclidean distance in $\mathbb{R}^b$:

$$d(e) = \sqrt{\sum_{k=1}^{b} (x^k_{ij} - x^k_{i'j'})^2} \quad \forall e = \{(i, j), (i', j')\} \in E.$$

Of course, an ad hoc distance should be defined for each specific problem. Obviously, the segmentation process will be strongly dependent on such a distance, to be carefully chosen, taking into account all features of the image under consideration together with some restrictions that may be associated to the final objective of segmentation.

Definition 2.1. Given the image I and a distance d between the measured properties of pixels, the *pixels network* is defined as

$$N(I) = (G(I), \{d(e)/e \in E\}).$$

Hence, taking into account the particular topology of $G(I)$, our pixels network can be also characterized by the set P plus two $r \times s$ matrices, D^1 and D^2, where

$$D^1_{i,j} = d((i, j), (i + 1, j))$$
$$\forall (i, j) \in \{1, \ldots, r - 1\} \times \{1, \ldots, s\}$$

and

$$D^2_{i,j} = d((i, j), (i, j + 1))$$
$$\forall (i, j) \in \{1, \ldots, r\} \times \{1, \ldots, s - 1\}.$$

Since our coloring procedure will be based upon this alternative representation, from now on we shall identify our pixels network $N(I)$ simply by (r, s, D^1, D^2).

Example 2.1. Let (r, s, D^1, D^2) be a pixels network with $r = 3$, $s = 4$ and

$$D^1 = \begin{pmatrix} 1.5 & 2.0 & 1.9 & 3.6 \\ 1.7 & 1.8 & 0.9 & 1.8 \end{pmatrix}, \quad D^2 = \begin{pmatrix} 3.1 & 1.9 & 3.5 \\ 1.6 & 4.0 & 1.8 \\ 2.1 & 0.7 & 0.8 \end{pmatrix}.$$

Such a pixels network is depicted in Fig. 1 (see Appendix A).

The key coloring algorithm proposed in the next section will take advantage of the above representation in terms of the pixels network $N(I)$, which shows intensity variation between adjacent pixels.

3. A crude coloring algorithm

A c-coloring (see, e.g., [13]) of a graph $G=(V, E)$ is a mapping

$$C : V \longrightarrow \{0, \ldots, c-1\}$$

such that $C(v) \neq C(v')$ whenever $\{v, v'\} \in E$. Any c-coloring induces a partition of the nodes set V:

$$V = \bigcup_{k=0}^{c-1} \{v \in V / C(v) = k\}.$$

Our objective here is to translate into a valued context the classical procedure in order to obtain a partition of pixels through a c-coloring C of the pixels network $N(I)$, in such a way that a color $k \in \{0, \ldots, c-1\}$ is associated to a pixel $p=(i, j) \in P$ whenever $C(p)=k$.

In particular, we propose to successively apply a basic binary coloring process, leading to a hierarchical coloring of the image. A binary coloring of a graph $G=(V, E)$ is a 2-coloring, given by a mapping

$$\text{col} : V \longrightarrow \{0, 1\}.$$

The first binary coloring analyzes the pixels set P, assigning to each pixel p either the value "0" or the value "1". A second binary coloring can be then applied, separately, to both the subgraph generated by those pixels previously colored as "0" (obtaining color classes "00" and "01"), and the subgraph generated by those pixels previously colored as "1" (obtaining color classes "10" and "11"). Repeating this process t times, a c-coloring C will be defined on $N(I)$, where $c=2^t-1$; for instance, if some pixel $p \in P$ has been colored three times as "1", "0" and "1", then, taking into account that 5 is the decimal representation of the binary number 101, the color of pixel p is $C(p)=5$.

3.1. The basic binary coloring procedure

The basic binary coloring we propose will assign values "0" and "1" to each pair of adjacent pixels depending on the distance between their measured descriptions, when compared to a prescribed threshold α. The choice of this parameter α will be detailed in the foreword.

Let

$$\text{col} : P \longrightarrow \{0, 1\}$$

be a binary coloring of G. The first binary coloring can be then obtained assigning an arbitrary color ("0" or "1") to an arbitrary pixel, and fixing the order in which pixels will be colored. One possibility is to start coloring the top-left pixel $\text{col}(1, 1)=0$, and the other pixels can be colored from left to right and from up to down, using the following horizontal and vertical rules, respectively:

$$\text{col}(i, j+1) = \begin{cases} \text{col}(i, j) & \text{if } d^2_{i,j} < \alpha, \\ 1 - \text{col}(i, j) & \text{if } d^2_{i,j} \geqslant \alpha \end{cases}$$

for all $(i, j) \in \{1, \ldots, r\} \times \{1, \ldots, s-1\}$.

$$\text{col}(i+1, j) = \begin{cases} \text{col}(i, j) & \text{if } d^1_{i,j} < \alpha, \\ 1 - \text{col}(i, j) & \text{if } d^1_{i,j} \geqslant \alpha \end{cases}$$

for all $(i, j) \in \{1, \ldots, r-1\} \times \{1, \ldots, s\}$.

Given a colored pixel (i, j), the adjacent pixels $(i, j+1)$ and $(i+1, j)$ can be then colored in a similar way. But notice that since pixel $(i+1, j+1)$ can be colored either from pixel $(i+1, j)$ or from pixel $(i, j+1)$, both coloring processes may not lead to the same color (we can call it an *inconsistent* coloring). This means of course that our binary coloring procedure is also dependent on the particular ordering we have chosen for coloring. For instance, in the Example 2.1, if $\alpha=3.0$ and $\text{col}(1, 1)=0$, then $\text{col}(1, 2)=1$ and $\text{col}(2, 1)=0$; however, $\text{col}(2, 2)=1$ if pixel $(2, 2)$ is colored from $(1, 2)$ but $\text{col}(2, 2)=0$ if it is colored from $(2, 1)$. These *inconsistent* situations need to be detected.

Inconsistent situations are therefore present when a pixel can be colored by two different ways, i.e. when there is a cycle in the graph $G(I)=(P, E)$. These cycles will be called *inconsistent cycles*.

If the graph is acyclic, we can obtain a consistent coloring by choosing, randomly, any initial pixel from every connected component, and assigning to each one of these pixels either color 0 or color 1, arbitrarily. In this case, once a value α has been fixed, coloring of each adjacent pixel is unique. This coloring process is based on the previously defined horizontal and vertical rules.

Otherwise, if the connected graph is non-acyclic, this is the case for the original pixels graph $G(I)=(P, E)$, but coloring is unique for any spanning tree $H=(P, T)$. Taking into account that every cycle of a graph can be generated by the set $\{e=\{p, p'\} \in E-T\}$, we can introduce the following definition.

Definition 3.1. Let $H(P')=(P', T')$ be a spanning tree of a subgraph $G(P')$ of the pixels network (r, s, D^1, D^2). Let $\text{col} : P \longrightarrow \{0, 1\}$ be a binary coloring obtained after applying to $H(P')$ the above horizontal and vertical rules at a given value α.

If there exists some edge $e \in E-T'$ of one of the following types:

- An horizontal edge $e=\{(i, j), (i, j+1)\}$ verifying $d^2_{i,j} \geqslant \alpha$ if $\text{col}(i, j)=\text{col}(i, j+1)$ or $d^2_{i,j} < \alpha$ if $\text{col}(i, j) \neq \text{col}(i, j+1)$.

- A vertical edge $e = \{(i, j), (i + 1, j)\}$ verifying $d^1_{i,j} \geqslant \alpha$ if $\mathrm{col}(i, j) = \mathrm{col}(i + 1, j)$ or $d^1_{i,j} < \alpha$ if $\mathrm{col}(i, j) \neq \mathrm{col}(i + 1, j)$.

Then, the cycle defined by $T \cup \{e\}$ is denoted as *inconsistent* at level α.

Let us consider for example the network given in Example 2.1, and let $H = (P, T)$ be the spanning tree depicted in Fig. 2, with the binary coloring obtained at level $\alpha = 3.1$, and starting with $\mathrm{col}(1, 1) = 0$. In this case, the vertical edge $\{(1, 2); (2, 2)\} \in E - T$ defines an inconsistent cycle at this level, because $d^1_{1,2} = 2.0 < 3.1 = \alpha$ and $\mathrm{col}(1, 2) = 1 \neq 0 = \mathrm{col}(2, 2)$. However, there are no inconsistent cycles at level $\alpha = 1.0$ (see Fig. 3).

Given a spanning tree with no inconsistent cycles for a given level α, then this property can be verified for any spanning tree.

Definition 3.2. A pixels network (r, s, D^1, D^2) is *consistent* at level α if it does not contain inconsistent cycles at this level.

For example, it can be checked that the pixels network given in Example 2.1 is inconsistent at level $\alpha = 3.1$, but it is consistent at level $\alpha = 1.0$.

Notice that we may find two extreme consistent cases:

- $\overline{\alpha} = \max_{(i,j)\in P} \{d^1_{i,j}; d^2_{i,j}\}$: if we fix a threshold $\alpha > \overline{\alpha}$, then the whole picture is considered as a unique color class ($\mathrm{col}(i, j) = \mathrm{col}(1, 1), \forall (i, j) \in P$).
- $\underline{\alpha} = \min_{(i,j)\in P} \{d^1_{i,j}; d^2_{i,j}\}$: in case $\alpha \leqslant \underline{\alpha}$, the picture looks like a chess board, being all adjacent pixels alternatively colored as "0" and "1", so:
  - $\mathrm{col}(i, j) = \mathrm{col}(1, 1)$ if $(i + j)$ is an even number.
  - $\mathrm{col}(i, j) = 1 - \mathrm{col}(1, 1)$ otherwise.

Indeed, determining an appropriate intermediate α level is not a trivial task (in practice we should be trying a sequence of levels in order to capture change sizes between adjacent pixels). But only the interval $[\underline{\alpha}, \overline{\alpha}]$ should be considered.

Definition 3.3. Given a pixels network (r, s, D^1, D^2), its *consistency level* α^* is the maximum value $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ for which the network is consistent.

Existence of such a consistency level α^* is always assured, at least meanwhile our image contains a finite number of pixels (the network will be consistent at least for $\underline{\alpha}$). If some inconsistent cycle is detected for certain level α, a decreasing procedure can be introduced in order to find a lower level α' assuring consistency. Such a procedure will be initialized with $\alpha = \overline{\alpha}$.

The procedure bincol(P', α^*, col) shows such a decreasing scheme computing the binary coloring col of the subgraph $G(P')$ generated by a subset of pixels $P' \subset P$. In general, this subgraph is not connected and the binary coloring process is applied to every connected components.

Initially, when the set P' is the overall set of pixels P, the procedure also computes the consistency level α^* of the connected pixels network.

The procedure bincol is depicted in Fig. 6 of Appendix B.

Applying the procedure bincol(P, α^*, col) to Example 2.1, we obtain $\alpha^* = 1.8$. Final result is shown in Fig. 4.

Since for the first binary coloring of G, the computed value of parameter α by the procedure bincol is α^*, this value assures the consistency of this binary coloring process and, consequently, any pixel $p \in P$ will be assigned into color class "0" or into color class "1".

In this way is defined the first segmentation of pixels trough the binary coloring procedure bincol.

3.2. The crude coloring algorithm

The segmentation process induced by the previous binary coloring can be repeated so that a more refined segmentation can be obtained.

For instance, color classes "00" and "01" will be obtained applying the binary coloring process to the subset $P' = \{p \in P/\mathrm{col}(p) = 0\}$. Analogously, color classes "10" and "11" will be obtained applying the binary coloring process to the subset $P' = \{p \in P/\mathrm{col}(p) = 1\}$. In this way, four color classes are being obtained: "00", "01", "10" and "11", which are identified by the mapping

$$C : P \longrightarrow \{0, 1, 2, 3\},$$

where $C(p)$ is the integer number associated to the binary number of the color class of p.

The binary coloring procedure can be successively applied to each family of pixels belonging to the same color class, meanwhile there are adjacent pixels being different.

However, and in order to avoid the exponential growth of the binary coloring processes, this process bincol will be successively applied only t times, being t previously fixed, obtaining in this way 2^t color classes.

The crude coloring procedure in this way defined, denoted as **crucol**(r, s, D^1, D^2, it_M, C) computes the coloring

$$C :\longrightarrow \{0, 1, \ldots, 2^{it_M} - 1\}$$

when the binary coloring process is applied it_M times to the pixels network (r, s, D^1, D^2).

The diagram of procedure crucol is depicted in Fig. 7 of Appendix B.

However, the procedure **crucol** does not guarantee that two pixels equally colored are neither connected or similar. So, the following definition is introduced.

Definition 3.4. Given an image I and the previous coloring C, a connected component of each color class induced by C will be said a *region*.

In this way, two pixels belonging to the same region are equally colored, but the converse is not true (i.e., the region partition is a subpartition of the color class partition).

The procedure crucol, when applied to Example 2.1, needs four iterations, in such a way that there are at most $2^4 = 16$ color classes. The final coloring C is depicted in Fig. 5 (see Appendix A).

It is obvious from Example 2.1 that some color classes can be void (the number of color classes is 16 meanwhile the number of pixels is 12). This is explained by the decreasing scheme of parameter α.

Indeed, the above crucol algorithm has no polynomial complexity. In the worst case, the number of iterations reaches the value $r \times s - 1$.

In the limit case, when $t \to \infty$, the coloring

$$C : P \longrightarrow \{0, 1, \ldots, 2^t - 1\}$$

thus obtained can be understood as a continuous $[0, 1]$-coloring.

Anyway, the above crude coloring algorithm can be inefficient when handling big size images. An appropriate decreasing scheme of parameter α, allowing an acceptable ratio of inconsistent pixel squares, will be the core of the relaxed coloring algorithm we introduce in the next section.

4. A relaxed coloring algorithm

Since we should be expecting the existence of quite a number of inconsistent cycles when dealing with medium or big size images, it may be the case that our decreasing search for α^* reaches the value $\underline{\alpha}$, so the obtained segmentation is the trivial one, which looks like a chess board. Consequently, pixels cannot be properly colored. In order to avoid such a problem, in this section we propose to relax the constraint of consistency in a new relaxed binary coloring procedure, allowing some few inconsistent cycles (in terms of some percentage, for example). But since evaluating all inconsistencies is a computational complex problem, in this paper we propose to approximate the percentage of inconsistencies by considering only the minimal cycles (in the sense of fewest number of edges), that is, cycles generated by four adjacent pixels. In this way we shall bound the percentage of *inconsistencies squares*.

Definition 4.1. Given a pixel set P, a *square* is a subset of four pixels

$$\mathrm{sq}(i, j) \equiv \{(i, j); (i + 1, j); (i, j + 1); (i + 1, j + 1)\}$$

being $i \in \{1, \ldots, r - 1\}$ and $j \in \{1, \ldots, s - 1\}$.

Let PS be the set of all squares defined in a pixels network. The total number of squares of a given $r \times s$ image is $|PS| = (r - 1)(s - 1)$.

Definition 4.2. Given a pixels network (r, s, D^1, D^2), a square $\mathrm{sq}(i, j) \in PS$ is said *inconsistent* at level α if the cycle defined by its pixels is inconsistent at this level; otherwise, the pixel square is said *consistent* at level α.

For instance, in Example 2.1, the square $\mathrm{sq}(1, 1)$ is inconsistent at level $\alpha = 3.1$, but it is consistent at level $\alpha = 3.2$.

At each iteration, let *ratio* be defined as the ratio of inconsistent squares of a binary coloring process, i.e., the number of inconsistent squares divided by the total number of squares subject to inconsistency. Large values of *ratio* are associated to a low number of color classes (the value α does not need to be decreased), but the greater this value *ratio*, the greater the number of pixels with a *wrong* color. Hence, we can look for some compromise between these two arguments. Such a compromise can be attained for small inconsistency ratios, lets say 0.01. Each inconsistent square should then be *isolated* so that its inconsistency does not induce inconsistency of adjacent squares.

The iterative binary coloring process is modified for the relaxed coloring algorithm and it is not based on the spanning tree of the connected components of $G(P')$. The relaxed coloring algorithm is based on the pixel square set PS. These squares are arranged (left to right and up to down) and then analyzed.

Indeed, following the horizontal rule, the coloring process of the first row, from left to right, does not produce inconsistent edges, neither first vertical coloring following the vertical rule. Any inconsistent square $\mathrm{sq}(i, j)$, if any, will appear when the coloring of the pixel $(i + 1, j + 1)$ from the left (pixel $(i + 1, j)$) will be different to the obtained color from the superior pixel $(i, j + 1)$. Let ninc be the total number of these inconsistent squares.

Given two horizontal adjacent pixels $(i + 1, j)$ and $(i + 1, j + 1)$ included in some $G(P')$, for some $P' \subset P$, each one being respectively colored from (i, j) and $(i, j + 1)$, we have that

$$\mathrm{ninc} = \mathrm{ninc} + 1 \quad \text{if} \quad \vee \begin{cases} \mathrm{col}(i + 1, j) \\ \quad = \mathrm{col}(i + 1, j + 1) \wedge d^2_{i+1,j} \geqslant \alpha, \\ \mathrm{col}(i + 1, j) \\ \quad \neq \mathrm{col}(i + 1, j + 1) \wedge d^2_{i+1,j} < \alpha. \end{cases}$$

The inconsistent squares must be appropriately managed in the sense that their pixels must be isolated in order to avoid contamination of adjacent squares; these pixels can be arbitrarily colored. Subsequent iterations will smooth the effects of such arbitrary coloring.

In order to avoid the exponential growth of computations, the number of binary partitions must be again bounded, as in the crude coloring algorithm. Let us denote such a bound by m.

However, and it is different from the crude coloring algorithm, once the value of parameter m has been fixed, different values of α must be selected: let $\{\alpha_{it}/it = 0, 1, \ldots, it_M\}$ be the family of these selected values verifying $\overline{\alpha} > \alpha_0 > \alpha_1 > \cdots > \alpha_m > \underline{\alpha}$. The decreasing scheme of the relaxed algorithm uses this sequence of values, being $\alpha = \alpha_0$ the initial value. If for some value α the proportion of inconsistent squares is greater than a prescribed threshold *ratio* the parameter α is updated to the next lower value.

It is clear that this process is strongly dependent on the α value selection (α_i) in each iteration. Of course, this selection is not a trivial task. If all selected values are close to $\overline{\alpha}$, we will identify only very different regions. If we consider all values close to $\underline{\alpha}$, we will be sensitive to small variations and a non-informative segmentation will be given with too many regions. In order to discriminate in terms of similar size sets of pixels at each step, a possibility is to choose those α values taking into account quartiles within the distance distribution. In this way we can impose certain equilibrium in coloration, of course subject to the final purpose of segmentation.

The procedure rxcol(r, s, D^1, D^2, C, ratio, $\{\alpha_{it}, it = 0, m\}$) is depicted in Fig. 8 of Appendix B.

5. Case studies

The above algorithm has been applied in first place to a standard *RGB* image, taken from the *MATLAB* package (a water colored tree, already considered by Amo et al. [9], see http://www.mat.ucm.es/∼fuzzycs). When the above sequence of binary partitions is developed, the image is divided into regions, and a common number is associated to all pixels within the same region. In order to get a better visualization of the obtained partition in our example, we have associated to each region a RGB color, obtained as the mean of the original color of pixels in the respective region. This mean color intends to represent a homogeneous region. Pixels in each region of the first segmentation (two colors) are recolored in the second segmentation (two new colors each), producing in this way a nested tree sequence of images. It can be checked that in this case main features seem to be captured with $it_M = 2$. A visualization can be found in http://www.mat.ucm.es/∼fuzzycs.

Of course alternative visualization procedures can be tried.

The same algorithm has been also applied to an orthoimage of Sevilla province (south Spain), taken on August 18, 1987, by the LANDSAT 5 satellite (*Worldwide Reference System* (*WRS*) image $202 - 34 - 4$). This image was taken with the Thematic Mapper sensor, which has a spatial resolution of 30 m and shows the same image selected in [10]. It can be checked, again, that segmentation obtained by means of the above relaxed algorithm with $it_M = 2$ seems to capture main features. See http://www.mat.ucm.es/∼fuzzycs.

Notice that in both cases we needed few iterations. Same comment applied to other well-known test-images we have tried with the above relaxed algorithm: WOMAN, HOUSE and HAND (see http://www.mat.ucm.es/∼fuzzycs for details). Of course, a more complete set of experiences is needed in order to develop a trustable criteria about the recommended number of iterations (depending on many context factors like granularity and smoothness, for example), but all the above experiences suggest no practical relevance if we introduce a bound on the number of iterations in the above relaxed algorithm. Notice that basic procedures (see *rxbincol*,*vecalpha*,*rxcol* in http://www.mat.ucm.es/∼fuzzycs) are polynomial whenever the number of coloring procedures has been bounded (e.g., $it \leqslant 6$).

6. Concluding remarks

The coloring algorithm we propose in this paper provides an aid technique in order to find homogeneous regions in any image. The nested binary tree sequence of partitions allows decision maker a consistent way of deciding ad hoc an appropriate precision level. In fact, one of the advantages of our approach is that possible regions can be presented in the form of a nested binary tree. The sequential application of a basic binary coloring allows a first approach to a picture in terms of regions, taking very much into consideration behavior of adjacent pixels.

We have to stress again that the objective of this paper is to offer a useful structured partition of the image, to be considered for example as a posterior classification process. These possible classes can be derived, for example, from any existing fuzzy classification technique, e.g., fuzzy C-means (see [1]). Alternatively, we can consider the approach proposed in [11,12] (see also [8,10]), taking now advantage of the neighborhood information associated to those regions we have obtained.

Also, for images of greater dimensions, the segmentation process introduced in this paper can be viewed as a preprocessing tool in order to reduce the amount of information needed to explain the image, allowing faster posterior studies (we shall not be classifying pixels but regions).

Notice that the obtained segmentation does not imply any comparison of measures observed in one region with a certain pattern or with any other region. In fact, two disjoint homogeneous regions are not guaranteed to belong to the same class even if the properties of the pixels they contain are highly similar; and two disjoint homogeneous regions can be assigned to the same class even if the pixels in one region are much different from pixels in the other.

Classification is a more complex problem than segmentation, but it usually requires a previous segmentation

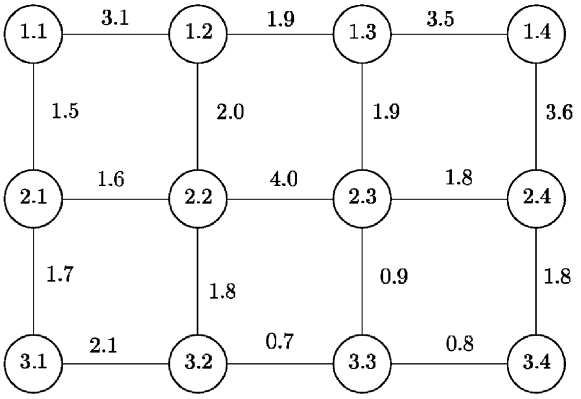

Fig. 1. Pixels network of Example 2.1.

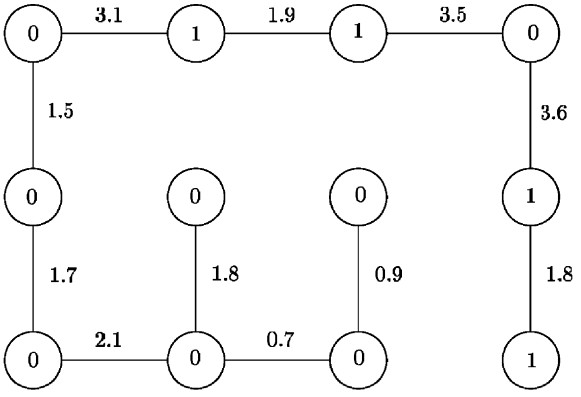

Fig. 2. Spanning tree of pixels network of Example 2.1 ($\alpha = 3.1$).

analysis. The connected regions obtained by means of the procedure proposed in this paper should provide a useful visualization of potential classes that may be considered by decision makers in order to explain the image.

As any heuristic approach, our algorithm is not free of some potential misbehavior, which will suggest future improvements (for example, consideration of additional rings surrounding each pixel, and not only adjacent pixels, can be tried when we face extremely *smooth* images with very small variation between adjacent pixels). Granularity of the output depends on the number of iterations, so some expertise is needed in order to get useful information: too few iterations will produce oversimplification, and therefore a wrong segmentation, but accuracy obtained by means of too many iterations may not be manageable be decision makers (as commonly accepted, simple models may not be accurate, but too complex models may be useless).

Nevertheless, coloring techniques should play a key role in the future, in order to help decision makers to capture a global view of complex images, including those where regions do not show clear boundaries. Without appropriate representation techniques, decision makers may look at some sophisticated models as *black boxes*, and therefore reject them in practice.

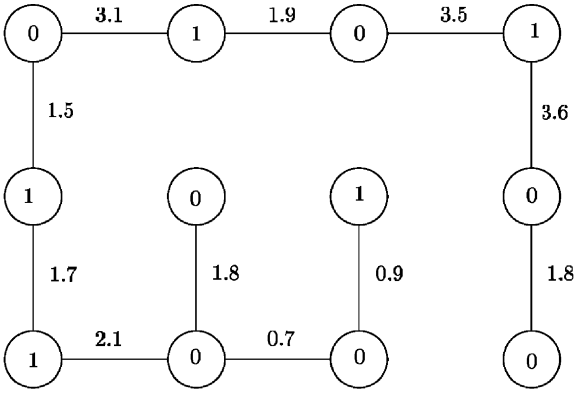

Fig. 3. Spanning tree of pixels network of Example 2.1 ($\alpha = 1.0$).

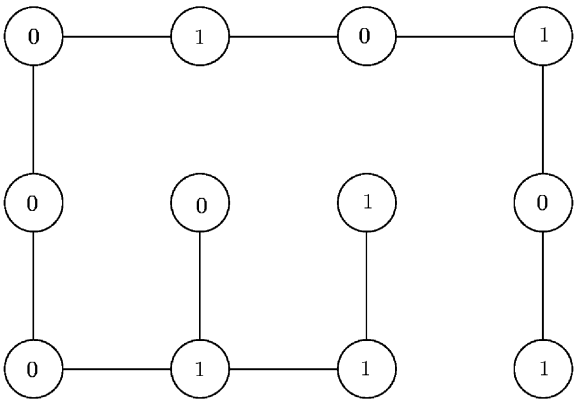

Fig. 4. First binary coloring of pixels network of Example 2.1.

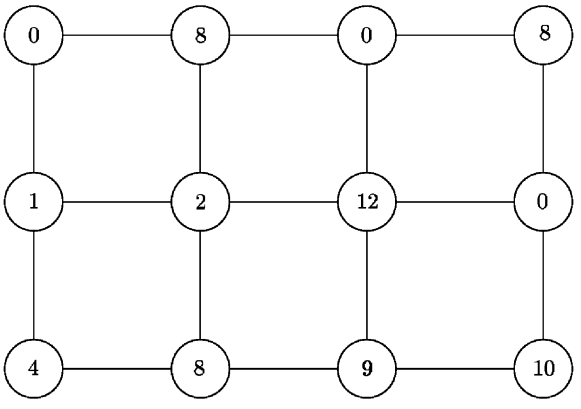

Fig. 5. Coloring of pixels network of Example 2.1.

Acknowledgements

This research has been partially supported by the Government of Spain, Grant BFM2002-0281. We would also like to express our gratitude to an anonymous referee, whose

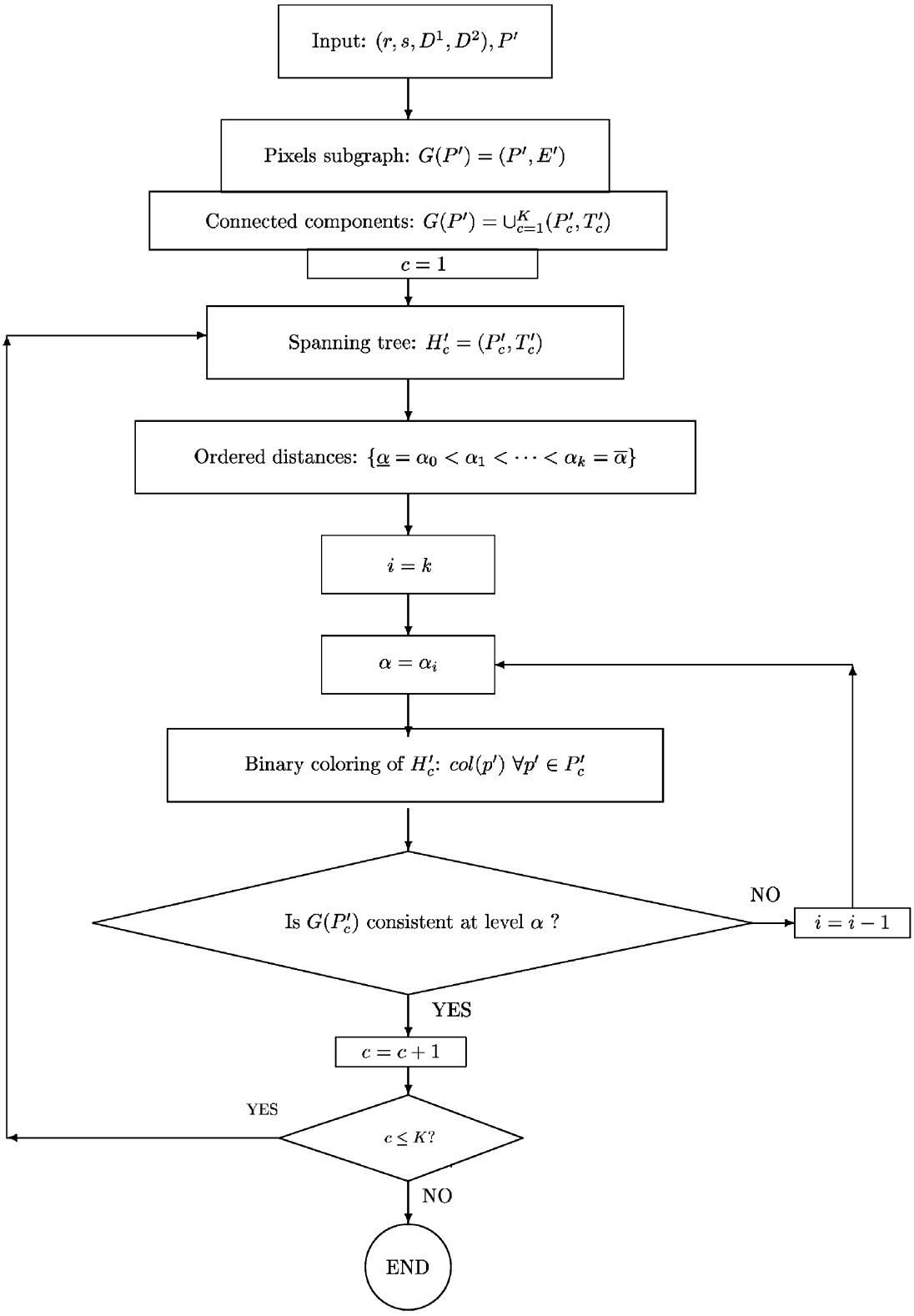

Fig. 6. Procedure bincol(P', α, col).

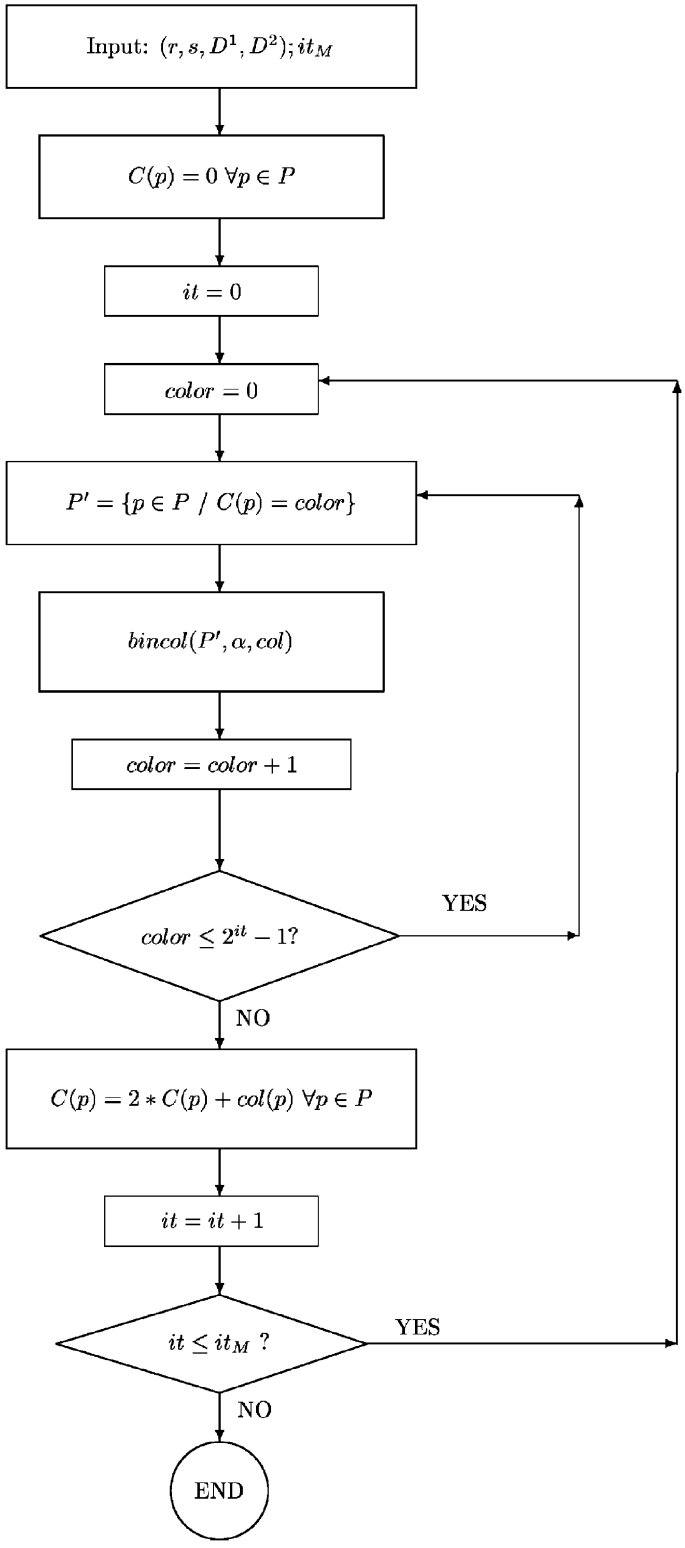

Fig. 7. Procedure crucol(r, s, D^1, D^2, it_M, C).

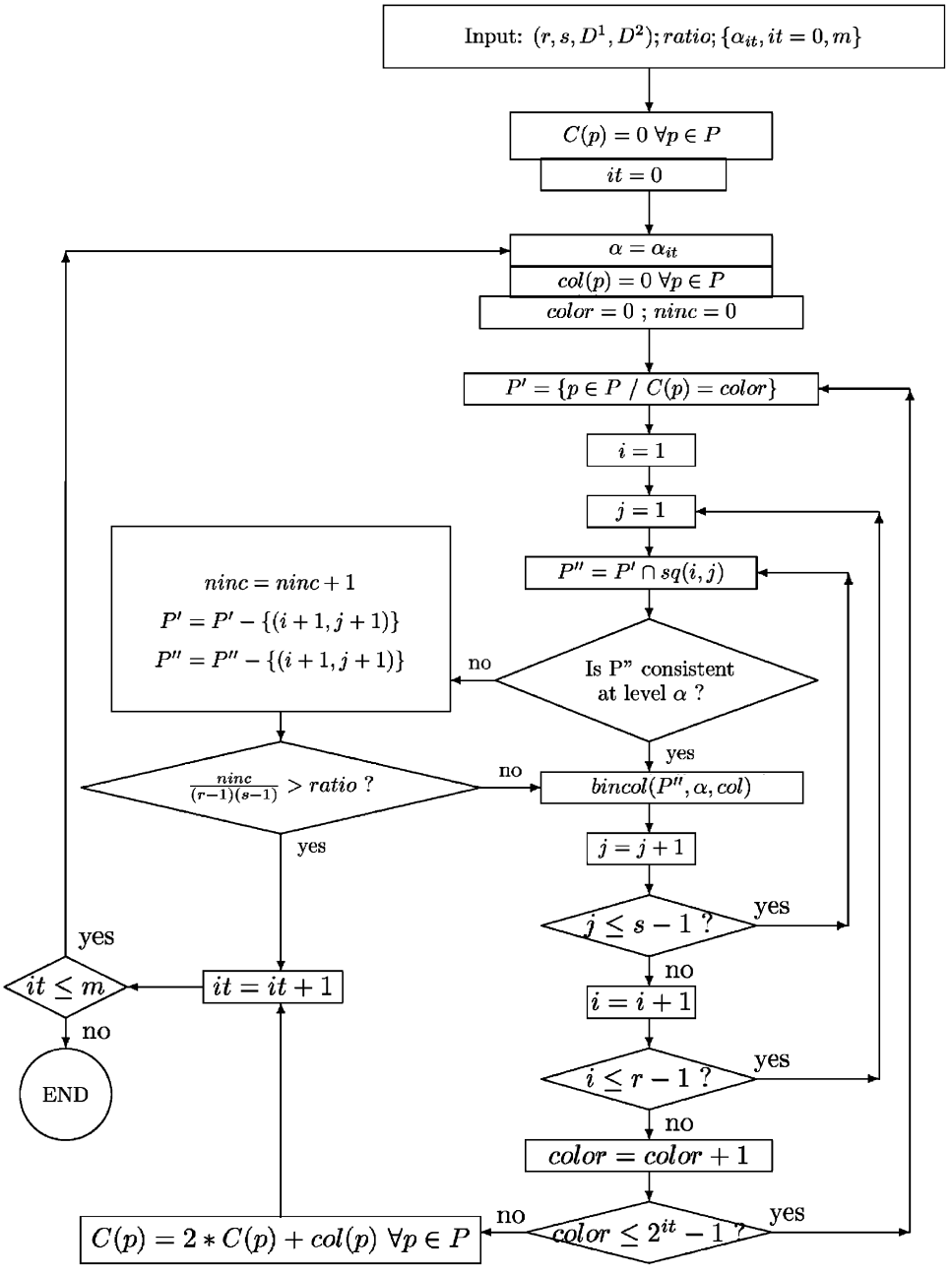

Fig. 8. Procedure rxcol(r, s, D^1, D^2, C, ratio, $\{\alpha_{it}, it = 0, m\}$).

comments allowed a deep improvement in our research (some of them those comments have been included in this paper).

Appendix A. Figures

Pixels network, spanning tree at level $\alpha = 3.1$, spanning tree at level $\alpha = 1.0$, binary coloring of pixels network, final coloring of pixels network are depicted in Figs. 1–5.

Appendix B. Diagrams

The procedure bincol, procedure crucol and procedure rxcol are depicted in Figs. 6–8.

References

[1] Bezdek JC. Pattern recognition with fuzzy objective function algorithms. New York: Plenum Press; 1981.

[2] Bezdek JC, Harris JD. Fuzzy partitions and relations: an axiomatic basis for clustering. Fuzzy Sets and Systems 1978;1:111–27.
[3] Foody GM. The continuum of classification fuzziness in thematics mapping. Photogrammetric Engineering and Remote Sensing 1999;65:443–51.
[4] Kerre EE, Nachtegael M. Fuzzy techniques in image processing. Heidelberg: Physica-Verlag; 2000.
[5] Pal SK, Ghosh A, Kundu MK. Soft computing for image processing. Heidelberg: Physica-Verlag; 2000.
[6] Muñoz S, Ortuño T, Ramírez J, Yáñez J. Coloring fuzzy graphs. Omega 2005;33(3):211–21.
[7] Yáñez J, Ramírez J. The robust coloring problem. European Journal of Operational Research 2003;148: 546–58.
[8] Amo A, Gómez D, Montero J, Biging G. Relevance and redundancy in fuzzy classification systems. Mathware and Soft Computing 2001;8:203–16.
[9] Amo A, Montero J, Biging G. Classifying pixels by means of fuzzy relations. International Journal of General Systems 2000;29:605–21.
[10] Amo A, Montero J, Fernández A, López M, Tordesillas J, Biging G. Spectral fuzzy classification: an application. IEEE Transactions on Systems Man and Cybernetics (C) 2002;32: 42–8.
[11] Amo A, Montero J, Biging G, Cutello V. Fuzzy classification systems. European Journal of Operational Research 2004;156:459–507.
[12] Amo A, Montero J, Cutello V. On the principles of fuzzy classification. Proceedings of the annual North American fuzzy information processing society conference (NAFIPS). 1999. p. 675–79.
[13] Pardalos PM, Mavridou T, Xue J. The graph coloring problem: a bibliographic survey. In: Du DZ, Pardalos PM, editors. Handbook of combinatorial optimization, vol. 2. Boston: Kluwer Academic Publishers; 1998. p. 331–95.