International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 11, No. 1, pp. 75 – 100, 2014

A SPATIAL SEGMENTATION METHOD

DUMITRU DAN BURDESCU, MARIUS BREZOVAN, LIANA STANESCU, COSMIN STOICA SPAHIU

Computers and information Technology Department, Faculty of Automatics, Computers and Electronics,
University of Craiova,
Craiova, Romania
dburdescu@yahoo.com, {mbrezovan, Stanescu, stoica.cosmin}@software.ucv.ro

The problem of partitioning images into homogenous regions or semantic entities is a basic problem for identifying relevant objects. Visual segmentation is related to some semantic concepts because certain parts of a scene are pre-attentively distinctive and have a greater significance than other parts. However, even if image segmentation is a heavily researched field, extending the algorithms to spatial has been proven not to be an easy task. A true spatial segmentation remains a difficult problem to tackle due to the complex nature of the topology of spatial objects, the huge amount of data to be processed and the complexity of the algorithms that scale with the new added dimension. Unfortunately there are huge amount of papers for planar images and segmentation methods and most of them are graph-based for planar images. There are very few papers for spatial segmentation methods. The major concept used in graph-based spatial segmentation algorithms is the concept of homogeneity of regions. For color spatial segmentation algorithms the homogeneity of regions is color-based, and thus the edge weights are based on color distance. Early graph-based methods use fixed thresholds and local measures in finding a spatial segmentation. Complex grouping phenomena can emerge from simple computation on these local cues. As a consequence we consider that a spatial segmentation method can detect visual objects from images if it can detect at least the most objects. The aim in this paper is to present a new and efficient method to detect visual objects from color spatial images and to extract their color and geometric features, in order to determine later the contours of the visual objects and to perform syntactic analysis of the determined shapes. In this paper we extend our previous work for planar segmentation by adding a new step in the spatial segmentation algorithm that allows us to determine regions closer to it. The key to the whole algorithm of spatial segmentation is the honeycomb cells.

Keywords: Spatial Segmentation; Graph-based segmentation; Adaptive algorithms

1. Introduction and Related Work

Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous. Formally, segmentation can be defined as follows.

Let F be the set of all pixels/voxels and P(.) be a uniformity (homogeneity) predicate defined on groups of connected pixels/voxels, then segmentation is a partitioning of the set F into a set of connected subsets or regions

$(S_1, S_2, \ldots, S_n)$ such that $\cup^{i=1}_{n} S_i = F$ with $S_i \cap S_j = \emptyset$ when $i \neq j$. The uniformity predicate $P(S_i)$ is true for all regions S_i and $P(S_i \cup S_j)$ is false when S_i is adjacent to S_j .

This definition can be applied to all types of images.

The goal of segmentation is typically to locate certain objects of interest which may be depicted in the image. Segmentation could therefore be seen as a computer vision problem. A simple example of segmentation is threshold a grayscale image with a fixed threshold 't': each pixel/voxel 'p' is assigned to one of two classes, P_0 or P_1, depending on whether I(p) < t or I(p) >= t.

For intensity images (i.e., those represented by point-wise intensity levels), four popular segmentation approaches are: threshold techniques, edge based methods, region-based techniques, and connectivity-preserving relaxation methods.

Threshold techniques make decisions based on local pixel/voxel information and are effective when the intensity levels of the objects fall squarely outside the range of levels in the background. Because spatial information is ignored, however, blurred region boundaries can create havoc.

Edge-based methods center is around contour detection: their weakness in connecting together broken contour lines make them, too, prone to failure in the presence of blurring.

A region-based method usually proceeds as follows: the image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving perhaps homogeneity or sharpness of region boundaries. Over-stringent criteria create fragmentation; lenient ones overlook blurred boundaries and over-merge.

A connectivity-preserving relaxation-based segmentation method, usually referred to as the active contour model, starts with some initial boundary shape represented in the form of spline curves, and iteratively modifies it by applying various shrink/expansion operations according to some energy function. Although the energy-minimizing model is not new, coupling it with the maintenance of an "elastic" contour model gives it an interesting new twist. As usual with such methods, getting trapped into a local minimum is a risk against which one must guard and this is no easy task.

Grouping can be formulated as a graph partitioning and optimization problem [1] and [2].

The graph theoretic formulation of image segmentation is as follows:

1. The set of points in an arbitrary feature space are represented as a weighted undirected graph G = (V,E), where the nodes of the graph are the points in the feature space

2. An edge is formed between every pair of nodes yielding a dense or complete graph.

3. The weight on each edge, w(i, j) is a function of the similarity between nodes 'i' and 'j'.

4. Partition the set of vertices into disjoint sets V_1, V_2, … , V_k where by some measure the similarity among the vertices in a set V_i is high and, across different sets V_i, V_j is low.

To partition the graph in a meaningful manner, we also need to:

- Pick an appropriate criterion (which can be computed from the graph) to optimize which would result in a good segmentation.
- Finding an efficient way to achieve the optimization.

In the image segmentation and data clustering community [3], there has been much previous work using variations of the minimal spanning tree or limited neighborhood set approaches [4]. Although those use efficient computational methods, the segmentation criteria used in most of them are based on local properties of the graph. Because perceptual grouping is about extracting the global impressions of a scene, as we saw earlier, this partitioning criterion often falls short of this main goal.

There are huge of papers for planar images and segmentation methods and most graph-based for planar images and few papers for spatial segmentation methods.

Early graph-based methods use fixed thresholds and local measures in finding a planar segmentation. The planar segmentation criterion is to break the edges with the largest weight, which reflect the low-cost connection between two elements. To overcome the problem of fixed threshold in [5], Urquhar determines the normalized weight of an edge by using the smallest weight incident on the vertices touching that edge. Other methods [6], [7] use an adaptive criterion that depends on local properties rather than global ones. In contrast with the simple graph-based methods, cut-criterion methods capture the non-local cuts in a planar graph are designed to minimize the similarity between pixels that are being split [8] [9]. The normalized cut criterion [9] takes into consideration self similarity of regions. An alternative to the graph cut approach is to look for cycles in a planar graph embedded in the image plane. For example in [10] the quality of each cycle is normalized in a way that is closely related to the normalized cuts approach. Other approaches to planar image segmentation consist of splitting and merging regions according to how well each region fulfills some uniformity criterion. Such methods [11] use a measure of uniformity of a region. In contrast [6] and [7] use a pair-wise region comparison rather than applying a uniformity criterion to each individual region. Complex grouping phenomena can emerge from simple computation on these local cues [12]. A number of approaches to segmentation are based on finding compact clusters in some feature space [13]. A recent technique using feature space clustering [14] first transforms the data by smoothing it in a way that preserves boundaries between regions.

Our previous works for planar images [15] and [16] are related to the works in [6] and [7] in the sense of pair-wise comparison of region similarity. In these papers we extend our previous work for planar images by adding a new step in the spatial segmentation algorithm that allows us to determine regions closer to it. The internal contrast of a component C represents the maximum weight of edges connecting vertices from C, and the external contrast between two components represents the maximum weight of edges connecting vertices from these two components. These measures are in our opinion closer to the human perception.

2. Constructing a Virtual Tree-Hexagonal Structure

The low-level system for spatial image segmentation and boundary extraction of visual objects described in this section can be designed to be integrated in a general framework of indexing and semantic image processing. The framework uses color and geometric

features of image regions in order to: (a) determine visual objects and their contours, and also (b) to extract specific color and geometric information from these objects to be further used into a higher-level image processing system.

The pre-processing module is used mainly to blur the initial RGB spatial image in order to reduce the image noise. Then the segmentation module creates virtual cells of prisms with tree-hexagonal structure defined on the set of the image voxels of the input spatial image and a spatial triangular grid graph having tree-hexagons as cells of vertices. In order to allow a unitary processing for the multi-level system at this level we store, for each determined component C, the set of the tree-hexagons contained in the region associated to C and the set of tree-hexagons located at the boundary of the component. In addition for each component the dominant color of the region is extracted. This color will be further used in the post-processing module if any. The contour extraction module determines for each spatial segment of the image its boundary. The boundaries of the de determined visual objects are closed contours represented by a sequence of adjacent tree-hexagons. At this level a linked list of points representing the contour is added to each determined component. The post-processing module (if any) extracts representative information for the above determined visual objects and their contours in order to create an efficient index for a semantic image processing system.

A spatial image processing task contains mainly three important components: acquisition, processing and visualization. After the acquisition stage an image is sampled at each point on a three dimensional grid storing intensity or color information and implicit location information for each sample. The rectangular grid is the most dominant of any grid structure in image processing and conventional acquisition devices acquire square sampled images. An important advantage of using rectangular grid is the fact that the visualization stage uses directly the square pixels of the digitized image. We do not use a rectangular grid and hexagonal lattice model because of the additional actions involving the double conversion between square and tree-hexagonal voxels. However we intent to use some of the advantages of the tree-hexagonal grid such as uniform connectivity. This implies that there will be less ambiguity in defining boundaries and regions [1]. As a consequence we construct a virtual tree-hexagonal structure over the square voxels of an input spatial image, as presented in Figure 1.

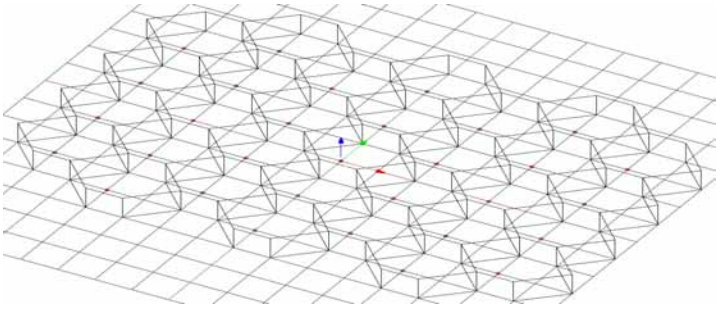

Fig. 1. Virtual Tree-Hexagonal structure constructed on the input image voxels.

This virtual tree-hexagonal grid is not a tree-hexagonal lattice because the constructed hexagons are not regular.

Let I be a spatial initial image having the dimension w×h×z (e.g. a matrix having 'h' rows, 'w' columns and 'z' deep of matrix voxels). In order to construct a tree-hexagonal grid on these voxels we retain an eventually smaller image with

$$h' = h-(h-1) \bmod 2,$$
$$w' = w-w \bmod 4,$$
$$z' = z- z \bmod 4. \quad (1)$$

In the reduced image at most the last line of voxels and at most the last three columns and deep of matrix of voxels are lost, assuming that for the initial image h > 3 and w > 4 and z > 4, that is a convenient restriction for input images.

Each tree-hexagon from the tree-hexagonal grid contains sixteen voxels: such twelve voxels from the frontier and four interior frontiers of voxels.

Because tree-hexagons voxels from an image have integer values as coordinates we select always the left up voxel from the four interior voxels to represent with approximation the gravity center of the tree-hexagon, denoted by the pseudo-gravity center.

We use a simple scheme of addressing for the tree-hexagons of the tree-hexagonal grid that encodes the spatial location of the pseudo-gravity centers of the hexagons as presented in Figure 1.

Let w×h×z the three dimension of the initial spatial image verifying the previous restriction (e.g. h mod 2 = 1, w mod 4 = 0, z mod 4 = 0, h ≥ 3 and w≥4 and z≥4). Given the coordinates (l,c,d) of a voxel 'p' from the input spatial image, we use the linearised function, $ip_{w,h,z}(l,c,d) = (l-1)w+c+z$, in order to determine an unique index for the voxel.

Let ps be the sub-sequence of the voxels from the sequence of the voxels of the initial spatial image that correspond to the pseudo-gravity center of tree-hexagons, and ls, cs and ds the sequence of tree-hexagons constructed over the voxels of the initial spatial image. For each voxel 'p' from the sequence ps having the coordinates (l,c,d), the index of the corresponding tree-hexagon from the sequence ls, cs and ds are given by the following relation:

$$f\,h_{w,h,z}(l,c,d) = [(l-2)w+c+d-2l]/4 +1 \quad (2)$$

Remark: It is easy to verify the following two properties related to the function *f* h:

1. The value [(l−2)w+c+d−2l] is always divisible by 4.

2. Let 'p' be a voxel from the sub-sequence 'p_s' of voxels representing the pseudo-gravity center of tree-hexagons, having the coordinates (l,c,d) and 'i' its index in this subsequence, then $p = p_{si}$.

In this case the following relation holds:

$$f\,h_{w,h,z}(l,c,d) = i. \quad (3)$$

The second remark states in fact that the scheme of addressing for the tree-hexagons is linear and it has a natural order induced by the sub-sequence of voxels representing the pseudo-gravity center of tree-hexagons.

Moreover it is easy to verify that the function '*f*h' defined by the relation (2) is bijective. Its inverse function is given by:

$$f\,h{-}1_{\,w,h,z}\,(k) = (l,c,d) \tag{4}$$

where:

$$l = (2+ 4(k-1)/w \quad \text{if } h < w$$

$$l = 2+ 4(k-1)/w + t_w \quad \text{if } h \geq w, \text{ and } h = t_w + h', \tag{5}$$

$$c = 4(k-1)+2l-(l-2)w, \tag{6}$$

$$d = 4(k-1)+2l-(l-2)w. \tag{7}$$

Relations (4), (5), (6) and (7) allow us to uniquely determine the coordinates of the voxel representing the pseudo-gravity center of a tree-hexagon specified by its index (its address). In addition these relations allow us to determine the sequence of coordinates of all sixteen voxels contained into a tree-hexagon with an address 'k'.

The sub-sequence 'ps' of the voxels representing the pseudo-gravity center and the function 'f h' defined by the relation (2) allow to determine the sequence of the tree-hexagons 'Hs' that is used by the segmentation and contour detection algorithms. After the processing step the relations (4), (5), (6) and (7) allow to update the voxels of the spatial initial image for the visualization step.

Each tree-hexagon represents an elementary item and the entire virtual tree-hexagonal structure represents a triangular grid graph, G = (V,E), where each tree-hexagon 'H' in this structure has a corresponding vertex v ∈V. The set E of edges is constructed by connecting tree-hexagons that are neighbors in a 20-connected sense. The vertices of this graph correspond to the pseudo-gravity centers of the hexagons from the tree-hexagonal grid and the edges are straight lines connecting the pseudo-gravity centers of the neighboring hexagons, as presented in Figure 2.

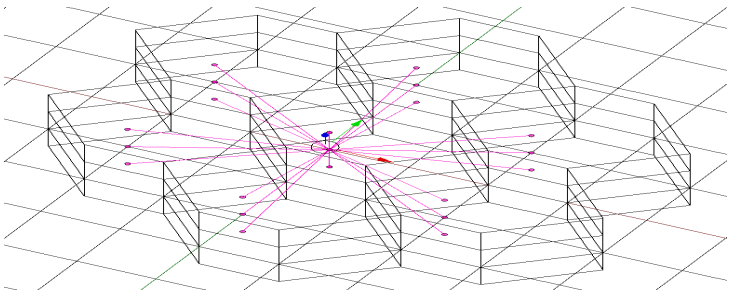

Fig. 2. The triangular grid graph constructed on the pseudo-gravity centers of the tree-hexagonal grid

There are two main advantages when using tree-hexagons instead of voxels as elementary piece of information:

• The amount of memory space associated to the spatial graph vertices is reduced. Denoting by 'np' the number of voxels of the initial spatial image, the number of the resulted tree-hexagons is always less than np/4, and thus the cardinal of both sets V and E is significantly reduced;

• The algorithms for determining the visual objects and their contours are much faster and simpler in this case.

We associate to each tree-hexagon 'H' from V two important attributes representing its dominant color and the coordinates of its pseudo-gravity center, denoted by g(h). The dominant color of a tree-hexagon is denoted by c(h) and it represents the color of the voxel of the tree-hexagon which has the minimum sum of color distance to the other twenty voxels. Each tree- hexagon 'H' in the tree-hexagonal grid is thus represented by a single point, g(h), having the color c(h). By using the values g(h) and c(h) for each tree-hexagon information related to all voxels from the initial image is taken into consideration by the spatial segmentation algorithm.

3. Spatial Segmentation Algorithm

Let $V = \{h_1, \ldots, h_{|V|}\}$ be the set of tree-hexagons constructed on the spatial image voxels as presented in previous section and G = (V,E) be the undirected spatial grid-graph, with E containing pairs of tree-hexagons (honey-beans cell) that are neighbors in a 20-connected sense. The weight of each edge $e = (h_i,h_j)$ is denoted by w(e), or similarly by $w(h_i,h_j)$, and it represents the dissimilarity between neighboring elements 'h_i' and 'h_j' in a some feature space. Components of an image represent compact regions containing voxels with similar properties. Thus the set V of vertices of the graph G is partitioned into disjoint sets, each subset representing a distinct visual object of the initial image.

As in other graph-based approaches [16] we use the notion of segmentation of the set V for planar/spatial image. A segmentation, S, of V is a partition of V such that each component $C \in S$ corresponds to a connected component in a spanning sub-graph GS = (V,ES) of G, with $ES \subseteq E$.

The set of edges $E - ES$ that are eliminated connect vertices from distinct components. The common boundary between two connected components $C', C'' \in S$ represents the set of edges connecting vertices from the two components:

$$cb(C',C'') = \{(h_i,h_j) \in E \mid h_i \in C',\ h_j \in C''\} \qquad (8)$$

The set of edges $E-ES$ represents the boundary between all components in S. This set is denoted by bound(S) and it is defined as follows:

$$bound(S) = \cup_{C',C'' \in S}\ cb(C',C''). \qquad (9)$$

In order to simplify notations throughout the paper we use C_i to denote the component of a segmentation S that contains the vertex $h_i \in V$.

We use the notions of segmentation too fine and too coarse as defined in [6] that attempt to formalize the human perception of visual objects from an input image. A segmentation S is too fine if there is some pair of components $C', C'' \in S$ for which there

is no evidence for a boundary between them. S is too coarse when there exist a proper refinement of S that is not too fine. The key element in this definition is the evidence for a boundary between two components.

The goal of a segmentation method is to determine a proper segmentation, which represent visual objects from an image.

Definition 1: Let G = (V,E) be the undirected spatial graph constructed on the tree-hexagonal structure of an input image, with V = $\{h_1, \ldots, h_{|V|}\}$. A proper segmentation of V, is a partition S of V such that there exists a sequence $[S_i, S_{i+1}, \ldots, S_{f-1}, S_f]$ of segmentations of V for which:

- S = S_f is the final segmentation and S_i is the initial segmentation,
- S_j is a proper refinement of S_{j+1} (i.e., $S_j \subset S_{j+1}$) for each j = i, . . . , f −1,
- segmentation S_j is too fine, for each j = i, . . . , f −1,
- any segmentation S_l such that $S_f \subset S_l$, is too coarse,
- segmentation S_f is neither too coarse nor too fine.

In the above definition S_a is a refinement of S_b in the sense of partitions, i.e. every set in S_a is a subset of one of the sets in S_b. We say that S_a is a proper refinement of S_b if S_a is a refinement of S_b and $S_a \neq S_b$. In the case of a proper refinement, S_a is obtained by splitting one or more components from S_b, or similarly, S_b is obtained by merging one or more components from S_a.

Let $C', C'' \in Sa$ be two components obtained by splitting a component $C \in S_b$. In this case C′ and C″ have a common boundary, $c_b(C', C'') \neq \emptyset$.

Our spatial segmentation algorithm starts with the most refined segmentation, $S_0 = \{\{h_1\}, \ldots, \{h_{|V|}\}\}$ and it constructs a sequence of segmentations until a proper segmentation is achieved. Each segmentation S_j is obtained from the segmentation S_{j-1} by merging two or more connected components for there is no evidence for a boundary between them. For each component of a spatial segmentation a spanning tree is constructed and thus for each segmentation we use an associated spanning forest [18].

The evidence for a boundary between two components is determined taking into consideration some features in some model of the image. When starting, for a certain number of segmentations the only considered feature is the color of the regions associated to the components and in this case we use a color-based region model. When the components became complex and contain too much tree-hexagons, the color model is not sufficient and geometric features together with color information are considered. In this case we use a syntactic-based regions model with a color-based region model for regions. In addition syntactic features bring supplementary information for merging similar regions in order determine salient objects.

For the sake of simplicity we will denote this region model as syntactic-based region model.

As a consequence, we split the sequence of all segmentations,

$$S_{i\,f} = [S_0, S_1, \ldots, S_{k-1}, S_k], \tag{10}$$

in two different subsequences, each subsequence having a different region model,

$$S_i = [S_0, S_1, \ldots, S_{t-1}, S_t],$$

$$S_f = [S_t, S_{t+1}, \ldots, S_{k-1}, S_k], \quad (11)$$

where S_i represents the color-based segmentation sequence, and S_f represents the syntactic-based segmentation sequence.

The final segmentation S_t in the color-based model is also the initial segmentation in the syntactic-based region model.

For each sequence of segmentations we develop a different algorithm. Moreover we use a different type of spanning tree in each case: a maximum spanning tree in the case of the color-based segmentation, and a minimum spanning tree in the case of the syntactic-based segmentation. More precisely our method determines two sequences of forests of spanning trees,

$$F_i = [F_0, F_1, \ldots, F_{t-1}, F_t],$$
$$F_f = [F_{t'}, F_{t'+1}, \ldots, F_{k'-1}, F_{k'}], \quad (12)$$

each sequence of forests being associated to a sequence of segmentations.

The first forest from Fi contains only the vertices of the initial undirected spatial graph, $F_0 = (V, \emptyset)$, and at each step some edges from E are added to the forest $F_l = (V, E_l)$ to obtain the next forest, $F_{l+1} = (V, E_{l+1})$. The forests from Fi contain maximum spanning trees and they are determined by using a modified version of Kruskal's algorithm [18], where at each step the heaviest edge (u,v) that leaves the tree associated to 'u' is added to the set of edges of the current forest.

The second subsequence of forests that correspond to the subsequence of segmentations Sf contains forests of minimum spanning trees and they are determined by using a modified form of Boruvka's algorithm [19]. This sequence uses as input a new graph, $G' = (V', E')$, which is extracted from the last forest, F_t , of the sequence F_i. Each vertex 'v' from the set V' corresponds to a component C_v from the segmentation S_t (i.e. to a region determined by the previous algorithm). At each step the set of new edges added to the current forest are determined by each tree T contained in the forest that locates the lightest edge leaving T. The first forest from F_f contains only the vertices of the graph G', $F_{t'} = (V', \emptyset)$.

In this section we focus on the definition of a logical predicate that allow us to determine if two neighboring regions represented by two components, $C_{l'}$ and $C_{l''}$, from a segmentation S_l can be merged into a single component C_{l+1} of the segmentation S_{l+1}. Two components, $C_{l'}$ and $C_{l''}$, represent neighboring (adjacent) regions if they have a common boundary:

$$\text{ad j}(Cl', Cl'') = \text{true if cb}(Cl', Cl'') \neq \emptyset,$$
$$\text{ad j}(Cl', Cl'') = \text{f alse if cb}(Cl', Cl'') = \emptyset \quad (13)$$

We use a different predicate for each region model, color-based and syntactic-based respectively.

$$PED(e,u) = [w_R(R_e - R_u)^2 + w_G(G_e - G_u)^2 + w_B(B_e - B_u)^2\ ]\ ½ \quad (14)$$

where the weights for the different color channels, w_R, w_G, and w_B verify the condition $w_R + w_G + w_B = 1$. Based on the theoretical and experimental results on spectral and real world data sets, Gijsenij et al. [20] is concluded that the PED distance with

weight-coefficients (w_R =0.26, w_G = 0.70, w_B =0.04) correlates significantly higher than all other distance measures including the angular error and Euclidean distance.

In the color model regions are modeled by a vector in the RGB color space. This vector is the mean color value of the dominant color of tree-hexagons belonging to the regions.

The evidence for a boundary between two regions is based on the difference between the internal contrast of the regions and the external contrast between them [6] and [17]. Both notions of internal contrast and external contrast between two regions are based on the dissimilarity between two colors.

Let h_i and h_j representing two vertices in the spatial graph G =(V,E), and let $w_{col}(h_i,h_j)$ representing the color dissimilarity between neighboring elements h_i and h_j, determined as follows:

$$w_{col}(h_i,h_j) = PED(c(h_i),c(h_j)) \quad \text{if } (h_i,h_j) \in E,$$
$$w_{col}(h_i,h_j) = \infty \quad \text{otherwise,} \tag{15}$$

where PED(e,u) represents the perceptual Euclidean distance with weight-coefficients between colors ‘e‘and ‘u’, as defined by Equation (14), and c(h) represents the mean color vector associated with the tree-hexagon ‘H’. In the color-based segmentation, the weight of an edge (h_i,h_j) represents the color dissimilarity, $w(h_i,h_j) = w_{col}(h_i,h_j)$.

Let Sl be a segmentation of the set V.

We define the internal contrast or internal variation of a component $C \in S_l$ to be the maximum weight of the edges connecting vertices from C:

$$IntVar(C) = \max_{(h_i,h_j)\in C}(w(h_i,h_j)). \tag{16}$$

The internal contrast of a component C containing only one hexagon is zero: IntVar(C) = 0, if |C| = 1.

The external contrast or external variation between two components, $C',C'' \in S$ is the maximum weight of the edges connecting the two components:

$$ExtVar(C',C'') = \max_{(h_i,h_j)\in cb(C',C'')}(w(h_i,h_j)). \tag{17}$$

We have chosen the definition of the external contrast between two components to be the maximum weight edge connecting the two components and not to be the minimum weight, as in [6] because: (a) it is closer to the human perception (in the sense of the perception of the maximum color dissimilarity), and (b) the contrast is uniformly defined (as maximum color dissimilarity) in the two cases of internal and external contrast.

The maximum internal contrast between two components, $C',C'' \in S$ is defined as follows:

$$IntVar(C',C'') = \max(IntVar(C'), IntVar(C'')), \tag{18}$$

The comparison predicate between two neighboring components C′ and C″ (i.e., ad j(C′,C″) = true) determines if there is an evidence for a boundary between C′ and C″ and it is defined as follows:

$$di\,f\, f_{col}(C',C'') = true, \quad \text{if } ExtVar(C',C'') > IntVar(C',C'')+\tau(C',C''),$$
$$di\,f\, f_{col}(C',C'') = f\,alse, \quad \text{if } ExtVar(C',C'') \le IntVar(C',C'')+\tau(C',C''), \tag{19}$$

with the the adaptive threshold τ(C′,C″) given by

$$\tau(C',C'') = \tau/\min(|C'|, |C''|) \ , \qquad (20)$$

where |C| denotes the size of the component C (i.e. the number of the tree-hexagons contained in C) and the threshold 'τ' is a global adaptive value defined by using a statistical model.

The predicate di f f_{col} can be used to define the notion of segmentation too fine and too coarse in the color-based region model.

Definition 2: Let G = (V,E) be the undirected spatial graph constructed on the tree-hexagonal structure of a spatial image and S a color-based segmentation of V. The segmentation S is too fine in the color-based region model if there is a pair of components $C', C'' \in S$ for which

$$\text{ad j}(C',C'') = \text{true} \wedge \text{di } f\ f_{col}(C',C'') = \text{false}.$$

Definition 3: Let G = (V,E) be the undirected spatial graph constructed on the tree-hexagonal structure of a spatial image and S a segmentation of V. The segmentation S is too coarse if there is a proper refinement of S that is not too fine.

There are many existing systems for arranging and describing colors, such as RGH, YUV, HSV, LUV, CIELAV, Munsell system, etc. We decided to use the RGB color space because it is efficient and no conversion is required. Although it also suffers from the non-uniformity problem where the same distance between two color points within the color space may be perceptually quite different in different parts of the space, within a certain color threshold it is still definable in terms of color consistency. We use the perceptual Euclidean distance with weight-coefficients (PED) as the distance between two colors.

Let G= (V,E) be the initial spatial graph constructed on the tree-hexagonal structure of a spatial image. The proposed segmentation algorithm will produce a proper segmentation of V according to the Definition 1. The sequence of segmentations, S_i^f, as defined by Equation (10), and its associated sequence of forests of spanning trees, F_i^f, as defined by Equation (12), will be iteratively generated as follows:

• The color-based sequence of segmentations, Si, as defined by Equation (11), and its associated sequence of forests, F_i, as defined by Equation (12), will be generated by using the color-based region model and a maximum spanning tree construction method based on a modified form of the Kruskal's algorithm [18].

• The syntactic-based sequence of segmentations, S_f, as defined by Equation (11), and its associated sequence of forests, F^f, as defined by Equation (12), will be generated by using the syntactic-based model and a minimum spanning tree construction method based on a modified form of the Boruvka's algorithm.

The general form of the spatial segmentation procedure is presented in Algorithm 1

Algorithm 1 Spatial Segmentation Algorithm

1: **procedure SEGMENTATION(l, c, d, P, H, Comp)

2: Input l, c, d, P

```
3: Output H, Comp
4: *H ←CREATEHEXAGONALSTRUCTURE(l, c, d, P)
5: *G←CREATEINITIALGRAPH(l, c, d, P,H)
6: *CREATECOLORPARTITION(G,H,Bound)
7: *G′ ←EXTRACTGRAPH(G,Bound, thkg)
8: *CREATESYNTACTICPARTITION(G,G′, thkg)
9: *Comp ←EXTRACTFINALCOMPONENTS(G′)
10: end procedure
```

The input parameters represent the image resulted after the pre-processing operation: the array 'P' of the spatial image voxels structured in 'l' lines, 'c' columns and 'd' depths. The output parameters of the segmentation procedure will be used by the contour extraction procedure: the tree-hexagonal grid stored in the array of tree-hexagons H, and the array Comp representing the set of determined components associated to the salient objects in the input spatial image. The global parameter thkg is the thresholds. The global parameter threshold is determinate by using Algorithm 1.

The color-based spatial segmentation and the syntactic-based spatial segmentation are determined by the procedures CREATECOLORPARTITION and CREATESYNTACTICPARTITION respectively.

The color-based and syntactic-based segmentation algorithms use the tree-hexagonal structure H created by the function CREATEHEXAGONALSTRUCTURE over the voxels of the initial spatial image, and the initial grid of spatial graph G created by the function CREATEINITIALGRAPH. Because the syntactic-based segmentation algorithm uses a graph contraction procedure, CREATESYNTACTICPARTITION uses a different graph, G', extracted by the procedure EXTRACTGRAPH after the color-based segmentation finishes.

Both algorithms for determining the color-based and syntactic-based segmentation use and modify a global variable (denoted by 'CC') with two important roles:

• to store relevant information concerning the growing forest of spanning trees during the spatial segmentation (maximum spanning trees in the case of the color-based segmentation, and minimum spanning trees in the case of syntactic-based segmentation),

• to store relevant information associated to components in a segmentation in order to extract the final components because each spatial tree in the spatial forest represent in fact a component in each segmentation S in the spatial segmentation sequence determined by the algorithm.

In addition, this variable is used to maintain a fast disjoint set-structure in order to reduce the running time of the color-based segmentation algorithm. The variable 'CC' is an array having the same dimension as the array of tree-hexagons H, which contains as elements objects of the class Tree with the following associated fields:

(isRoot, parent, compIndex,frontier, surface, color)

The field 'isRoot' is a boolean value specifying if the corresponding tree-hexagon index is the root of a tree representing a component, and the field 'parent' represents the index of the tree-hexagon which is the parent of the current tree-hexagon.

The rest of fields are used only if the field isRoot is true. The field compIndex is the index of the associated component. The field surface is a list of indices of the tree-hexagons belonging to the associated component, while the field frontier is a list of indices of the tree-hexagons belonging to the frontier of the associated component. The field color is the mean color of the tree- hexagon colors of the associated component.

The procedure EXTRACTFINALCOMPONENTS determines for each determined component C of Comp, the set sa(C) of tree-hexagons belonging to the component, the set sp(C) of tree-hexagons belonging to the frontier, and the dominant color c(C) of the component.

4. Color-based Spatial Segmentation Algorithm

This value is used at the line 19 of Algorithm 2, where the expression $\tau(ti, t_j)$ is given by the relation (20), 't_i' and 't_j' representing the components Cti and Ct j respectively.

Because we use maximum spanning trees instead of minimum spanning trees the list of the edges E(G) is sorted in non-increasing edge weight. The forest of spanning trees is initialized in such a way each element of the forest contains exactly one tree-hexagon.

Algorithm 2 Color-based segmentation
1: procedure CREATECOLORPARTITION(G,H, Bound)
2: Input G = (V,E), H = $\{h_1, \ldots, h_{|V|}\}$
3: Output Bound
4: $\tau \leftarrow$DETERMINETHRESHOLD(G)
5: Bound $\leftarrow$hi ⊲ Initialize Bound
6: for all i$\leftarrow$1, |V| do
7: MAKESET(hi) ⊲ Initialize the disjoint set data structures
8: end for
9: ⊲ At this point l $\leftarrow$0
10: ⊲ and S0 $\leftarrow$$\{\{h_1\}, \ldots, \{h_{|V|}\}\}$
11: SORT(E,Eπ)
12: ⊲ Eπ= ($e_{\pi 1}$, . . ., $e_{\pi|E|}$) is the sorting of E
13: ⊲ in order of non-increasing weight
14: for all k$\leftarrow$1, |E| do
15: ⊲ Let eπk = (h_i ,h_j) be the current edge in Eπ
16: ti $\leftarrow$FINDSET(hi)
17: t j $\leftarrow$FINDSET(hj)
18: if ti $\neq$t j then
19: if w(hi ,hj) $\leq$ INTVAR(ti, t j)+τ(ti, t j) then
20: UNION(ti, t j,w(hi,hj))
21: ⊲ l $\leftarrow$l+1
22: ⊲ Sl $\leftarrow$Sl$-$1$-$$\{\{Cti\}$,$\{$Ct j $\}\}\cup\{$Cti $\cup$Ct j $\}$
23: else

24: Add the edge (hi ,hj) the the list Bound
25: ⊲ bound(S_l)←bound(S_l−1)∪{(h_i,h_j)}
26: end if
27: else
28: ⊲ Do nothing, $t_i \in C_{t\,j}$
29: end if
30: end for
31: end procedure

The expression $\tau(t_i, t_j) = \tau/\min(|C_{ti}|,|C_{j\,j}|)$ at the line 19 of Algorithm 2 is very important at the beginning of the algorithm because initially the components considered contains only one tree-hexagon and in this case

$IntVar(C_{ti}, C_{j\,j})=0$, and $\tau/\min(|C_{ti}|,|C_{j\,j}|) =\tau$. In order to consider an edge (h_i,h_j) to belonging to the non-boundary class of edges and in consequence to merge the components Ct_i and Cj j corresponding to hi and hj respectively, it is necessary that $w(h_i,h_j) < \tau$.

When the components grow and both components C_{ti} and $C_{j\,j}$ contain more than one tree-hexagon, the external variation between C_{ti} and $C_{j\,j}$ decreases, and in this case the decision for merging or non-merging C_{ti} and $C_{j\,j}$ is affected more by their size than by the global threshold 'τ '.

For each segmentation Sl determined by Algorithm 2 and for each connected component C of the corresponding spanning graph G_l there is a unique maximum spanning tree, $F_l(C)$, that maximize the sum of edge weights for this component.

The forest of all maximum spanning trees associated to the segmentation S_l is

$$F_l = \cup_{C \in Sl} F_l(C), \tag{21}$$

and algorithm makes greedy decisions about which edges to add to Fl . Every time when an edge is added to the maximum spanning tree a union of the two partial spanning trees containing the two vertices of the edge is made. In this way the sequence of the edges contained in the forest F_l of spanning trees is implicit determined at the line 14 of Algorithm 2.

Conversely for each spatial tree T from the forest F_l , the set of all vertices of the initial graph contained in the tree T is denoted by Set(T) and it represents the connected component of Sl associated to maximum spanning tree T:

$$T = F_l(Set(T)). \tag{22}$$

The functions MAKESET, FINDSET and UNION used by the segmentation algorithm implement the classical MAKESET, FIND-SET and UNION operations for disjoint set data structures with union by rank and path compression [18]. In addition the function call, UNION(t_i, t_j ,w(h_i,h_j)), performs the following operation, assuming that t_i is the root of the new spanning tree resulted by combining the spanning trees represented by 't_i' and 't_j' :

• determining CC[t_i].surface as the concatenation of the lists CC[t_i].surface and CC[t_j].surface,

• determining CC[t_i].frontier as a list of indices of tree-hexagons belonging to the frontier of the new component $\{C_{ti} \cup C_{tj}\}$,

• determining CC[t_i].color as the value $(n_i*c_i+n_j*c_j)/(n_i+n_j)$, where c_i = CC[t_i].color, and n_i represents the number of elements in the tree CC[t_i].

Let 'n' be of the input the number of the vertices of the input graph G = (V,E) of the color-based algorithm, n = |V|.

The computational complexity of the color-based segmentation algorithm is given by

T (CREATECOLORPARTITION) = O(n*log n)

5. The Threshold Determination Algorithm

Suppose that a sample (y,z) is generated with the density p(y,z|Θ), where y = [y_1, . . . ,y_n]T is observable data, z = [z_1, . . . , zn]^T is hidden or missing data, and Θ represents a parameter vector, and let p(y|Θ) be the density generating the observable sample y. The purpose of the maximum likelihood (ML) estimation is to maximize the log-likelihood L(Θ) from incomplete data. In addition the EM algorithm [21] performs the ML estimation by iteratively maximizing the following objective function, the expectation of log-likelihood from complete L_{comp}(Θ) over the posterior p(z_i|y_i, Θ (t)):

$$Q(\Theta,\Theta(t)) = \Sigma_i \Sigma z_i \; p(z_i|y_i, \Theta(t)) \log p(y_i, z_i|\Theta), \tag{22}$$

Where Θ(t) represents the estimation of Θ after the t^{th} iteration of the algorithm.

We consider a two-dimensional gamma mixture,

$$p(x|\Theta) = g(x) = \Sigma^2_{j=1} \, p(j) \, f(x|\Theta_j), \tag{23}$$

where p(j) represents the a priori probability of the j-th component, and f (x|Θ) is the gamma distribution:

$$f(x|\Theta) = [x^{a-1}/\Gamma(a)b^a] \, e^{-xb} \tag{24}$$

with the parameter Θ = [a,b]^T having two components, 'a' representing the shape parameter, and 'b' representing the scale parameter.

Denoting the a priori values p(j) by α_j , j = 1,2, the entire vector of parameters, Θ can be written as

$$\Theta = [\alpha_1 \; \alpha_2 \; a_1 \; b_1 \; a_2 \; b_2]^T \tag{25}$$

Because we use the histogram of distances g instead of the vector y (the histogram g is the only available information), the Q-objective function will be defined as

$$Q(\Theta,\Theta(t)) = \sum k_{i=1} \sum^2_{j=1} [\, p(j|xi,\Theta(t)) g(xi) \log(\alpha j \, f(x|\theta j))]. \tag{26}$$

The (t +1)-th iteration step of the EM algorithm performs the following two steps:

1. E-step: Compute Q(Θ,Θ(t)) by computing the posterior p(j|x_i, Θ(t)), for each i= 1, . . . ,k, and j =1,2. Denoting the posterior value p(j|x_i, Θ(t)) by w_{ij}(Θ (t)), we have for each i = 1, . . . ,k, and j = 1,2:

$$w_{ij}(\Theta^{(t)}) = \alpha_j f(x_i|\theta^{(t)}_j) / \sum_{s=1}^{2} \alpha_s f(x_i|\theta^{(t)}_s). \quad (27)$$

2. M-step: Compute the (t +1)-th estimate

$$\Theta(t+1) = \text{argmax } Q(\Theta, \Theta(t)). \quad (28)$$

The M-step is equivalent to update the statistical parameters

$\Theta^{(t+1)} = [\alpha^{(t+1)}_1 \alpha^{(t+1)}_2 a^{(t+1)}_1 b^{(t+1)}_1 a^{(t+1)}_2 b^{(t+1)}_2]^T$ as follows, for each $j = 1,2$

$$\alpha^{(t+1)}_j = v_1,$$
$$a^{(t+1)}_j = \psi^{-1}((v_2 - v_1 \log b^{(t)}_j)/v_1),$$
$$b^{(t+1)}_j = v_3/v_1 \, a^{(t)}_j \quad (29)$$

where

$$v_1 = \sum_{i=1}^{k} w_{ij}(\Theta^{(t)}) \, g(x_i),$$

$$v_2 = \sum_{i=1}^{k} w_{ij}(\Theta^{(t)}) \, g(x_i) \log x_i,$$

$$v_3 = \sum_{i=1}^{k} w_{ij}(\Theta(t)) \, g(x_i) \, x_i \quad (30)$$

and $\Psi(x) = \Psi'(x)/\Psi(x)$ is the digamma function.

Since the EM algorithm uses at iteration the parameter estimates from the previous one, such parameters need to be initialized for the first iteration. However, depending on this initialization, the EM algorithm may not converge to the global maximum if the solution space is not convex and the search is stuck in a local maximum of the log-likelihood surface. This problem is known as the local maxima problem.

In order to make the method more practical, it is necessary to avoid such dependence on the initial conditions. In the following we will apply a deterministic annealing procedure on the mixture decomposition. The Deterministic Annealing EM algorithm (DAEM) [22] helps to overcome the problem of the local maxima, reformulating the maximization of the likelihood into the minimization of the free energy, a concept extracted from thermodynamics.

The idea behind such a procedure is to parameterize the objective function defining the hyper-surface that has to be explored, so that for high values of the temperature T = 1/ß, the curves are smooth enough to allow us to find safely the global maximum using the traditional EM algorithm.

In the DAEM framework the free energy is decomposed as follows:

$$-F_\beta(\Theta) = Q_\beta(\Theta, \Theta^{(t)}) - (1/\beta) H_\beta(\Theta, \Theta^{(t)}). \quad (31)$$

In our case of using the two gamma mixture model and the histogram of distances as observable data the $Q_ß$objective function from the relation (31) can be written as

$$Q_\beta(\Theta, \Theta^{(t)}) = \sum_{i=1}^{k} \sum_{j=1}^{2} q(j|x_i, \Theta^{(t)}) \, g(x_i) \log(\alpha_j f(x_i|\theta_j)), \quad (32)$$

where for each i=1, . . . ,k and j=1,2, the posterior $q(j|x_i, \Theta^{(t)})$ (denoted by $w_{ij\beta}(\Theta^{(t)})$ is given by

$$w_{ij\beta}(\Theta^{(t)}) = ([\alpha_j f(x_i|\theta^{(t)}_j)]^\beta) / \sum_{s=1}^{2} [\alpha_s f(x_i|\theta^{(t)}_s)]^\beta. \quad (33)$$

For a given temperature, T = 1/ß, the DAEM algorithm repeats the following two steps until converged:

1. E-step: Compute $Q_{ß}(\Theta,\Theta(t))$ by computing the posterior $w_{i\,j}\Theta(\Theta(t))$ given by relation (33), for each i=1, . . . ,k, and j = 1,2.
2. M-step: Compute the (t +1)-th estimate

$$\Theta\,(t+1) = \text{argmax}\; Q_{ß}(\Theta,\Theta(t)) \tag{34}$$

The statistical parameters Θ (t+1) and Θ (t+1) are updated by using relations (29) and (30), where posteriors $w_{i\,j}$ are replaced by $w_{i\,j}ß$.

The DAEM algorithm differs from the standard EM algorithm by using the posterior calculated in relation (33) instead of in relation (29). By controlling the 'ß' from a small value to one, we obtain the deterministic annealing process of EM [22]. We start with a initial value T = Tmax = 500 and at each iteration 'k' the temperature becomes $T^{(k)} = T_{max} \times \varepsilon^{k}$,, with a decreasing rate ε= 0.95.

At the first step of the DAEM algorithm, parameters are initialized uniformly with values different from zero. The estimates obtained by a iteration 't' of the EM for a 'β'value are used for the first iteration of EM for the next value of 'β'. Algorithm 3 implements the method described above for determining the value of the threshold 'τ'.

Algorithm 3 The threshold determination

1: **function DETERMINETHRESHOLD(G)
2: Input G = (V,E)
3: Output τ
4: Determine the histogram $g = [g_1, \ldots, g_k]^T$ as defined by Eq. (21)
5: T ←Tmax ←500; ε←0.95; Iter ←10
6: ß←*INITIALESTIMATE()
7: ßmax ← 1/Tmax
8: tmax ←⌈log ß max/ log ε⌉
9: for all t ←0, tmax −1 do
10: ß← 1/T
11: Θ (0) ←Θ
12: for all l ←1, Iter do
13: for all i←1, k and j←1,2 do
14: Compute $w_{i\,j}$ ß using Eq. (33)
15: end for
16: for j←1,2 do
17: ◅ Compute $\alpha^{(t+1)}_{j}$, $a^{(t+1)}_{j}$, and $b^{(t+1)}_{j}$ using Eq. (29) and (30)
18: end for
19: end for
20: T ←T ×ε

21: Θ←Θ(t)
22: end for
23: τ← DISTRIBINTERSECTION(Θ)
24: return τ
25: end function

The internal loop implements the E and M steps of the modified EM algorithm, while the exterior loop implements the schedule of the DAEM algorithm. We used a fixed number of iterations for the EM algorithm for two reasons:

(a) we reduce the computational complexity of the entire algorithm, and

(b) the schedule of the DAEM algorithm assure sufficiently good estimation of the vector of parameters Θ.

The function DISTRIBINTERSECTION calculates the threshold 'τ' at the intersection of the two determined Gaussian distributions. The computational complexity of this function is O(k), because it implements a linear searching by traversing in parallel two vectors of dimension 'k', where 'k' is the dimension of the histogram g(x).

The computational complexity of the function INITIALESTIMATE is also O(k).

The computational complexity of the histogram generation is O(m), where 'm' represents the cardinal of the set E, because this operation involves a traversal of the list representing the set of the edges E.

The computational complexity of the DAEM algorithm is given by the running time for estimating the values $w_{i\,j}ß$, α_j , a_j, and b_j , determinate by the relations (33), (29) and (30) respectively:

1. The estimation value for $w_{i\,j}ß$ can be determinate in constant time, because it involves the estimation of the Gamma function, τ(a), from the relation (24), and this estimation can be done in constant time [23].

2. The running time for estimating α_j and b_j is $t_1 = O(k)$, as resulted by the relations (29) and (30).

3. The running time for estimating aj is also $t_2 = O(k)$, as resulted by the relations (29) and (30), and by the fact that the trigamma function,Ψ−1(x), can be estimated by using a Newton method in maximum 5 iterations with an approximation of 15 digits [23].

It follows that the running time for an iteration of the inner loop of the Algorithm 3 related to the EM algorithm is $t_3 = O(k)$, and the running time for a iteration of the external loop is also $t_4 = O(k)$, because Iter is a constant. In conclusion the running time for the DAEM algorithm is $t_5 = O(k)$, because tmax, the length of the sequence of temperature values associated to the schedule of the DAEM algorithm, is also a constant in the algorithm.

We conclude that the computational complexity of Algorithm 3 is

$$T\,(\text{DETERMINETHRESHOLD}) = O(m), \tag{35}$$

because the number of the bins of the histogram is less than the number of the edges of the graph, $k < m$.

6. Syntactic-Based Region Algorithm

The syntactic-based region model uses some geometric properties of regions together with color information. We use a subset of syntactic features advocated [24] including homogeneity, compactness and regularity.

The region model contains the area of the region and the region boundary. As presented in the previous Subsection, for each region C the segmentation algorithm determines the set sa(C) containing the tree-hexagons forming the region, and the set sp(C) containing the tree-hexagons located at the boundary of the region. Because for each tree-hexagon 'H' we determine its dominant color c(h) and its pseudo-gravity center g(h), for each region C the following information can be further determined:

- the mean color of the region, c(C), the area of the region, a(C), and the length of the contour of the region, p(C). In addition, for each pair of regions, C_i and C_j , the length $p(C_i,C_j)$ of the common boundary between these region can be determined.

In order to reduce the time complexity of the segmentation algorithm we estimate the area a(C) and the perimeter p(C) of a region C in function of the length of the sets sa(C) and sp(C) respectively. Assuming that the distance between two neighboring voxels situated on axis Ox, Oy or Oz has the value 1, the area of a tree-hexagon is 12 and thus the area of a region C is given by the following relation:

$$a(C) = 12 \times |sa(C)|, \tag{36}$$

where |sa(C)| represents the cardinal of the set sa(C).

In order to determine a good final segmentation and to discover the salient objects from the input image, the syntactic based sequence of segmentations, S_f , as defined by Equation (11), can decomposed into several subsequences, each subsequence being determined by a modified form of the Boruvka's algorithm.

Let $i_1 < i_2 < \ldots < i_x < i_{x+1}$ be a sequence of indices, with $i_1 = t$ and $i_{x+1} = k$, that allows a decomposition of the sequence Sf as follows:

$$S_f = (S_{i1} ,S_{i1+1}, \ldots ,S_{i2-1},S_{i2} , S_{i2+1},S_{i2+2}, \ldots ,S_{i3} , \ldots .S_{ix+1},S_{ix+2}, \ldots ,S_{ix+1}) \tag{37}$$

As presented in Algorithm 4 the procedure CREATESYNTACTICPARTITION implements the syntactic based segmentation, while the function GENERATEPARTITION is used to generate the subsequences of segmentations, S_{f1} , . . . ,S_{fx} , each subsequence of the form,

$$S_{f\,j} = (S_{i\,j} ,S_{i\,j+1}, \ldots ,S_{i\,j+1-1},S_{i\,j+1}), \tag{38}$$

being determined by the function GENERATEPARTITION at the j-th call. The last segmentation of the subsequence Sf j generate by GENERATEPARTITION is also the input sequence of the (j+1)-th call of GENERATEPARTITION. The first input segmentation Si1 is the final segmentation St of the color based segmentation algorithm. The function DETERMINEWEIGHTS determines the set A of weights as defined by following relation.

The construction of A is realized as following:

1. Let SB = $[b_1,b_2,b_3,b_4]$ be the sequence contained the same elements as the set B in non-decreasing order. For this reasoning we choose another set of weight values, which is related to the initial set B;

2. Let ‘r’ be the lowest common divisor of the numbers ($b_2 - b_1$), ($b_3 - b_2$), and ($b_4 - b_3$),
3. Let $s = (b_4 - b_1)/r$,
4. The set of weights that we use are:

$$A = \{a_0, a_1, \ldots, a_s\} \tag{39}$$

where $a_0 = b_1$, $a_k = b_4$, $a_i = a_{0+i} * r$, for i = 1, . . . , s, and in addition $b_2, b_3 \in A$.

Algorithm 4 Syntactic-based Segmentation
1: procedure CREATESYNTACTICPARTITION(G,G′, thkg)
2: Input G, G′, thkg
3: Output G′
4: A←DETERMINEWEIGHTS(G′)
5: count ←0
6: repeat
7: G′ ← GENERATEPARTITION(G,G′, thkg,newPart)
8: if newPart then
9: count ←0
10: k←$[a_0\ a_0\ a_0\ a_0]T$
11: end if
12: thkg ←MODIFYWEIGHTS(G′, k)
13: count ←count +1
14: NEXTKVECTOR(k)
15: until count = |A|4
16: end procedure

More formally, the j-th call of the function GENERATEPARTITION, for which the output parameter newPart has the value true, is associated to the non-empty subsequence Sf j of segmentations and it generates a sequence of graphs,

$$G_{i\,j} = (G^{i\,j}{}_{i\,j}, G^{i\,j}{}_{i\,j+1}, \ldots, G^{i\,j}{}_{i\,j+1-1}, G^{i\,j}{}_{i\,j+1}), \tag{40}$$

and a sequence of associated forests of minimum spanning trees,

$$F_{i\,j} = (F^{i\,j}{}_{i\,j}, F^{i\,j}{}_{i\,j+1}, \ldots, F^{i\,j}{}_{i\,j+1-1}, F^{i\,j}{}_{i\,j+1}), \tag{41}$$

such that the last forest is empty, $F^{i\,j}{}_{i\,j+1} = \emptyset$. For each graph $G^{i\,j}{}_l$ from the sequence $G_{i\,j}$, $F^{i\,j}{}_l$ represents the forest of minimum spanning trees of $G^{i\,j}{}_l$, and $G^{i\,j}{}_{l+1}$ is the contraction of $G^{i\,j}{}_l$ over all the edges that appear in $F^{i\,j}{}_l$, as presented in Algorithm 5.

Because the last graph, $G^{i\,j}{}_{i\,j+1}$, of the sequence $G_{i\,j}$ can not be further contracted the dissimilarity vectors of functions associated to the edge weights, $d(C(v_i),C(v_j))$, are not modified, and thus the edge weights, $w(v_i,v_j)$, as defined by the function GRAPH EXTRACTION.are not modified. In order to restart the process for determining the new subsequence,

$$S_{f\,j+1} = (S_{i\,j+1}, S_{i\,j+1+1}, \ldots, S_{i\,j+2}), \tag{42}$$

the first graph, $G^{i\,j+1}{}_{i\,j+1}$ of the sequence $G_{i\,j+1}$ differs from the last graph, $G^{i\,j}{}_{i\,j+1}$, of the sequence G_{ij} by modifying only the weighted vector $k \in K$. The function MODIFYWEIGHTS of Algorithm 4 realizes this modification and recalculates the new

global weighted threshold. In this case the values for the weighted vector k are sequential determined in the lexicographic order, generated by the procedure NEXTKVECTOR.

This constraint is necessary in order to realize a stopping criterion for the algorithm: the last graph cannot be modified and for all distinct values of the weighted vectors $k \in K$ and thus another partition cannot be determined. Each time when GENERATEPARTITION generates a non-empty sequence of segmentations, the output parameter newPart became true and the first vector of the set K is generated.

When GENERATEPARTITION generates an empty sequence of segmentations, newPart is 'false' and the next vector in lexicographic order is generated by the procedure NEXTKVECTOR.

When sequentially for all distinct weighted vectors $k \in K$ (e.g. $|A|4$ distinct vectors, with the set A specified by the relation (39)) generated in lexicographic order the function GENERATEPARTITION generates a empty sequence of segmentations, the procedure GCREATESYNTACTICPARTITION finishes.

Between the last graph, $G^{i\,j}{}_{i\,j+1}$, of the sequence $G_{i\,j}$ and the first graph, $G^{i\,j+1}{}_{i\,j+1}$ of the sequence $G_{i\,j+1}$, there is a sequence of graphs that differ only by the edge weights,

$$b\,G_{i\,j} = (\,b\,G^{i\,j}{}_1\,,\, b\,G^{i\,j}{}_2\,,\,\ldots\,,\, b\,G^{i\,j}{}_{bni\,j}\,), \tag{43}$$

such that $b\,G^{i\,j}{}_1 = G^{i\,j}{}_{i\,j}$ and $b\,G^{i\,j}{}_{bni\,j} = G^{i\,j+1}{}_{i\,j+1}$. This sequence is obtained when the function GENERATEPARTITION generates an empty sequence of segmentations, with $bni\,j < |A|4$.

As presented in Algorithm 5 the function GENERATEPARTITION generates at the j-th call the sequence of graphs $G_{i\,j}$ defined by Equation (40), and the sequence of forests of minimum spanning trees defined by Equation (41), where:

- the first graph of the sequence $G_{i\,j}$ is the input graph of the function (i.e. the parameter G'),
- the last graph of this sequence is the graph returned by the function.

The function GENERATEPARTITION is a generalized Greedy algorithm for constructing minimum spanning trees, as presented in [25]. At each iteration, 'l', of the function GENERATEPARTITION, the contraction of the tree $G^{i\,j}{}_l$ over all the edges that appear in the minimum spanning tree $F^{ij}{}_l$ is performed by the function CONTRACTGRAPH.

Algorithm 5 Generate a new sequence of partitions

```
1: function GENERATEPARTITION(G,G′, thkg, newPartition)
2: Input G, G′, thkg, G′   ◅ G′ = G^{i j}_{i j} is the input graph
3: Output newPartition
4: newPartition← false   ◅ l ←0
5: repeat
6: k←0
7: for i←1,G′.n do
8: if G′.ad jEdges[i] ≠() then
9: Determine the lightest edge 'e' adjacent to G′.V[i]
```

10: ⊲ Let ei ∈ G′.ad jEdges[i] such that
11: ⊲ e = G′.E[ei] = (vi , vj) is the lightest edge
12: thkl← DETERMINETHL(vi, vj)
13: if e.w ≤ min(thkg, thkl) then
14: ⊲ Determination of the MST $F^{i\,j}_{i\,j+1}$
15: k ←k+1
16: e.inMST ←true
17: end if
18: end if
19: end for
20: if k > 0 then
21: G′ ←CONTRACTGRAPH(G,G′, thkg)
22: ⊲ Determination of the graph $G' = G^{i\,j}_{i\,j+l+1}$
23: ⊲ l ←l+1
24: newPartition←true
25: end if
26: until k = 0
27: return G′ ⊲ $G' = G^{i\,j}_{i\,j+1}$ is the output graph
28: end function

The function DETERMINETHL returns the local weighted threshold thhl associated to the components C_{vi} and C_{vj} , as presented in the following relations:

- the local weighted threshold associated with the weighted vector k ∈K and with the adjacent components C′ and C″ of the segmentation S_l is denoted by thkl(C′,C″) and it is determined by considering the average of dissimilarity functions for anly adjacent components with C′ and C″ from the segmentation S_l ,

$$thkl(C',C'') = bkT_{\,l}(C',C''), \tag{44}$$

where the components of the vector l(C′,C″) are determined, for i = 1,2,3,4, as follows:

$$l_i(C',C'') = [\Sigma_p(C',C'',C_a,C_b)edi(C',C'')]/[\Sigma_p(C',C'',C_a,C_b)\ 1]\ , \tag{45}$$

where the predicate p(C′,C″,Ca,Cb) is defined as

$$p(C',C'',C_a,C_b) = ((C_a,C_b) \in S_l) \wedge (ad\ j(C',C_a) = true) \wedge (ad\ j(C'',C_b) = true). \tag{46}$$

The function implementing the contraction procedure, CONTRACTGRAPH, is similarly to the function EXTRACTGRAPH with the following differences:

- It detects the connected components specified by the edges marked as MST in the GENERATEPARTITION, and assigns to each vertex of the new generated graph the component it belongs to. The function DETERMINECOMPONENTS implements a Depth-First-Search traversal method on the input graph, in order to enumerate the connected components.
- As in the color-based segmentation algorithm (see Algorithm 2), for each edge from the minimum spanning tree a union of the two partial spanning trees containing the two vertices of the edge is made by using the procedure UNION. In this way it is

realized a reunion of the components associated to the vertices from each connected component of the input graph:

$$C(v) = \cup_{u \in Set(Tv)} C(u), \tag{47}$$

where Tv denotes the minimum spanning tree from the input graph associated to the connected component that represents the new created vertex in the output graph, and Set(Tv) represents the connected component associated to Tv.

- The weights of the new created edges and also the weighted threshold of the output graph use a weighted vector $k \in K$ such that its components have a value random chosen from the set $A = \{a_0, a_1, \ldots, a_s\}$ by using the procedure ALEAKCHOOSE. This is an important aspect of the syntactic based segmentation algorithm and in this way the distribution of the weights of the four dissimilarity functions tends to became uniform.

The sequence F f of forests of minimum spanning trees as defined by Equation (12) can be decomposed as the sequence S_f of segmentations as follows:

$$F_f = (F_{i'1}, F_{i'1+1}, \ldots, F_{i'2-1}, F_{i'2}, F_{i'2+1}, \ldots, F_{i'3-1}, \ldots F_{i'x}, F_{i'x+1}, \ldots, F_{i'x+1-1}). \tag{48}$$

Because the graph $G^{i\,j}{}_{i\,j+l}$ and its corresponding minimum spanning tree $F^{i\,j}{}_{i\,j+l}$, for $j = 1, \ldots, x$ and $l = 0, \ldots, i\,j+1 - i\,j - 1$, share the same set of vertices, from algorithm of graph contraction one can see that each subsequence of forests determined at the j^{th} call of the function GENERATEPARTITION,

$$F_{fj} = (F_{i'j}, F_{i'j+1}, \ldots, F_{i'j+1-1}, F_{i'j+1}), \tag{49}$$

can determined for each $l = 0, \ldots, i\,j+1-i\,j-1$ as follows:

$$E_{i\,j+l+1} = E'_{i\,j+l} \cup_{e \in Fi\,ji\,j+l} Orig(e), \tag{50}$$

where E'_u represents the set of the edges associated to the forest $F'_u = (V', E'_u)$, and Orig(e) represents the edge from the initial graph G corresponding to the edge 'e' from the current graph $G^{i\,j}{}_{i\,j+1}$.

The call of the procedure UNION at the line 22 of graph contraction allows the determination of the sequence of the segmentations S f as defined by Boruvka's algorithm.

$$S_{i\,j+l+1} = \{Set(T) \mid T \in F_{i\,j+l+1}\} = \{C(v) \mid v \in G^{i\,j}{}_{i\,j+l+1}\}, \tag{51}$$

for each $j = 1, \ldots, x$ and $l = 0, \ldots, i\,j+1-i\,j-1$. This relation specifies the fact that there is a bijective mapping between the components from the segmentations Si j+l+1 (or equivalently between the trees from the forests Fi j+l+1) and the vertices of the contracted graphs Gi ji j+l+1.

At j-th call of the function GENERATEPARTITION, each call of the function CONTRACTGRAPH generates a new segmentation, $S_{i\,j+l+1}$, with $l = 0, \ldots, i\,j+1 - i\,j - 1$, as defined by relation (51), which tends to merge the components of the previous segmentation until regions closer to salient objects are detected.

Algorithm 6 Graph contraction

1: function CONTRACTGRAPH(G,G′, thkg)
2: Input G, $G' \lhd G' = G^{i\,j}{}_{i\,j+1}$ is the input graph
3: Output thkg
4: $n'' \leftarrow$ DETERMINECOMPONENTS(G′, cIndex)

5: ⊲ Determine connected components of G′
6: ⊲ Let n″ the number of connected components
7: ⊲ Assign to each component an index in the array cIndex
8: G″ ←CREATEGRAPH(n″, cIndex)
9: ⊲ Create a new graph with one vertex for each
10: ⊲ connected component in G′, i.e., G″.n = n″
11: Initialize two arrays of bins, B′ and B″, of dimension n″
12: for i←1,G′.m do ⊲ Let G′.E[i] = e = (v_i , v_j)
13: cj ←G′.V[v_j].comp
14: Add i to the bin B′[c_j]
15: if e.inMST then
16: ei0 ←e.origEdge
17: (hi,hj)←(G.E[e_{i0}].v_i,G.E[e_{i0}].v_{ji})
18: ⊲ (h_i ,h_j) is the original edge from G
19: ⊲ corresponding to the current edge (v_i , v_j)
20: (t_i, t_j)←(FINDSET(h_i,CC), FINDSET(h_j ,CC))
21: if t_i 6= t_j then
22: UNION(t_i, t_j, e.w,CC)
23: ⊲ Determination of the MST $F_{i\,j+l+1}$
24: ⊲ and of the segmentation $S_{i\,j+l+1}$:
25: ⊲ $F_{i\,j+l+1}$ ←$F_{i\,j+l}$∪{Orig(e)},
26: ⊲ $S_{i\,j+l+1}$ ←$S_{i\,j+l}$ −{{Ct_i},{Ct_j }}∪
27: ⊲∪{Ct_i ∪Ct_j }
28: end if
29: end if
30: end for
31: for i←1,n″ do
32: for all ei ∈ B′[i] do ⊲ Let (vi, vj) = G′.E[ei]
33: ci ←G′.V[v_i].comp
34: Add ei to the bin B″[c_i]
35: end for
36: end for
37: ALEAKCHOOSE(k)
38: for i←1,n″ do
39: if B″[i] 6= hi then
40: Determine the lightest edge from the bin B″[i]
41: ⊲ Let ei ∈ B″[i] such that
42: ⊲ G′.E[ei] = (v_i , v_j) is the lightest edge
43: e_{i0} ←G′.E[ei].origEdge
44: (h_i,h_j)←(G.E[e_{i0}].v_i,G.E[e_{i0}].v_{ji})
45: (t_i, t_j)←(FINDSET(h_i,CC), FINDSET(h_j ,CC))
46: dist ←COLORDIST(t_i, t_j ,CC)
47: w← WEIGHT(dist, t_i, t_j,CC, k)

48: h_{ci}, c_{ji}←hG′.V[v_i].comp,G′.V[v_j].compi 49: ADDEDGE(G″, c_i, c_j ,w, e_{i0})
50: end if
51: end for
52: thkg ←DETERMINETHG(G″, k)
53: return G″ ⊲ G″ = $G^{i\,j}{}_{i\,j+l+1}$ is the output graph
54: end function

7. Conclusions

Based on number of the tree-edges of the input spatial graph G = (V,E) of the color-based and syntactic-based algorithms, and the number of the vertices of input spatial graph we say that the time of all presented algorithms are linear. We can use the graph facilities and their related algorithms and computational complexity can be viewed as slow as the fundamental graph algorithms. We have presented a unified framework for spatial image segmentation and threshold extraction algorithms that use a virtual tree-hexagonal structure defined on the set of the image voxels. The proposed spatial graph-based segmentation method is divided into two different steps: (a) a pre-segmentation step that produces a maximum spatial spanning tree of the connected components of the tree-triangular grid spatial graph constructed on the tree-hexagonal structure of the spatial input image, and (b) the final spatial segmentation step that produces a minimum spatial spanning tree of the connected components, representing the visual objects, by using dynamic weights based on the geometric features of the regions.

The problem of all segmentation methods is a well-studied one in literature and there are a wide variety of approaches that are used. Different approaches are suited to different types of input images and the quality of output of a particular algorithm is difficult to measure quantitatively due to the fact that there may be many ‘correct’ segmentation method for a single image. Here, a graph-based theoretic framework is considered by modeling image segmentation as a graph partitioning and optimization problem using input spatial graph.

References

[1] P. K. Nathan Silberman, D.Hoiem, R. Fergus (2012). Indoor segmentation and support inference from RGBD images, in ECCV.
[2] C. All`ene, J.-Y. Audibert, M. Couprie, and R. Keriven (2010). Some links between extremum spanning forests, watersheds and min-cuts. Image and Vision Computing, 28(10), pp.1460–1471,.
[3] P. Arbelaez, Pont-Tuset, J., Barron, J., Marqués, F., and Malik, J., (2014) Multiscale Combinatorial Grouping, in Computer Vision and Pattern Recognition (CVPR).
[4] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In Proc. of IEEE Computer Vision and Pattern Recognition (CVPR 2010), 2010.
[5]] Urquhar, R. (1982). Graph theoretical clustering based on limited neighborhood sets. Pattern Recognition, 15(3), pp. 173–187.
[6] Felzenszwalb, P., Huttenlocher, W. (2004). Efficient graph-based image segmentation. International Journal of Computer Vision, 59(2), pp.167–181.

[7] Guigues, L., Herve, L., Cocquerez, L.P. (2003). The hierarchy of the cocoons of a graph and its application to image segmentation. Pattern Recognition Letters, 24(8), pp.1059–1066.
[8] Gdalyahu, Y., Weinshall, D., Werman, M. (2001). Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(10), pp.1053–1074.
[9] Shi, J., Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), pp. 885–905.
[10] Jermyn, I., Ishikawa, H. (2001). Globally optimal regions and boundaries as minimum ratio weight cycles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(8), 1075–1088
[11] Cooper, M. (1998). The tractibility of segmentation and scene analysis. International Journal of Computer Vision, 30(1), pp. 27–42.
[12] Malik, J., Belongie, S., Leung, T., Shi, J. (2001). Contour and texture analysis for image segmentation. International Journal of Computer Vision, 43(1), pp. 7–27.
[13] Comaniciu, D., Meer, P. (2002b). Robust analysis of feature spaces: color image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), pp. 603–619.
[14] Comaniciu, D., Meer, P. (1999). Mean shift analysis and applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, pp. 1197–1203.
[15] Brezovan, M., Burdescu, D., Ganea, E., Stanescu, L. (2010). An Adaptive Method for Efficient Detection of Salient Visual Object from Color Images. In Proceedings of the 20th International Conference on Pattern Recognition, Istambul, Turkey, pp. 2346–2349.
[16] Burdescu, D., Brezovan, M., Ganea, E., Stanescu, L. (2009). A new method for segmentation of images represented in a HSV color space. Lecture Notes in Computer Science, 5807, 606–616
[17] Stanescu L., Burdescu, D., Brezovan, M., Mihai, CR. G., (2011), Creating New Medical Ontologies for Image Annotation, Springer-Verlag New York Inc. ISBN 13: 9781461419082, ISBN 10: 1461419085
[18] Cormen, T., Leiserson, C., Rivest, R. (1990). Introduction to algorithms. Cambridge, MA: MIT Press.
[19] Donoser, M., Bischof, H. (2007). ROI-SEG: Unsupervised Color Segmentation by Combining Differently Focused Sub Results. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, SUA, pp. 1–8.
[20] Gijsenij, A., Gevers, T., Lucassen, M. (2008). A perceptual comparison of distance measures for color constancy algorithms. In Proceedings of the European Conference on Computer Vision, Marseille, France , pp. 208–221.
[21] McLachlan, G.J., Krishnan, T. (1997). The EM algorithm and extensions. Springer-Verlag.
[22] Ueda, N., Nakano, R. (1998). Deterministic annealing EM algorithm. Neural Networks, 11(2), 271–281.
[23] Abramowitz, M., Stegun, I.A. (1964). *Handbook of Mathematical Functions*. New York: Dover Publications.
[24] Bennstrom, C., Casas, J. (2004). Binary-partition-tree creation using a quasi-inclusion criterion. In Proceedings of the Eighth International Conference on Information Visualization, London, UK (pp. 259–294).
[25] Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E. (1986). Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica, 6, 109–122.