Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 659–668
DOI: 10.15439/2014F174
ACSIS, Vol. 2

Efficient Volumetric Segmentation Method

Dumitru Dan Burdescu, Liana Stanescu, Marius Brezovan, Cosmin Stoica Spahiu
Computers and Information Technology Department
Faculty of Automation, Computers and Electronics, University of Craiova,
Craiova, Dolj, Romania
Address: Bvd. Decebal, Nr. 107, 200440, Tel./Fax: +40-251 438198
dburdescu@yahoo.com; lia_stanescu@yahoo.com

***Abstract* -- In this paper we extend our previous work for planar images by adding a new step in the volumetric segmentation algorithm that allows us to determine regions closer to it. There are huge of papers for planar images and segmentation methods and most of them are graph-based for planar images and very few papers for volumetric segmentation methods. 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 volumetric 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. 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. A number of approaches to segmentation are based on finding compact regions in some feature space. A recent technique using feature space regions first transforms the data by smoothing it in a way that preserves boundaries between regions. The key to the whole own algorithm of volumetric segmentation is the honeycomb cells. The pre-processing module is used mainly to blur the initial RGB spatial image in order to reduce the image store and to make algorithms to be efficient. Then the volumetric 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 volumetric grid graph having tree-hexagons as cells of vertices. Early graph-based methods use fixed thresholds and local measures in finding a volumetric segmentation.**

Index terms- Volumetric Segmentation; Graph-based segmentation; Color segmentation; Syntactic segmentation

I. Introduction and Related Works

THERE is a wide range of computational vision problems for planar images that could use of segmented images. The problem of partitioning images into homogenous regions or semantic entities is a basic problem for identifying relevant objects. Higher-level problems such as object recognition and image indexing can also make use of segmentation results in matching, to address problems such as figure-ground separation and recognition by parts. In both intermediate level and higher-level vision problems, contour detection of objects in real images is a fundamental problem. However the problems of planar image segmentation and grouping remain great challenges for computer vision. 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. We develop a visual feature-based method which uses a virtual spatial graph constructed on cells of prisms with tree-hexagonal structure containing half of the image voxels in order to determine a forest of spanning trees for connected component representing visual objects. Thus the spatial image segmentation is treated as a spatial graph partitioning problem. In addition our spatial segmentation algorithm produces good results from both from the perspective perceptual grouping and from the perspective of determining homogeneous in the input images. Early graph-based methods use fixed thresholds and local measures in finding a spatial segmentation.

In [1] one determined the normalized weight of an edge by using the smallest weight incident on the vertices touching that edge. Other methods for planar images [2], [3] 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 graph are designed to minimize the similarity between pixels that are being split [4] [5]. The normalized cut criterion [5] takes into consideration self similarity of regions. An alternative to the graph cut approach is to look for cycles in a graph embedded in the image plane. In [6] 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 [7] use a measure of uniformity of a region. In contrast [2] and [3] 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 [8]. A number of approaches to segmentation are based on finding compact regions in some feature space [9]. A recent technique using feature space regions [10] first transforms the data by smoothing it in a way that preserves boundaries between regions. Our previous works [11] and [12] are related to the works in [2] and [3] in the sense of pair-wise comparison of region similarity. In these papers we extend our previous work by adding a new step in the spatial segmentation algorithm that allows us to determine regions closer to it.

II. Constructing a Virtual Tree-Hexagonal Structure

The pre-processing module is used mainly to blur the initial RGB spatial image in order to reduce the image noise [13] and to apply the spatial segmentation algorithm. 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 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 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 this grid is the fact that the visualization stage uses directly the voxels of the digitized image. We do not use a 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 [14]. As a consequence we construct a virtual tree-hexagonal structure over the grid voxels of an input image, as presented in Figure 1. This virtual tree-hexagonal grid is not a tree-hexagonal lattice because the constructed hexagons are not regular.

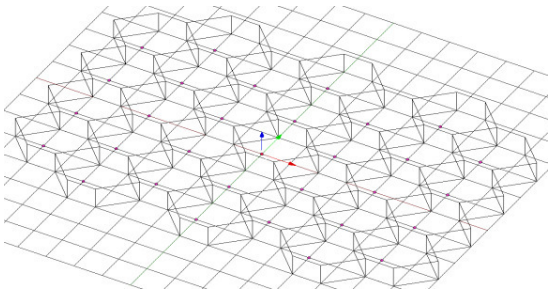

Fig. 1. Virtual Tree-Hexagonal structure constructed on the image voxels

Let I be a spatial initial image having the dimension h×w× 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

$$\begin{aligned} h' &= h-(h-1) \bmod 2, \\ w' &= w-w \bmod 4, \\ z' &= z- z \bmod 4. \end{aligned} \tag{1}$$

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 spatial 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 tree-hexagons as presented in Figure 1.

Let h× w× 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 linearized function, $ip_{h,w,z}(l,c,d) = (l-1)w+c+d$, 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' hs', 'ws' and 'zs' 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 hs, ws and zs are given by the following relation:

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

In this case the following relation holds:

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

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

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

where:

$l = (2+ 4(k-1)/w$ if $h < w$,

$l = 2+ 4(k-1)/w +tw$ if $h \geqslant w$, and $h = tw+h'$, (5)

$c = 4(k-1)+2l-(l-2)w$, (6)

$d = 4(k-1)+2l-(l-2)w$. (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 'fh' 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), (7) allow to up-date the voxels of the spatial initial spatial 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.

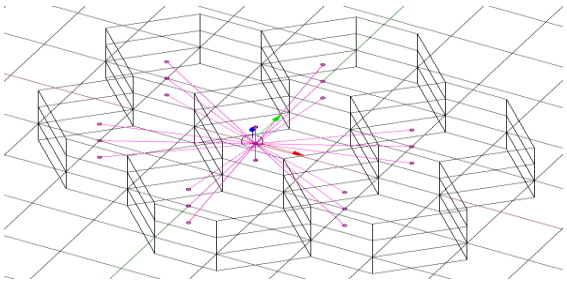

Fig.2. Triangular grid graph constructed on the pseudo-gravity centers of the tree-hexagonal grid

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

• The amount of memory space associated to the 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 c(h) and 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.

III. Volumetric Segmentation Algorithm

Let $V = \{h_1, \ldots, h_{|V|}\}$ be the set of virtual tree-hexagons constructed on the input spatial image voxels as presented in previous section and $G = (V,E)$ be the undirected spatial grid-graph, with E containing pairs of honey-beans cell (tree-hexagons) 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(hi,hj), and it represents the dissimilarity between neighboring elements 'hi' and 'hj' in a some feature space. Components of a spatial 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 [15] we use the notion of segmentation of the set V. 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:

$$cb(C',C'') = \{(hi,hj) \in E \mid hi \in C', hj \in C''\} \quad (8)$$

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:

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

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

We use the notions of segmentation too fine and too coarse as defined in [2] that attempt to formalize the human perception of salient visual objects from an 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 exists 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 spatial 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 virtual tree-hexagonal structure of an image, with $V = \{h1, \ldots, h|V|\}$. A proper segmentation of V, is a partition S of V such that there exists a sequence $[Si, Si+1, \ldots, S\ f-1, S\ f]$ 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 $j = i, \ldots, f-1$,

• segmentation S_j is too fine, for each $j = i, \ldots, f-1$,

• any segmentation S_l such that $S_f \subset S_l$, is too coarse,

• segmentation S^f is neither too coarse nor too fine.

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

Our 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 segmentation a spanning tree is constructed and thus for each segmentation we use an associated spanning forest.

The evidence for a boundary between two components is determined taking into consideration some features in some model of the spatial input 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 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,

$$S_i^f = [S_0,S_1, \ldots, S_{k-1},S_k], \quad (10)$$

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 Si represents the color-based segmentation sequence, and Sf represents the syntactic-based segmentation sequence.

The final segmentation St 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 graph, F0 = (V, Ø), 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, 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. This sequence uses as input a new graph,

$G' = (V', E')$, which is extracted from the last forest, Ft , of the sequence Fi. Each vertex 'v' from the set V' corresponds to a component Cv from the segmentation St (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' , $Ft' = (V', \emptyset)$.

We focus on the definition of a logical predicate that allow us to determine if two neighboring regions represented by two components, Cl′ and Cl'', from a segmentation Sl can be merged into a single component Cl+1 of the segmentation Sl+1. Two components, Cl′ and Cl'' , represent neighboring (adjacent) regions if they have a common boundary:

$$adj(Cl', Cl'') = true \text{ if } cb(Cl', Cl'') \neq \emptyset,$$
$$adj(Cl', Cl'') = false \text{ if } cb(Cl', Cl'') = \emptyset \quad (13)$$

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

$$PED(e,u) = [w_R(R_e - R_u)^2 + w_G(G_e - G_u)^2 + w_B(B_e - B_u)^2]^{1/2} \quad (14)$$

where the weights for the different color channels, wR, wG, and wB verify the condition wR +wG +wB = 1. Based on the theoretical and experimental results on spectral and real world data sets, Gijsenij et al. [16] is concluded that the PED distance with weight-coefficients (wR =0.26, wG = 0.70, wB =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 [2] and [15]. 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 graph G =(V,E), and let $w_{col}(h_i,h_j)$ representing the color dissimilarity between neighboring elements hi and hj, determined as follows:

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

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 (hi,hj) 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:

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

The internal contrast of a component C containing only one tree-hexagon is zero:

$$IntVar(C) = 0, \quad \text{if } |C| = 1.$$

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

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

We 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 [2] 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:

$$IntVar(C',C'') = \max(IntVar(C'), IntVar(C'')), \quad (18)$$

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

$$diffcol(C',C'') = true, \text{ if}$$
$$ExtVar(C',C'') > IntVar(C',C'') + th^{kg}(C',C''),$$

$$diffcol(C',C'') = false, \text{ if}$$
$$ExtVar(C',C'') \leqslant IntVar(C',C'') + th^{kg}(C',C''), \quad (19)$$

with the the adaptive threshold $th^{kg}(C',C'')$ given by

$$th^{kg}(C',C'') = th^{kg} / \min(|C'|, |C''|) , \quad (20)$$

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

The predicate 'diffcol' 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 input 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

$$adj(C',C'') = true \wedge diffcol(C',C'') = false.$$

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

We decided to use the RGB color space because it is efficient and no conversion is required.

Let G= (V,E) be the initial graph constructed on the virtual 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, Si f , as defined by Equation (10), and its associated sequence of forests of spanning trees, Fi 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, Fi, 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 [17].

• The syntactic-based sequence of segmentations, Sf, 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 segmentation procedure is presented in Algorithm 1

Algorithm 1 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, th^{kg})
8: CREATESYNTACTICPARTITION(G,G', th^{kg})
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 th^{kg} is the thresholds.

The color-based segmentation and the syntactic-based 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 triangular grid 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 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 tree in the forest represent in fact a component in each segmentation S in the 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 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.

IV. Color-Based Region Algorithm

Let G = (V,E) be the undirected spatial graph constructed on the tree-hexagonal structure of a spatial image. The proposed color-based segmentation algorithm will produce a proper segmentation of V according to the Definition 1, where the notion of segmentation too fine is given by the Definition 2.

The sequence of segmentations, $(S_0,S_1, \ldots ,S_{t-1},S_t)$, and its associated sequence of growing forests, $(F_0,F_1, \ldots ,F_{t-1},F_t)$, will be iteratively generated, based on a maximum spanning tree construction method. We use a modified form of the Kruskal's algorithm [17] presented in Algorithm 2, where the trees generated at each step represent the connected components of spatial segmentation.

The input parameters of the color-based segmentation algorithm are the initial spatial graph 'G' and the array 'H' of the tree-hexagons from the tree-hexagonal grid. The output parameter is the list 'Bound' of edges representing the boundary of the final spatial segmentation. The global parameter threshold 'th^{kg}' is determinate by using Algorithm 1.

This value is used at the line 19 of Algorithm 2, where the expression th^{kg} (ti, t j) is given by the relation (20), t_i and t_j representing the components Ct_i 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 = {h1, . . .,h|V|}
3: Output Bound
4: th^{kg} ←*DETERMINETHRESHOLD(G)
5: Bound ←hi ◅ Initialize Bound
6: for all i←1, |V| do
7: *MAKESET(hi) ◅ Initialize the disjoint set data structures
8: end for
9: ◅ At this point l ←0
10: ◅ and S0 ←{{h1}, . . . ,{h|V|}}
11: *SORT(E,$E\pi$)
12: ◅ $E\pi$ = ($e_{\pi 1}$, . . ., $e_{\pi |E|}$) is the sorting of E
13: ◅ in order of non-increasing weight
14: **for all** k←1, |E| **do**
15: ◅ Let $e_{\pi k}$ = (h_i ,h_j) be the current edge in $E\pi$
16: t_i ←*FINDSET(h_i)
17: t_j ←*FINDSET(h_j)
18: **if** $t_i \neq t_j$ **then**
19: if w(hi ,hj) ⩽ INTVAR(ti, t j)+ th^{kg} (ti, t j) then
20:* UNION(ti, t j,w(hi,hj))
21: ◅ l ←l+1
22: ◅ Sl ←Sl−1−{{Cti},{Ct j }} ∪ {Cti ∪Ct j }
23: else
24:* Add the edge (hi ,hj) the the list Bound
25: ◅ bound(Sl)←bound(Sl−1) ∪ {(hi,hj)}
26: end if
27: else
28: ◅ Do nothing, ti ∈ Ct j
29: end if
30: end for
31: end procedure

The expression th^{kg} (ti, t j) = th^{kg} /min(|Cti |,|Cj $_j$ |) 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(Cti ,Cj $_j$)=0, and th^{kg} /min(|Cti |,|Cj $_j$ |) = th^{kg}. In order to consider an edge (hi,hj) to belonging to the non-boundary class of edges and in consequence to merge the components Cti and Cj $_j$ corresponding to 'h_i' and 'h_j' respectively, it is necessary that $w(h_i,h_j) < th^{kg}$.

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

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

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

$Fl = \cup C \in Sl\ \ Fl(C)$,

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 Fl of spanning trees is implicit determined at the line 14 of Algorithm 2.

Conversely for each spatial tree T from the forest Fl , 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:

$T = Fl(Set(T))$.

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 [17]. In addition the function call, UNION(ti, t j ,w(hi,hj)), performs the following operation, assuming that ti is the root of the new spanning tree resulted by combining the spanning trees represented by 'ti' and 't j' :

• determining CC[ti].surface as the concatenation of the lists CC[ti].surface and CC[tj].surface,

• determining CC[ti].frontier as a list of indices of tree-hexagons belonging to the frontier of the new component

$\{Cti \cup Ct\,j\}$,

• determining CC[ti].color as the value (ni*ci+nj*cj)/(ni+nj) , where ci = CC[ti].color, and 'ni' represents the number of elements in the tree CC[ti].

V. 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 [18] 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, Ci and Cj , the length $p(Ci,Cj)$ 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:

$$a(C) = 12 \times |sa(C)|, \tag{21}$$

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, Sf , as defined by Equation (11), can decomposed into several subsequences, each subsequence being determined by a modified form of the Boruvka's algorithm.

Let $i1 < i2 < \ldots < ix < ix+1$ be a sequence of indices, with $i1 = t$ and $ix+1 = k$, that allows a decomposition of the sequence Sf as follows:

$$\begin{aligned} Sf = (\ & Si1, Si1+1, \ldots, Si2-1, Si2, \\ & Si2+1, Si2+2, \ldots, Si3, \\ & \ldots \\ & Six+1, Six+2, \ldots, Six+1\ ). \end{aligned} \tag{22}$$

As presented in Algorithm 1 the procedure CREATESYNTACTICPARTITION implements the syntactic based segmentation, while the function GENERATEPARTITION is used to generate the subsequences of segmentations, $Sf1$, . . . ,Sfx , each subsequence of the form,

$$Sf\,j = (Si\,j, Si\,j+1, \ldots, Si\,j+1-1, Si\,j+1\ ), \tag{23}$$

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 (b2− b1), (b3−b2), and (b4−b3),
3. Let s = (b4−b1)/r ,
4. The set of weights that we use are:

$$A = \{a_0, a_1, \ldots, a_s\}, \tag{24}$$

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 3 Syntactic-based Segmentation

1: ****procedure** CREATESYNTACTICPARTITION(G,G', th^kg)
2: **Input** G, G', th^kg
3: **Output** G'
4: $A \leftarrow$*DETERMINEWEIGHTS(G')
5: $count \leftarrow 0$
6: **repeat**
7: $G' \leftarrow$ *GENERATEPARTITION(G,G', th^kg,$newPart$)
8: **if** $newPart$ **then**
9: $count \leftarrow 0$
10: $k \leftarrow [a0\ a0\ a0\ a0]^T$
11: **end if**
12: $th^kg \leftarrow$*MODIFYWEIGHTS(G', k)
13: $count \leftarrow count$ +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,

$$Gi\ j = (G^{i,j}i\ j\ ,G^{i,j}i\ j+1, \ldots ,G^{i,j}i\ j+1-1,G^{i,j}i\ j+1\ ), \quad (25)$$

and a sequence of associated forests of minimum spanning trees,

$$Fi\ j = (F^{i,j}i\ j\ ,F^{i,j}i\ j+1, \ldots ,F^{i,j}i\ j+1-1,F^{i,j}i\ j+1\ ), \quad (26)$$

such that the last forest is empty, $F^{i,j}i\ j+1 = \emptyset$. For each graph $G^{i,j}l$ from the sequence $Gi\ j$, $F^{i,j}l$ 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 4.
Because the last graph, $G^{i,j}i\ j+1$, of the sequence $Gi\ j$ cannot be further contracted the dissimilarity vectors of functions associated to the edge weights, $d(C(vi),C(vj))$, are not modified, and thus the edge weights, $w(vi,vj)$, as defined by the function GRAPH EXTRACTION are not modified. In order to restart the process for determining the new subsequence,

$$Sf\ j+1 = (Si\ j+1\ ,Si\ j+1+1, \ldots ,Si\ j+2\ ), \quad (27)$$

the first graph, $G^{i,j+1}i\ j+1$ of the sequence $Gi\ j+1$ differs from the last graph, $G^{i,j}i\ j+1$, of the sequence $Gi\ j$ by modifying only the weighted vector $\mathbf{k} \in \mathsf{K}$. The function MODIFYWEIGHTS of Algorithm 2 realizes this modification and recalculates the new global weighted threshold. In this case the values for the weighted vector $\mathbf{k}$ 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 $\mathbf{k} \in \mathsf{K}$ 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 $\mathbf{k} \in \mathsf{K}$ (e.g. $|A|^4$ distinct vectors, with the set A specified by the relation (24)) 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 $Gi\ j$ and the first graph, $G^{i,j+1}i\ j+1$ of the sequence $Gi\ j+1$, there is a sequence of graphs that differ only by the edge weights,

$$\mathrm{b}\ Gi\ j = (\ \mathrm{b}\ G^{i,j}1\ ,\ \mathrm{b}\ G^{i,j}2\ , \ldots ,\ \mathrm{b}\ G^{i,j}\mathrm{b}ni\ j\ ), \quad (28)$$

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

As presented in Algorithm 4 the function GENERATEPARTITION generates at the j-th call the sequence of graphs $Gi\ j$ defined by Equation (25), and the sequence of forests of minimum spanning trees defined by Equation (26), where:

- the first graph of the sequence $Gi\ j$ 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 [19]. 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 $F^{i,j}l$ is performed by the function CONTRACTGRAPH.

Algorithm 4 Generate a new sequence of partitions
1: ****function** GENERATEPARTITION(G,G', th^kg, $newPartition$)
2: **Input** G, G', th^kg, G' ◅ $G' = G^{i,j}i\ j$ is the input graph
3: **Output** $newPartition$
4: $newPartition \leftarrow false$ ◅ $l \leftarrow 0$
5: **repeat**
6: $k \leftarrow 0$
7: **for** $i \leftarrow 1, G'.n$ **do**
8: **if** $G'.adjEdges[i] \neq ()$ **then**
9: Determine the lightest edge 'e' adjacent to $G'.V[i]$
10: ◅ Let $ei \in G'.adjEdges[i]$ such that
11: ◅ $e = G'.E[ei] = (vi\ ,\ vj)$ is the lightest edge
12: $th^kl \leftarrow$ *DETERMINETHL(vi, vj)
13: **if** $e.w \leq \min(th^kg, th^kl)$ **then**
14: ◅ Determination of the MST $F^{i,j}i\ j+l$
15: $k \leftarrow k+1$
16: $e.inMST \leftarrow true$
17: **end if**
18: **end if**
19: **end for**
20: **if** $k > 0$ **then**
21: $G' \leftarrow$*CONTRACTGRAPH(G,G', th^kg)
22: ◅ Determination of the graph $G' = G^{i,j}i\ j+l+1$
23: ◅ $l \leftarrow l+1$
24: $newPartition \leftarrow true$
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 th^{hl} associated to the components Cvi and Cvj , as presented in the following relations:

- the local weighted threshold associated with the weighted vector $\mathbf{k} \in \mathsf{K}$ and with the adjacent components C' and C'' of the segmentation Sl is denoted by $\mathbf{th}^{kl}(C',C'')$ and it is determined by considering the average of dissimilarity functions for anly adjacent components with C' and C'' from the segmentation Sl ,

$$\mathbf{th}^{kl}(C',C'') = \mathrm{b}\mathbf{k}T\ \mathbf{l}(C',C''), \quad (29)$$

where the components of the vector $\mathbf{l}(C',C'')$ are determined, for

$i = 1,2,3,4$, as follows:

$$li(C',C'') = [\sum p(C',C'',Ca,Cb)edi(C',C'')]/[\sum p(C',C'',Ca,Cb)\ 1]\ , \quad (30)$$

where the predicate $p(C',C'',Ca,Cb)$ is defined as

$$p(C',C'',Ca,Cb) = ((Ca,Cb) \in Sl) \wedge (ad\,j(C',Ca) = true) \wedge (ad\,j(C'',Cb) = true). \quad (31)$$

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:

$$C(v) = \cup u \in Set(Tv) C(u), \quad (32)$$

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 $\mathbf{k} \in \mathsf{K}$ 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 Sf of segmentations as follows:

$$\begin{aligned} F\,f = (&\ Fi'1, Fi'1+1, \ldots, Fi'2-1,\\ &Fi'2, Fi'2+1, \ldots, Fi'3-1,\\ &\ldots\\ &Fi'x, Fi'x+1, \ldots, Fi'x+1-1\ ). \end{aligned} \quad (33)$$

Because the graph $G^{i\ j}i\ j+l$ and its corresponding minimum spanning tree $F^{ij}i\ j+l$, 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 jth call of the function GENERATEPARTITION,

$$F^{f}j = (Fi'j, Fi'j+1, \ldots, Fi'j+1-1, Fi'j+1\ ), \quad (34)$$

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

$$E'i_{j}+l+1 = E'i_{j}+l \cup e \in F^{ij}i\ j+l\ Orig(e), \quad (35)$$

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 $G^{ij}i\ j+l$.

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.

$$S^{i\,j+l+1} = \{Set(T) \mid T \in Fi\ j+l+1\} = \{C(v) \mid v \in G^{ij}i\ j+l+1\}, \quad (36)$$

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 $G^{ij}i\ j+l+1$.

At *j-th* call of the function GENERATEPARTITION, each call of the function CONTRACTGRAPH generates a new segmentation, $S^{i\,j+l+1}$, with $l = 0, \ldots, i\ j+1 - i\ j - 1$, which tends to merge the components of the previous segmentation until regions closer to salient objects are detected.

Algorithm 5 Graph contraction

1: ****function** CONTRACTGRAPH(G,G', th^kg)
2: **Input** G, G' ⊲ $G' = G^{ij}i\ j+l$ is the input graph
3: **Output** th^kg
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: $G'' \leftarrow$ *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** $i \leftarrow 1, G'.m$ **do** ⊲ Let $G'.E[i] = e = (vi\ ,\ vj)$
13: $cj \leftarrow G'.V[vj\ ].comp$
14: Add i to the bin $B'[cj\ ]$
15: **if** $e.inMST$ **then**
16: $ei0 \leftarrow e.origEdge$
17: $(hi,hj) \leftarrow (G.E[ei0].vi, G.E[ei0].vji)$
18: ⊲ $(hi\ ,hj)$ is the original edge from G
19: ⊲ corresponding to the current edge $(vi\ ,\ vj)$
20: $(ti,\ t\,j) \leftarrow$ (FINDSET(hi,CC), FINDSET(hj ,CC))
21: **if** $ti \neq t\,j$ **then**
22: *UNION(ti, $t\,j$, $e.w$,CC)
23: ⊲ Determination of the MST $Fi\,j+l+1$
24: ⊲ and of the segmentation $Si\,j+l+1$:
25: ⊲ $Fi\,j+l+1 \leftarrow Fi\,j+l \cup \{Orig(e)\}$,
26: ⊲ $Si\,j+l+1 \leftarrow Si\,j+l - \{\{Cti\},\{Ct\,j\ \}\} \cup$
27: ⊲ $\cup \{Cti \cup Ct\,j\ \}$
28: **end if**
29: **end if**
30: **end for**
31: **for** $i \leftarrow 1, n''$ **do**
32: **for all** $ei \in B'[i]$ **do** ⊲ Let $(vi, vj) = G'.E[ei]$
33: $ci \leftarrow G'.V[vi].comp$
34: Add ei to the bin $B''[ci]$
35: **end for**
36: **end for**
37: *ALEAKCHOOSE(k)
38: **for** $i \leftarrow 1, n''$ **do**
39: **if** $B''[i] \neq$ hi **then**

40: Determine the lightest edge from the bin $B'`[i]$
41: ◅ Let $ei \in B'`[i]$ such that
42: ◅ $G'.E[ei] = (vi , vj)$ is the lightest edge
43: $ei0 \leftarrow G'.E[ei].origEdge$
44: $(hi,hj) \leftarrow (G.E[ei0].vi, G.E[ei0].vji)$
45: $(ti, tj) \leftarrow$ (FINDSET(hi,CC), FINDSET(hj ,CC))
46: $dist \leftarrow$ *COLORDIST(ti, tj ,CC)
47: $w \leftarrow$ *WEIGHT($dist$, ti, tj,CC, k)
48: $hci, cji \leftarrow hG'.V[vi].comp, G'.V[vj].compi$
49: *ADDEDGE($G'`$, ci, cj ,w, $ei0$)
50: **end if**
51: **end for**
52: $th^kg \leftarrow$*DETERMINETHG($G'`$, k)
53: **return** $G'`$ ◅ $G'` = G^{ij}ij+l+1$ is the output graph
54: **end function**

VI. Segmentation Results and Quantitative Evaluation

These modalities produce high-resolution voxel based datasets which are in fact data points on a regularly spaced three dimensional grid.

Because sampling data points from the real world is performed slice by slice the existing spatial segmentation techniques are often planar in nature, applying existing planar algorithms to the volume data slice by slice. The results are inferior to native volumetric based solution because these algorithms ignore the interaction between adjacent slices [20], [21], [22], [23].

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 volumetric segmentation remains a difficult problem to tackle due to the complex nature of the topology of volumetric objects, the huge amount of data to be processed and the complexity of the algorithms that scale with the new added dimension.

Martin thesis [24] states that human segmentation can be used as the ground-truth reference in benchmarking segmentations produced by different methods. On the other hand, one may argue that human segmentation is subjective and will produce different segmentations for the same image but in most cases they will differ only in certain regions of local refinement. This idea has been considered in [25], [26] as a method of avoiding penalizing segmentations that are coarser or more refined than others.

In pattern recognition and information retrieval, Precision-Recall method has received a world-wide acceptance and it's considered as a standard measure because it offers good results for relevance [26].

In the general case, precision (or confidence) is defined as the fraction of retrieved cases that are relevant, while recall (or sensitivity) is the fraction of relevant cases that are retrieved. In other words, in the context of classification, the precision for a class is equivalent with the true positives accuracy which is the number of true positives (i.e. the number of cases that are correctly labeled as belonging to that class) divided by the total number of cases labeled as belonging to that class (including false positives, which are cases that were incorrectly labeled as belonging to the class).

$$\text{Precision} = TP/(TP + FP) \quad (37)$$

Also in this context, recall is equivalent with the true positives rate which is defined as the number of true positives divided by the total number of cases that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are cases that were not labeled as belonging to the positive class but should have been).

$$\text{Recall} = TP/(TP + FN) \quad (38)$$

The terms: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) compare the classifier's prediction against apriority external information that is considered as the ground truth (observation). These are synthesized in the contingency table (or confusion matrix), expressed in Table I.

TABLE I. PRECISION-RECALL CONTINGENCY TABLE

	Observation	
Prediction	TP – Correct result	FP – Unexpected positive result
	TN - Correct absence of result	FN – Missing negative result

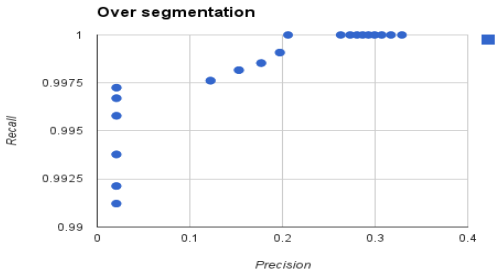

Fig. 3. Experiment Results

As said before, for image segmentation algorithms, Martin [24] proposes a method that outputs Precision-Recall curves as a mean to evaluate segmentation consistency. The curve offers a rich descriptor where both axes are sensitive and intuitive and the inherent trade-off between these two quantities can be easily analyzed.

Recall is defined as the proportion of boundary pixels/voxels in the ground truth that were successfully detected by the automatic segmentation, while precision is the proportion of boundary pixels/voxels in the automatic segmentation that correspond to the true boundary pixels. Precision is in fact a measure of the amount of noise in the classifier's result. The segmentation method used for the experimental results is based on simple hysteresis threshold. All voxels with the density within a specified threshold 'tk^{gh}' will be treated as boundary voxels while the others as empty space [27], [28], [29].

The results are as expected: the over-segmented volume has high recall and low precision (see figure 3), while the under-segmented image has low recall because it fails to find salient features for the volume, and also low precision (since because many boundary pixels remain unmatched).

VII. Conclusion

In this paper we present original and efficient volumetric segmentation methods. The major concept used in graph-based volumetric segmentation method is the concept of homogeneity of regions and thus the edge weights are based on color distance. Our previous works for planar images are related to other works in the sense of pair-wise comparison of region similarity. The key to the whole algorithm of volumetric segmentation is the honeycomb cells.

Here we presented only Color-based Segmentation, Syntactic-based Segmentation and Generate New Sequence of Partitions with Graph Contraction algorithms besides general algorithm of volumetric segmentation due to the entire space. Of course we have many procedures into general algorithm of volumetric segmentation methods. We have presented the original and efficient algorithm of volumetric segmentation methods and honeycomb cells used is the first run in volumetric segmentation algorithm. Then we can use the graph facilities and their related algorithms and computational complexity can be viewed as slow as the fundamental graph algorithms. Our original algorithms for Color-based Segmentation and Syntactic-based Segmentation are linear. Enhancement and generalization of this method is possible in several further directions. First, it could be modified to handle open curves for the purpose of medical diagnosis. Second, research direction is the using of composed shape indexing for both semantic and geometric image reasoning.

VIII. REFERENCES

[1] R. Urquhar, Graph theoretical clustering based on limited neighborhood sets. Pattern Recognition, 15(3), 173–187, 1982.

[2] P. Felzenszwalb, W. Huttenlocher, Efficient graph-based image segmentation. International Journal of Computer Vision, 59(2), 167–181, 2004.

[3] L. Guigues, L. Herve, L.P. Cocquerez, The hierarchy of the cocoons of a graph and its application to image segmentation. Pattern Recognition Letters, 24(8), 1059–1066, 2003.

[4] Y. Gdalyahu, D. Weinshall, M. Werman, 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), 1053–1074, 2001.

[5] J. Shi, J. Malik, Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 885–905, 2000.

[6] I. Jermyn, H. Ishikawa, Globally optimal regions and boundaries as minimum ratio weight cycles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(8), 1075–1088, 2001

[7] M. Cooper, The tractibility of segmentation and scene analysis. International Journal of Computer Vision, 30(1), 27–42, 1998

[8] J. Malik, S. Belongie, T. Leung, J. Shi, Contour and texture analysis for image segmentation. International Journal of Computer Vision, 43(1), 7–27, 2001.

[9] D. Comaniciu, P. Meer, Robust analysis of feature spaces: color image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(5), 603–619, 2002.

[10] D. Comaniciu, P. Meer, Mean shift analysis and applications. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, pp. 1197–1203, 1999.

[11] M. Brezovan, D. Burdescu, E. Ganea, L. Stanescu, 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, 2010.

[12] D. Burdescu, M. Brezovan, E. Ganea, L. Stanescu, A new method for segmentation of images represented in a HSV color space. Lecture Notes in Computer Science, 5807, 606–616, 2009

[13] R. Gonzales, P. Wintz, Digital Image Processing. Reading, MA: Addison-Wesley, 1987.

[14] L. Middleton, J. Sivaswamy, Hexagonal Image Processing; A Practical Approach (Advances in Pattern Recognition). Springer-Verlag, 2005.

[15] L. Stanescu, D. Burdescu, M. Brezovan, CR. G. Mihai, Creating New Medical Ontologies for Image Annotation, Springer-Verlag New York Inc. ISBN 13: 9781461419082, ISBN 10: 1461419085", 2011

[16] A. Gijsenij, T. Gevers, M. Lucassen, A perceptual comparison of distance measures for color constancy algorithms, European Conference on Computer Vision, Marseille, France, pp. 208–221, 2008.

[17] T. Cormen, C. Leiserson, R. Rivest, Introduction to algorithms, Cambridge, MA: MIT Press, 1990.

[18] Bennstrom, C., Casas, J., Binary-partition-tree creation using a quasi-inclusion criterion. In Proceedings of the Eighth International Conference on Information Visualization, London, UK, pp. 259–294, 2004.

[19] Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E., Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica, 6, pg. 109–122., 1986

[20] P. Arbelaez, Pont-Tuset, J., Barron, J., Marqués, F., and Malik, J., Multiscale Combinatorial Grouping, in Computer Vision and Pattern Recognition (CVPR), 2014.

[21] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus, Indoor segmentation and support inference from RGBD images, in ECCV, 2012

[22] Abramowitz, M., Stegun, I.A. Handbook of Mathematical Functions. New York: Dover Publications, 1964

[23] R. Huang, V. Pavlovic, and D. N. Metaxas, A tightly coupled region shape framework for 3d, in Medical Image Segmentation, IEEE International Symposium on Biomedical Imaging (ISBI06), 2006.

[24] David Martin. An Empirical Approach to Grouping and Segmentation. PhD thesis, University of California, Berkeley, 2002.

[25] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in In Proceedings of International Conference on Computer Vision, no. 2, pp. 416–432, 2001.

[26] Y. Haxhimusa, A. Ion, and W. Kropatsch, Evaluating graph-based segmentation algorithms, in Proceedings of the 18th Internation Conference on Pattern Recognition, 2006.

[27] D. Powers, Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation, Journal of Machine Learning Technologies, vol. 2, no. 1, pp. 37–63, 2011.

[28] P. Arbelaez, C. Fowlkes, and D. Martin. The Berkeley segmentation dataset and benchmark. Computer Science Department, Berkeley University. [Online]. Available: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[29] F. J. Estrada and A. D. Jepson, "Benchmarking image segmentation algorithms," International Journal of Computer Vision, vol. 85, no. 2, pp. 167–181, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-025