Efficient Graph-Based Volumetric Segmentation

Dumitru Dan Burdescu, Marius Brezovan, Liana Stanescu and Cosmin Stoica Spahiu

Abstract The emergence and increasing importance of digital society increased the role of software applications in smart environments. Associated with these paradigms are a multitude of applications that generate and require analysis of massive volumes of diverse, heterogeneous, complex, and distributed data. The problem of partitioning images into homogenous regions or semantic entities is a basic problem for identifying relevant objects. There is a wide range of computational vision problems for 2D images that could use of segmented images. However the problems of 3D image segmentation and grouping remain great challenges for computer vision. 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. Many approaches aim to create large regions using simple homogeneity criteria based only on color or texture. However, 3D applications for such approaches are limited as they often fail to create meaningful partitions due to the computation complexity. We are introducing new algorithm for spatial segmentation based on Virtual Tree-Hexagonal Structure constructed on the image voxels. Then the paper depicts a Spatial Segmentation Algorithm. Spatial Segmentation Algorithm contains many other algorithms but only Color-based segmentation algorithm is presented based on the limited space of paper. Then the paper describes the Computational Complexity Analysis of the Color-Based Spatial Segmentation Algorithm.

D.D. Burdescu (✉) · M. Brezovan · L. Stanescu · C.S. Spahiu
University of Craiova, Bd. Decebal 107, Craiova, Romania
e-mail: burdescu_dumitru@software.ucv.ro

M. Brezovan
e-mail: brezovan_marius@software.ucv.ro

L. Stanescu
e-mail: stanescu_liana@software.ucv.ro

C.S. Spahiu
e-mail: stoica_cosmin@software.ucv.ro

K. Arai et al. (eds.), *Intelligent Systems in Science and Information 2014*,
Studies in Computational Intelligence 591, DOI 10.1007/978-3-319-14654-6_9

Keywords Spatial segmentation · Graph-based segmentation · Color segmentation

1 Introduction

The emergence and increasing importance of digital society, cyber-physical systems, and semantic, pervasive, and mobile computing are expanding the role of software and applications in smart or intelligence environments. Associated with these paradigms are instruments, sensors, and a multitude of applications that generate and require analysis of massive volumes of diverse, heterogeneous, complex, and distributed data.

The problem of partitioning images into homogenous regions or semantic entities is a basic problem for identifying relevant objects. There is a wide range of computational vision problems for planar images that could use of segmented images. However the problems of volumetric image segmentation and grouping remain great challenges for computer vision. For instance intermediate-level vision problems motion estimation and tracking require determination of salient objects from frames. The major concept used in graph-based volumetric segmentation method is the concept of homogeneity of volumes and thus the edge weights are based on color distance.

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. Many approaches aim to create large regions using simple homogeneity criteria based only on color or texture. However, spatial applications for such approaches are limited as they often fail to create meaningful partitions due to either the complexity of the scene or difficult lighting conditions. 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.

For example, salient objects are defined as visually distinguishable image compounds that can characterize visual properties of corresponding object classes and they have been proposed as an effective middle-level representation of image content. An important approach for salient object detection is segmentation for planar and volumetric images, and developing an accurate image segmentation technique which partitions image into salient visual objects is an important step toward salient object detection. As a consequence we consider that a volumetric segmentation method can detect visual objects from images if it can detect at least the most objects.

We are introducing new method for volumetric segmentation based on Virtual Tree-Hexagonal Structure constructed on the image voxels. We develop a visual feature-based method which uses a spatial graph constructed on cells of prisms with tree-hexagonal structure containing less than half of the image voxels in order to determine a forest of spanning trees for connected component representing visual objects. Thus the volumetric image segmentation is treated as a spatial graph partitioning problem.

We determine the spatial segmentation of a color image in two distinct steps: a pre-segmentation step when only color information is used in order to determine an initial volumetric segmentation, and a syntactic-based segmentation step when we define a predicate for determining the set of nodes of connected components based both on the color distance and geometric properties of volumes representing visual objects.

The novelty of our contribution concerns: (a) the virtual cells of prisms with tree- hexagonal structure used in the unified framework for volumetric image segmentation, (b) the using of maximum spanning trees for determining the set of nodes representing the connected components in the pre-segmentation step, (c) a method to determine the thresholds used both in the pre-segmentation and in the spatial segmentation step, and (d) an automatic stopping criterion used in the volumetric segmentation step.

In addition our volumetric segmentation algorithm produces good results from both from the perspective perceptual grouping, and from the perspective of determining homogeneous in the input images. We refer the term of perceptual grouping as a general expectation for volumetric segmentation algorithms to produce perceptually coherent segmentation of volumes at a level comparable to humans.

Of course into Volumetric Segmentation Method there are many other algorithms but only Color-based segmentation algorithm and Syntactic segmentation algorithm are designed based on the space of paper. Based on number of the tree-edges of the input spatial graph G = (V, E) of the color-based algorithm, and the number of the vertices of input graph we say and prove that the time of Volumetric Segmentation Algorithm is linear.

Our previous works for digital 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. We present the original and efficient algorithm of volumetric segmentation methods and honeycomb used is the first run into Segmentation Volumetric Method.

1.1 Related Work

In this section we briefly consider some of the related works that are most relevant related to our approach.

Someone determined the normalized weight of an edge by using the smallest weight incident on the vertices touching that edge [1]. Other methods for planar images [2, 3] use 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, 7] the quality of each cycle is normalized in a way that is closely related to the

normalized cuts approaches. Other approaches to digital planar image segmentation consist of splitting and merging regions according to how well each region fulfills some uniformity criterion. Such methods [8] use a measure of uniformity of a region. In contrast [2, 3] use a pair-wise region comparison rather than applying a uniformity criterion to each individual region. Complex organizing phenomena can emerge from simple computation on these local cues [9]. A number of approaches to segmentation are based on finding compact regions in some feature space [10]. Recent techniques for planar digital images using feature space regions [11, 12] first transform the data by smoothing it in a way that preserves boundaries between regions. We use different measures for internal contrast of a connected component and for external contrast between two connected components than the measures used in [13].

Our previous works [11, 14–16] are related to the works in [2, 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.

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. We use maximum spanning tree instead of minimum spanning tree in the pre-segmentation step in order to manage external contrast between connected components.

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 volumes in order to: (a) determine visual objects and their spatial surface, 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 by applying a spatial Gaussian kernel [17]. 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 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 surface

extraction module determines for each segment of the image its boundary. The boundaries of the de determined visual objects are closed surfaces represented by a sequence of adjacent tree-hexagons. At this level a linked list of voxels representing the surface is added to each determined component. The post-processing module (if any) extracts representative information for the above determined visual objects and their surfaces in order to create an efficient index for a semantic image processing system.

A volumetric 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. 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 spatial surface and volumes [18]. As a consequence we construct a virtual tree-hexagonal structure over the voxels of an input image, as presented in Fig. 1. This virtual tree-hexagonal grid is not a tree-hexagonal lattice because the constructed hexagons are not regular.

Let I be an initial volumetric image having the dimension $h \times w \times z$ (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:

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

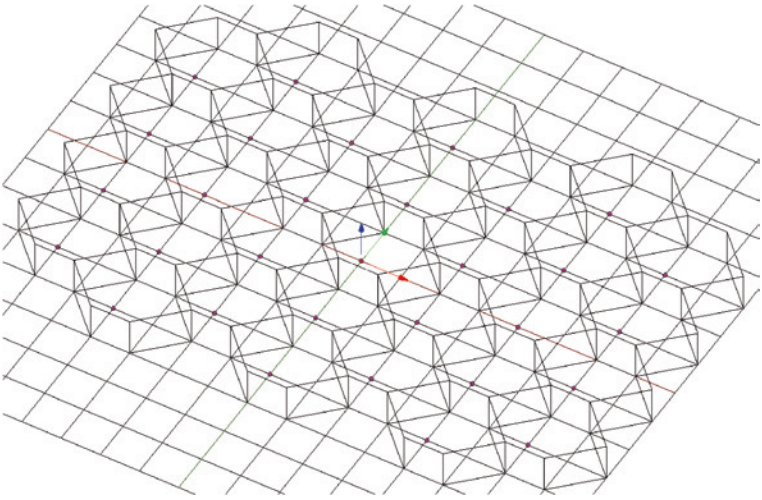

Fig. 1 Virtual tree-hexagonal structure constructed on the image voxels

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 $z \geq 1$, that is a convenient restriction for input images.

Each tree-hexagon from the tree-hexagonal grid contains 16 voxels: such 12 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 Fig. 1.

Let $h \times w \times z$ the three dimension of the initial volumetric image verifying the previous restriction. Given the coordinates $\langle l, c, d \rangle$ of a voxel p from the input volumetric image, we use the linearized function,

$$ip_{h,w,z}(l, c, d) = (l - 1) \times w \times z + (c - 1) \times z + d, \tag{2}$$

in order to determine an unique index for the voxel.

It is easy to verify that the function ip defined by the Eq. 2 is bijective. Its inverse function is given by:

$$ip^{-1}_{h,w,z}(k) = \langle l, c, d \rangle, \tag{3}$$

where:

$$l = k/(w \times z), \tag{4}$$

$$c = (k - (l - 1) \times w \times z)/z, \tag{5}$$

$$d = k - (l - 1) \times w \times z + (c - 1) \times z. \tag{6}$$

Relations 4, 5, and 6 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 ip defined by the relation 2 allow to determine the sequence of the tree-hexagons that is used by the segmentation and surface detection algorithms. After the processing step the Relations 3, 4, 5, and 6 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 spatial grid graph, $G = (V, E)$, where each tree-hexagon H in this structure has a corresponding vertex $v \in V$. The set E of edges is constructed by connecting tree-hexagons that are neighbors in a 8-connected sense. The vertices of this graph correspond to the pseudo-gravity centers of the

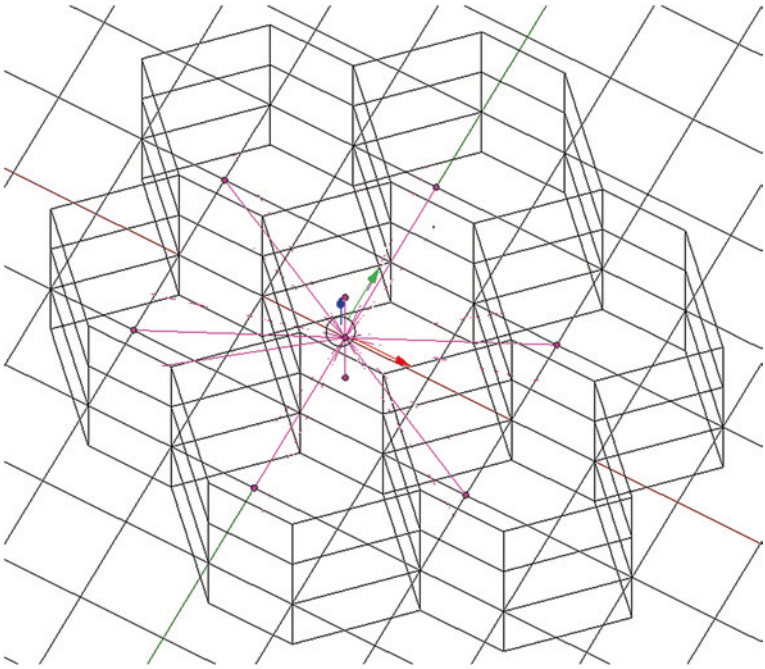

Fig. 2 The grid graph constructed on the pseudo-gravity centers of the tree-hexagonal grid

hexagons from the tree-hexagonal grid and the edges are straight lines connecting the pseudo-gravity centers of the neighboring hexagons, as presented in Fig. 2.

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 $np/8$, and thus the cardinal of both sets V and E is significantly reduced;
- The algorithms for determining the visual objects and their surfaces 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.

3 Volumetric 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 honey-beans cell (tree-hexagons) that are neighbors in a 8-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 volumes 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] for planar images 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:

$$cb(C', C'') = \{(hi, hj) \in E | hi \in C', \quad hj \in C''\}. \tag{7}$$

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:

$$bound(S) = \bigcup_{C', C'' \in S} cb(C', C''). \tag{8}$$

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. A segmentation 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 segmentation method is to determine a proper segmentation, which represent visual objects from a volumetric image.

Definition 1 Let $G = (V, E)$ be the undirected spatial graph constructed on the tree-hexagonal structure of an 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 $\langle S^i, S^{i+1}, \ldots, S^{f-1}, S^f \rangle$ 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.

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 $C^{'}$, $C^{''} \in S^a$ be two components obtained by splitting a component $C \in S^b$. 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 image. When starting, for a certain number of segmentations the only considered feature is the color of the volumes 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 volumes. In addition syntactic features bring supplementary information for merging similar volumes 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,

$$S_{if} = \langle S^0, S^1, \ldots, S^{k-1}, S^k \rangle, \tag{9}$$

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

$$\begin{aligned} S_i &= \langle S^0, S^1, \ldots, S^{t-1}, S^t \rangle, \\ S_f &= \langle S^t, S^{t+1}, \ldots, S^{k-1}, S^k \rangle, \end{aligned} \tag{10}$$

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

$$\begin{aligned} F^i &= \langle F_0, F_1, \ldots, F_{t-1}, F_t \rangle, \\ F^f &= \langle F_{t'}, F_{t'+1}, \ldots, F_{k'-1}, F_{k'} \rangle, \end{aligned} \tag{11}$$

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

The first forest from F^i contains only the vertices of the initial 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 F^i contain maximum spanning trees and they are determined by using a modified version of Kruskal's algorithm [19], 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, 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 volumes 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) volumes if they have a common spatial surface:

$$\begin{aligned} adj(C_{l'}, C_{l''}) &= \text{true}, \quad \text{if } cb(C_{l'}, C_{l''}) \neq \emptyset, \\ adj(C_{l'}, C_{l''}) &= \text{false}, \quad \text{if } cb(C_{l'}, C_{l''}) = \emptyset. \end{aligned} \tag{12}$$

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

$$PED(e, u) = \sqrt{w_R(R_e - R_u)^2 + w_G(G_e - G_u)^2 + w_B(B_e - B_u)^2}, \tag{13}$$

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 volumes 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 spatial surface between two volumes is based on the difference between the internal contrast of volumes and the external contrast between them [2, 16]. Both notions of internal contrast and external contrast between two volumes 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 h_i and h_j, determined as follows:

$$w_{col}(h_i, h_j) = PED(c(h_i), c(h_j)), \text{ if } (h_i, h_j) \in E, \\ w_{col}(h_i, h_j) = \infty, \quad \text{otherwise}, \tag{14}$$

where $PED(e, u)$ represents the perceptual Euclidean distance with weight-coefficients between colors e and u, as defined by Eq. 13, 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 S_l 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:

$$IntVar(C) = \max_{(h_i,h_j)\in C}(w(h_i, h_j)). \tag{15}$$

The internal contrast of a component C containing only one tree-hexagon is zero: $IntVar(C) = 0, \text{ if} |C| = 1$.

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

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

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:

$$IntVar(C', C'') = \max(IntVar(C'), \ IntVar(C'')). \tag{17}$$

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:

$$diff_{col}(C', C'') = true, \quad \text{if } \ ExtVar(C', C'') > IntVar(C', C'') + \tau(C', C'') \\ diff_{col}(C', C'') = false, \quad \text{if } \ ExtVar(C', C'') = IntVar(C', C'') + \tau(C', C''), \tag{18}$$

with the the adaptive threshold $\tau(C', C'')$ is given by

$$\tau(C', C'') = \tau/(\min(|C'|, |C''|), \tag{19}$$

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 $diff_{col}$ 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 volumetric 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 $adj(C', C'') = true \wedge diff_{col}(C', C'') = false$.

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

There are many existing systems for arranging and describing colors, such as RGB, 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 graph constructed on the tree-hexagonal structure of a volumetric image. The proposed segmentation algorithm will produce a proper segmentation of V according to the Definition 1. The sequence of segmentations, S_{if}, as defined by Eq. 9, and its associated sequence of forests of spanning trees, F^{if}, as defined by Eq. 11, will be iteratively generated as follows:

- The color-based sequence of segmentations, S^i, as defined by Eq. 10, and its associated sequence of forests, F^i, as defined by Eq. 11, 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.
- The syntactic-based sequence of segmentations, S^f, as defined by Eq. 10, and its associated sequence of forests, F^f, as defined by Eq. 11, 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 Comp
4:     H ← CREATEHEXAGONALSTRUCTURE(l, c, d, P)
5:     G ← CREATEINITIALGRAPH(l, c, d, P, H)
6:     CREATECOLORPARTITION(G, H)
7:     G′ ← EXTRACTGRAPH(G)
8:     G′ ← CREATESYNTACTICPARTITION(G, G′)
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 surface 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 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.

4 Color-Based Segmentation Algorithm

Let $G = (V, E)$ be the undirected spatial graph constructed on the tree-hexagonal structure of a volumetric input 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, $\langle S^0, S^1, \ldots, S^{t-1}, S^t \rangle$, and its associated sequence of growing forests, $\langle F_0, F_1, \ldots, F_{t-1}, F_t \rangle$, will be iteratively generated, based on a maximum spanning tree construction method. We use a modified form of the Kruskal's algorithm presented in Algorithm 2, where the trees generated at each step represent the connected components of volumetric segmentation.

The input parameters of the color-based segmentation algorithm are the initial 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 τ is determinate by using Algorithm 1. This value is used at the line 18 of Algorithm 2, where the expression $\tau(t_i, t_j)$ is given by the Relation 19, where t_i and t_j representing the components C_{t_i} and C_{t_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.

The expression $\tau(t_i, t_j) = \tau/(\min(|C_{t_i}|, |C_{t_j}|)$ at the line 18 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

$$IntVar(C_{t_i}, C_{t_j}) = 0 \ \wedge \ \tau \ (\min(|C_{t_i}|, |C_{t_j}|) = \tau.$$

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 C_{t_i} and C_{t_j} corresponding to h_i and h_j respectively, it is necessary that $w(h_i, h_j) < \tau$.

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

For each segmentation S_l 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.

Algorithm 2 Color-based segmentation

```
1: procedure createColorPartition(G, H)
2:                                          ▷ G = (V, E), H = {h_1, ..., h_|V|}
3:     τ ← determineThreshold(G)
4:     Bound ← ⟨⟩                                          ▷ Initialize Bound
5:     for all i ← 1, |V| do
6:         makeSet(h_i)                    ▷ Initialize the disjoint set data structures
7:     end for
8:                                                          ▷ At this point l ← 0
9:                                          ▷ and S^0 ← {{h_1}, ..., {h_|V|}}
10:    sort(E, E_π)
11:                                         ▷ E_π = ⟨e_π1, ..., e_π|E|⟩ is the sorting of E
12:                                                  ▷ in order of non-increasing weight
13:    for all k ← 1, |E| do
14:                                      ▷ Let e_πk = (h_i, h_j) be the current edge in E_π
15:        t_i ← findSet(h_i)
16:        t_j ← findSet(h_j)
17:        if t_i ≠ t_j then
18:            if w(h_i, h_j) ≤ intVar(t_i, t_j) + τ(t_i, t_j) then
19:                union(t_i, t_j, w(h_i, h_j))
20:                                                          ▷ l ← l + 1
21:                                  ▷ S^l ← S^(l−1) − {{C_ti}, {C_tj}} ⋃ {C_ti ∪ C_tj}
22:            else
23:                Add the edge (h_i, h_j) the the list Bound
24:                                          ▷ bound(S^l) ← bound(S^(l−1)) ∪ {(h_i, h_j)}
25:            end if
26:        else
27:                                                  ▷ Do nothing, t_i ∈ C_tj
28:        end if
29:    end for
30: end procedure
```

The forest of all maximum spanning trees associated to the segmentation S^l is

$$Fl = \bigcup_{C \in S^l} Fl(C), \tag{20}$$

and algorithm makes greedy decisions about which edges to add to F_l.

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 13 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 S_l associated to maximum spanning tree T:

$$T = F_l(Set(T)). \tag{21}$$

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 [19]. 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 $\{C_{t_i} \cup C_{t_j}\}$,
- 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 $CC[ti]$.

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

The computational complexity of the color-based segmentation algorithm is given by $T(CREATECOLORPARTITION) = O(n * \log(n))$.

5 Syntactic-Based Volumetric Segmentation Algorithm

Let $G = (V, E)$ be the undirected spatial graph constructed on the tree-hexagonal structure of a volumetric image. The global parameter threshold is determinate by using Algorithm 1. In order to determine a good final segmentation and to discover the objects from the input image, the syntactic based sequence of volumetric segmentations, Sf, as defined by Eq. 10, can decomposed into several subsequences, each subsequence being determined by a modified form of the Boruvka's algorithm.

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

$$\begin{aligned} S_f = \langle & S^{i_1}, S^{i_1+1}, \ldots, S^{i_2-1}, S^{i_2}, \\ & S^{i_2+1}, S^{i_2+2}, \ldots, S^{i_3}, \\ & \cdots \\ & S^{i_x+1}, S^{i_x+2}, \ldots, S^{i_{x+1}} \rangle. \end{aligned} \tag{22}$$

As presented in Algorithm 3 the procedure *CREATESYNTACTICPARTITION* implements the syntactic based volumetric segmentation, while the function *GENERATEPARTITION* is used to generate the subsequences of segmentations, $S_{f_1}, \ldots, S_{f_x}$, each subsequence of the form,

$$S_{f_j} = \langle S^{i_j}, S^{i_j+1}, \ldots, S^{i_{j+1}-1}, S^{i_{j+1}} \rangle, \tag{23}$$

being determined by the function *GENERATEPARTITION* at the *jth* call. The last segmentation of the subsequence S_{f_j} generate by *GENERATEPARTITION* is also the input sequence of the $(j + 1)th$ call of *GENERATEPARTITION*. The first input

segmentation S^{i_1} is the final segmentation S^t of the color based segmentation algorithm. The function *DETERMINEWEIGHTS* determines the set A of weights.

Algorithm 3 Syntactic-based segmentation

```
1: procedure createSyntacticPartition(G, G', th_g^k)
2:     Input G, G', th_g^k
3:     Output G'
4:     A ← determineWeights(G')
5:     count ← 0
6:     repeat
7:         G' ← generatePartition(G, G', th_g^k, newPart)
8:         if newPart then
9:             count ← 0
10:            k ← [a_0 a_0 a_0 a_0]^T
11:        end if
12:        th_g^k ← modifyWeights(G', k)
13:        count ← count + 1
14:        nextKVector(k)
15:    until count = |A|^4
16: end procedure
```

More formally, the *jth* call of the function *GENERATEPARTITION*, for which the output parameter *newPart* has the value true, is associated to the non-empty subsequence S_{f_j} of volumetric segmentations and it generates a sequence of graphs,

$$G^{i_j} = \langle G^{i_j}_{i_j}, G^{i_j}_{i_j+1}, \ldots, G^{i_j}_{i_{j+1}-1}, G^{i_j}_{i_{j+1}} \rangle, \tag{24}$$

and a sequence of associated forests of minimum spanning trees,

$$F^{i_j} = \langle F^{i_j}_{i_j}, F^{i_j}_{i_j+1}, \ldots, F^{i_j}_{i_{j+1}-1}, F^{i_j}_{i_{j+1}} \rangle, \tag{25}$$

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 3.

Because the last graph, $G^{i_j}_{i_{j+1}}$, of the sequence G^{i_j} cannot 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 *GRAPHEXTRACTION* are not modified. In order to restart the process for determining the new subsequence,

$$S_{f_{j+1}} = \langle S^{i_{j+1}}, S^{i_{j+1}+1}, \ldots, S^{i_{j+2}} \rangle, \tag{26}$$

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^{i_j} by modifying only the weighted vector $k \in \mathbb{K}$. The function *MODIFYWEIGHTS* of Algorithm 3 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*.

The function *MODIFYWEIGHTS* 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 $k \in \mathbb{K}$ 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 $\mathbb{K}$ 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 $k \in \mathbb{K}$ (e.g. $|A|^4$ distinct vectors, with the set A specified by the Relation 23) generated in lexicographic order the function *GENERATEPARTITION* generates a empty sequence of segmentations, the procedure *CREATESYNTACTICPARTITION* 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,

$$\widehat{G}^{i_j} = \langle \widehat{G}^{i_j}_1, \widehat{G}^{i_j}_2, \ldots, \widehat{G}^{i_j}_{\widehat{n}^i_j} \rangle, \tag{27}$$

such that $\widehat{G}^{i_j}_1 = G^{i_j}_{i_j}$ and $\widehat{G}^{i_j}_{\widehat{n}^i_j} = G^{i_{j+1}}_{i_{j+1}}$. This sequence is obtained when the function *GENERATEPARTITION* generates an empty sequence of segmentations, with $\widehat{n}^i_j \leq |A|^4$.

6 Computational Complexity Analysis of the Color-Based Spatial Segmentation Algorithm

Let $m = |E|$ be the number of the tree-edges of the input spatial graph $G = (V, E)$ of the color-based algorithm, and $n = |V|$ the number of the vertices of G. The running time of the color-based spatial segmentation Algorithm 2 can be factored into four parts:

- The running time required to determinate the threshold τ, denoted by t_0 (line 4), where $t_0 = O(m)$ from relation

$$T(CREATEHEXAGONALSTRUCTURE) = O(n),$$

because $O(n) = O(n_p)$ (the assertion that the number of the resulted tree-hexagons is always less than $np/8$)

- The running time required to initialize the array *CC* at the lines 4–6, denoted by t_1,

$$t_1 = O(n). \tag{28}$$

- The running time required to sort the edges into non-increasing order of weights at the line 9, denoted by t_2.
- The running time of the main part of the algorithm at the lines 12–27, denoted by t_3.

Because $m \leq 3n - 6$ it follows that $O(m) = O(n)$, and thus the running time t_0 is

$$t_0 = O(n). \tag{29}$$

The running time required to sort the edges into non-increasing order of weights can be done in $O(m \log m)$ by using one of several sorting methods (e.g., the Quicksort method). It follows that $O(m \log m) = O(n \log n)$, and thus the running time t_2 is

$$t_2 = O(n \log n). \tag{30}$$

In the following we will discuss the running time t_3. The running time of the function *UNION* at the line 18 can be also factored into two parts:

- the running time for the operations concerning disjoint-set data structures, denoted by t_3^s,
- the running time of the additional operations for determining the values for the fields of the *Tree* objects when merging two components, denoted by $t_3{}^l$.

As a consequence the running time t_3 can be written as

$$t_3 = t_3^s + t_3^l, \tag{31}$$

where t_3^s is the part of t_3 by considering only the operations for disjoint-set data structures in the union function, and t_3^l is the part of t_3 by considering only the additional operations in *UNION*.

Because the function *FINDSET* performs standard operations on disjoint-set data structures and the operation at the line 17 is done in constant time it follows that

$$t_3^s = O(m * \alpha(n)), \tag{32}$$

where $\alpha(n)$ is a very slowly growing function, the inverse of the extremely quickly-growing Ackermann function $A(n, n)$ [19]. Because we have $m = 3n - 6$, and because

$$a(n, n) = O(\log^* n), \tag{33}$$

where

$$\log^* n = \min_{i \geq 0}(\log^{(i)} n, 1). \tag{34}$$

it follows that

$$t_3^s = O(n \log^* n). \tag{35}$$

The running time t_3^l for determining the values for the fields of the *Tree* objects when merging two components is factored as follows:

- the running time for determining the values for the fields *isRoot*, *parent*, *compIndex*, *surface*, and *color*, denoted by t_c, is $t_c = O(m)$, because at each iteration determining these values can be done in constant time,
- the running time for determining the value of the field frontier, denoted by t_f.

In order to determine t_f, let $sp(C^{'})$ and $sp(C^{''})$ be the two lists of tree-hexagons belonging to the frontier of the two components, $C^{'}$ and $C^{''}$, that are merged by the union function, and let $t_f(C)$ be the running time for determining the frontier of the merged component, C. Determining the value of the field frontier associated to the merged component require the traversal of the shortest list from the pair of lists $sp(C^{'})$ and $sp(C^{''})$. Because for every component C the number of the tree-hexagons contained in the region associated to C is less than the number of tree-hexagons from its frontier, the running time $t_f(C)$ verify the following condition:

$$t_f(C) = |C|/2, \tag{36}$$

where $|C|$ represents the number of the tree-hexagons contained in the region associated to C. For the sake of simplicity we assume that $n = 2^k$ for a some integer k. In the worst case the final segmentation S^t contains only one component, $S^t = \langle C^t \rangle$, with $|C^t| = n$, and at each merge operation, the two merged components have the same frontier length

$$\min(|sp(C')|, |sp(C'')|) = |sp(C')| = |sp(C'')|. \tag{37}$$

Thus the worst scenario is in the case when all pairs of merged components have the same frontier length and the same area: first are merged all components containing one hexagon, then are merged all components containing two tree-hexagons, etc. It follows that the running time for determining all the values frontier verify the following relation:

$$t_f = \frac{n}{2} + 2\frac{n}{2^2} + 2^2\frac{n}{2^3} + \cdots + 2^{k-1}\frac{n}{2^k}, \tag{38}$$

where for each term, $2^{i-1}\frac{n}{2^i}$, the factor $\frac{n}{2^i}$ represents the number of the tree-hexagons associated to a component, and 2^{i-1} represents the number of components with the same area. Because

$$\frac{n}{2} + 2\frac{n}{2^2} + 2^2\frac{n}{2^3} + \cdots + 2^{k-1}\frac{n}{2^k} = k\frac{n}{2} = \frac{n \log n}{2 \log 2}$$

it follows that $t_f = \frac{n \log n}{2 \log 2}$, and, in conclusion, $t_f = O(n \log n)$.

Because G is a spatial graph and $m \leq 3n - 6$ it follows that $t_c = O(m) = O(n)$ and thus the running time t_3^l is determined as

$$t_3^l = O(n \log n), \tag{39}$$

and from the relations 31, 35 and 39 it follows that

$$t_3 = O(n \log n). \tag{40}$$

Finally from the relations 28, 30 and 40 it follows the overall running time of Algorithm 2 is

$$T(CREATECOLORPARTITION) = O(n \log n). \tag{41}$$

7 Conclusions

Image segmentation plays a crucial role in effective understanding of digital images, planar or volumetric images. Past few decades saw hundreds of research contributions in this field. However, the research on the existence of general purpose segmentation algorithm that suits for variety of applications is still very much active. Among the many approaches in performing image segmentation, graph based approach is gaining popularity primarily due to its ability in reflecting global image properties. The current research in graph based methods orients towards producing approximate solution (or sub-optimal solution) for such graph matching problem to reduce processing time. Also, use of a priori information that include shape, topology and appearance model of the category of images to be segmented is getting more popularity [21].

The problems of volumetric image segmentation and grouping remain great challenges for computer vision. The problem of all segmentation methods is a well-studied one in literature and there are a wide variety of approaches that are used [6]. 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 [13]. We plan to use a larger image database to confirm the quality of the obtained results, and do the evaluation with additional low level cues as well as different statistical measures.

Here, a graph-based theoretic framework is considered by modeling image segmentation as a graph partitioning and optimization problem using input spatial graph.

We are introducing new algorithm for volumetric segmentation based on Virtual Tree-Hexagonal Structure constructed on the image voxels [22, 23]. 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. The key to the whole algorithms of volumetric segmentation method is the honeycomb cells.

The major concept used in graph-based volumetric segmentation method is the concept of homogeneity of volumes and thus the edge weights are based on color distance. Our original algorithms for Color-based Segmentation and Syntactic-based Segmentation are linear. The proposed volumetric graph-based segmentation method is divided into two different steps: (a) a segmentation step that produces a maximum spatial spanning tree of the connected components of the tree-grid spatial graph constructed on the tree-hexagonal structure of the volumetric input image, and (b) the final volumetric 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 volumes.

Then the paper describes the Computational Complexity Analysis of the Color-Based Spatial Segmentation Algorithm.

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. Incorporation of the fuzzy set theory into graph based frameworks can achieve enhanced segmentation performances.

References

1. Janakiraman, T., Mouli, P.C.: Image segmentation using euler graphs. Int. J. Comput. Commun. Control **5**(3), 314–324 (2010)
2. Felzenszwalb, P., Huttenlocher, W.: Efficient graph-based image segmentation. Int. J. Comput. Vision **59**(2), 167–181 (2004)
3. Guigues, L., Herve, L., Cocquerez, L.P.: The hierarchy of the cocoons of a graph and its application to image segmentation. Pattern Recogn. Lett. **24**(8), 1059–1066 (2003)
4. Gdalyahu, Y., Weinshall, D., Werman, M.: Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. IEEE Trans. Pattern Anal. Mach. Intell. **3**(10), 1053–1074 (2001)
5. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 885–905 (2000)
6. Camilus, K.S., Govindan, V.: A review on graph based segmentation. Int. J. Image Graph. Sig. Proc. **5**, 1–13 (2012)
7. Jermyn, I., Ishikawa, H.: Globally optimal regions and boundaries as minimum ratio weight cycles. IEEE Trans. Pattern Anal. Mach. Intell. **23**(8), 1075–1088 (2001)
8. Cooper, M.: The tractibility of segmentation and scene analysis. Int. J. Comput. Vision **30**(1), 27–42 (1998)
9. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. Int. J. Comput. Vision **43**(1), 7–27 (2001)
10. Comaniciu, D., Meer, P.: Robust analysis of feature spaces: color image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
11. Brezovan, M., Burdescu, D., Ganea, E., Stanescu, L.: An adaptive method for efficient detection of salient visual object from color images. In: Proceedings of the 20th International Conference on Pattern Recognition, pp. 2345–2349. Istanbul, Turkey (2010)

12. Comaniciu, D., Meer, P.: Mean shift analysis and applications. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1197–1203. Madison, Wisconsin (1999)
13. Powers, D.M.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. J. Mach. Learn. Technol. **2**(1), 37–63 (2011)
14. Burdescu, D., Brezovan, M., Ganea, E., Stanescu, L.: A new method for segmentation of images represented in a HSV color space. In: Proceedings of the Advanced Concepts for Intelligent Vision Systems Conference, pp. 606–617 (2009)
15. Stanescu, L., Burdescu, D., Brezovan, M.: A comparative study of some methods for color medical images segmentation. EURASIP Journal on Advances in Signal Processing, 128 (2011)
16. Stanescu, L., Burdescu, D., Brezovan, M., Mihai, G.: Creating New Medical Ontologies for Image Annotation. Springer, Berlin (2011)
17. Gonzales, R., Wintz, P.: Digital Image Processing. Addison-Wesley, Reading (1987)
18. Middleton, L., Sivaswamy, J.: Hexagonal Image Processing; A Practical Approach. Advances in Pattern Recognition. Springer, Berlin (2005)
19. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. MIT Press, Cambridge (1990)
20. Gijsenij, A., Gevers, T., Lucassen, M.P.: A perceptual comparison of distance measures for color constancy algorithms. In: Proceedings of the 10th European Conference on Computer Vision, pp. 208–221. Marseille, France (2008)
21. Sanfeliu, A., Alquézar, R., Andrade, J., Climent, J., Serratosa, F., Verges, J.: Graph-based representations and techniques for image processing and image analysis. Pattern Recogn. **35**(3), 639–650 (2001)
22. Burdescu, D.D., Brezovan, M., Stanescu, L., Spahiu, C.S.: A spatial segmentation method. Int. J. Comput. Sci. Appl. **1**(5), 75–100 (2014)
23. Burdescu, D.D., Stanescu, L., Brezovan, M., Spahiu, C.S.: Computational complexity analysis of the graph extraction algorithm for 3d segmentation. In: Proceedings of the IEEE Tenth World Congress on Services, pp. 462–470. Alaska, USA (2014)