A Hierarchical Image Segmentation Algorithm

Edwin Zarrazola[1,3], Javier Yáñez[1], Daniel Gómez[2], and Javier Montero[1,4]

[1]Facultad de Matemáticas, Universidad Complutense de Madrid, P de Ciencias 3, 28040 Madrid-Spain
[2]Facultad de Estudios Estadísticos, Universidad Complutense de Madrid, Avda. Puerta de Hierro s/n, 28040 Madrid-Spain.
[3]Instituto de Matemáticas, Universidad de Antioquia, Calle 67, Número 53-108, Medellín-Colombia
[4]Instituto de Geociencas (CSIC-UCM), P de las Ciencias 3, 28040 Madrid-Spain.
{edwzar,jayage,monty}@mat.ucm.es
dagomez@estad.ucm.es

Abstract. In this paper, we develop an efficient and polynomial hierarchical clustering (*unsupervised classification*) algorithm for images. The output of this algorithm shows the cluster evolution in a divisive way: since the first iteration in which all the pixels are in the same group until the last iteration in which all the pixels belong to a singleton cluster.

Keywords: clustering, graph-based segmentation, cluster evolution, image segmentation

1 Introduction

Image segmentation is one of the most important research areas in computer vision and its applications such as pattern recognition, medicine, robotic and industry. These techniques are well known for their utility and complexity.

In general, segmentation algorithms can be classified into the following classes: clustering, edge detection, regions, histogram thresholding and graph-based techniques. The graph-based image segmentation techniques can be categorized into two classes:

1. Supervised image segmentation methods: minimum cut / maximum flow models (see [3, 4, 14, 19]), random walk models (see [1, 8–10]).
2. Unsupervised image segmentation methods: minimum spanning tree (see [5]), normalized cut (see [12, 15–17]), isoperimetric graph partitioning (see [11]).

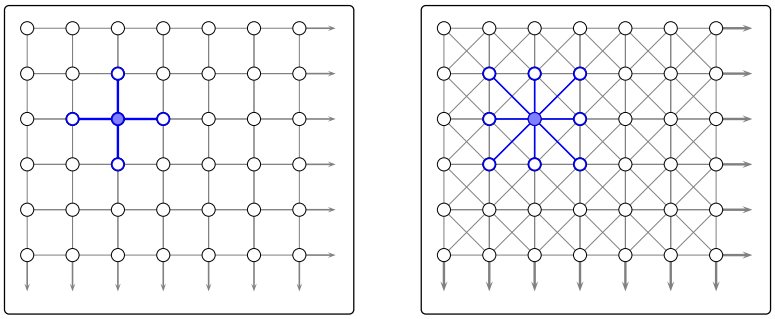
The algorithm shown in this article can be classified as an unsupervised graph-based image segmentation technique. The hierarchical approach defined here shares key elements of the segmentation algorithm proposed in [7] and extended in [6]. In this work we improve the computational complexity of these algorithms.

2 Preliminaries

Let $V = \{P_1, P_2, \ldots, P_n\}$ be the finite set of elements to be clustered. Let $E = \big\{\{P_a, P_b\} \mid P_a,\ P_b \in V\big\}$ be the set of non-ordered pairs of related (neighboring) items of V: if two elements $P_a,\ P_b \in V$ are related, then there exists an edge $e_{ab} = \{P_a, P_b\} \in E$; otherwise, $\{P_a, P_b\} \notin E$. Hence, we define a graph $G = (V, E)$ that shows the relations between the items. The graph G can be assumed to be connected; otherwise, its connected components must be analyzed separately.

Given $e_{ab} = \{P_a, P_b\} \in E$, let $d_{ab} \geq 0$ be the degree of dissimilarity between its endpoints P_a and P_b: the greater d_{ab} is, the more dissimilar P_a and P_b are. This measure is defined taking into account the specific problem and the characteristics of the elements considered. Its construction and properties are beyond the objectives of this paper.

Dealing with images, P_a can represent a single node or a set of nodes through some aggregator operator $\mathcal{A}$ (see [2]). One useful topology for images could be that one where a pixel is linked with four or eight neighbors (see Fig. 1).



(a) (b)

Fig. 1. Two topologies (networks) applied to images: (a) four neighbours, and (b) eight neighbours.

The choice of the 4-connectivity will be used in order to explain our approach. If $D = \big\{d_{ab} \mid e_{ab} \in E\big\}$, then the available information can therefore be summarized by the network

$$N = \big\{\, G\,;\, D \,\big\}. \tag{1}$$

3 Hierarchical Segmentation Algorithm

Given a binary-colored node $P_a \in V$, ($\mathrm{col}(P_a) \in \{0, 1\}$), the color that we assign to any adjacent node $P_b \in V$ depends on the measure $d_{ab} \in D$ compared with a prescribed threshold α:

$$\mathrm{col}(P_b) = \begin{cases} \mathrm{col}(P_a) & \text{if } d_{ab} < \alpha \\ 1 - \mathrm{col}(P_a) & \text{if } d_{ab} \geq \alpha \end{cases} \tag{2}$$

for all $e_{ab} \in E$. Now, for a given α_t, it says that nodes P_a and P_b lie on the same group (denoted $\mathrm{g}_t(P_b) = \mathrm{g}_t(P_a)$) if $e_{ab} \in E$ and $\mathrm{col}(P_b) = \mathrm{col}(P_a)$.

Let us observe that the binary coloring procedure given in (2) is not always consistent: a node can be colored differently depending on the choice of the chain (the sequence of adjacent edges) linking it with the first node. Inconsistent situations are therefore present when there is a cycle in the graph G and a node can be colored differently depending on the chain chosen, starting at the initial node. These cycles will be called *inconsistent* cycles.

An arbitrary spanning tree for a pixel network was defined in [6, 7]. Such a spanning tree allows a binary coloring of G, and inconsistent cycles appear when other edges are added. The number of such inconsistent cycles depends on the chosen spanning tree and the value of the threshold α. In [6, 7], when dealing with inconsistent cycles in real digital images, the large size of these pixel networks was taken into account, and a low ratio of inconsistent cycles was allowed without loss of quality of the segmentation procedure; but we now realize that the particular topology of the pixel network may allow interesting shortcuts in the arrangement of the pixels. Another approach to dealing with inconsistent cycles was introduced in [18], where the least inconsistent spanning tree was defined for any value of α; but this approach has a high computational cost and does not ensure consistency of the binary coloring procedure.

Given a network N as shown in (1) and a family of threshold values

$$\{\alpha_0 > \alpha_1 > \cdots > \alpha_{K-1} > \alpha_K\}.$$

Let $G^0 = G$; the following family of partial graphs is defined as:

$$\{\ G^t = (V, E^t)\ \} \tag{3}$$

where $t \in \{1, \ldots, K\}$, and

$$E^t = \left\{ e_{ab} \in E \ \middle|\ d_{ab} < \alpha_t \ \wedge\ \mathrm{g}_{t-1}(P_a) = \mathrm{g}_{t-1}(P_b) \right\}. \tag{4}$$

Note that $\mathrm{g}_{t-1}(P_a) = \mathrm{g}_{t-1}(P_b)$ in (4), means that P_a and P_b are in the same group to α_{t-1} (the before step). Now, for $t \in \{1, \ldots, K\}$, the family of partial graphs

$$\{\ F^t = (V, W^t)\ \} \tag{5}$$

is constructed following a two-step procedure:

1. Following a Kruskal-like algorithm, a partial graph $F^t = (V, W^t)$ of $G^{t-1} \setminus G^t = (V,\ E^{t-1} - E^t)$ is constructed: given the arrangement in decreasing order of the edges where weights are in the internal $[\alpha_t, \alpha_{t-1})$. These edges are introduced as long as their addition does not create a cycle.
2. If the partial graph F^t is a spanning forest of $G^{t-1} \setminus G^t = (V,\ E^{t-1} - E^t)$, the procedure stops; otherwise, the edges of E^t (edges with weights lower than α_t) are arranged in increasing order and are added iteratively to W^t so long as they do not form a cycle, and the process finishes when F^t becomes a spanning tree.

The connected components of F^t are neither maximum spanning trees nor minimum spanning trees because the order in which the edges are included in them is reversed between the two steps above. Indeed, in order for us to specify explicitly the algorithm outlined above, the order of the edges in the two-step procedure must take into account the case of ties in the weights of the edges; in this sense, the following remark is pertinent.

Inconsistent cycles can be distinguished into two types of cycles:

- Cycles where all edges have weights greater than or equal to the current threshold α_t. In this case, by the construction of F^t, the edge which generates the cycle has the greatest weight and, in the case of ties between some edges, is one of the last edges in the ordered list of edges. One way of breaking these ties is to consider, as a secondary criterion, the arrangement of the edges in decreasing order of the difference between the degrees of the endpoints of the edges: first, those edges with the greatest difference (in absolute value) between their endpoints.
- Cycles where an edge has a weight lower than the current threshold α. In this case, the cycle is generated by an edge which maintains its endpoints in the same cluster, and the arrangement of these edges is the opposite of that in the first case: first, those edges with the lowest difference (in absolute value) between their endpoints.

3.1 The threshold α

One of the inputs to the basic binary coloring procedure is the value of the threshold α; different values of α_t can produce different spanning forests F^t and, consequently, different colors of the nodes. The selection of these values is the keystone of the overall coloring algorithm.

It seems appropriate to follow an decreasing scheme for the values of the threshold α, with an appropriate selection of the parameter K and values α_t:

$$\alpha_0 > \overline{\alpha} = \alpha_1 > \alpha_2 > \cdots > \alpha_{K-1} \geq \underline{\alpha} > \alpha_K$$

where $\overline{\alpha} \equiv \max D$ and $\underline{\alpha} \equiv \min D$.

Similarly to the result of the binary coloring procedure, F^{t+1} and F^t will be colored differently only if there exists an edge $e_{ab} \in E$ such that $\alpha_{t+1} \leq d_{ab} < \alpha_t$. In this way, the number of different values for α is bounded by m, the number of edges in the graph; and $K \leq m$. In real situations, as the size of the graph increases, it is enough to take a value of K much lower than m.

3.2 The hierarchical segmentation of the network

In this part, we analyze the information obtained by the application of successive binary coloring procedures to the network N.

The successive binary coloring procedure previously introduced suggests a hierarchical partition process: the first clusters are defined by the connected

components of equally colored nodes, and any of them can be partitioned again by subsequent binary colorings. In this way, separated nodes cannot be joined again in this iterative process. Hence, we can understand that the earlier stages of binary coloring are more relevant than the later ones.

Note that the partitions that we produce are independent of the particular color (0 or 1) chosen, since in each iteration two nodes belonging to the same cluster will share the same cluster in the next iteration if they share the same color, independently of whether this color is 0 or 1.

If the number of successive binary colorings K is very large, the partition might, obviously, be uninformative. One way to solve this problem is to reduce the parameter K. This parameter must be chosen taking into account the requirements that, on the one hand, the partition must consider the different values of the distances and, on the other hand, the computational cost of obtaining the partition must be reduced. Moreover, after the parameter K has been fixed, the various values must satisfy the condition $\alpha_0 > \cdots > \alpha_K$. Both of these decisions depend on the size and characteristics of the problem.

Illustrative example.

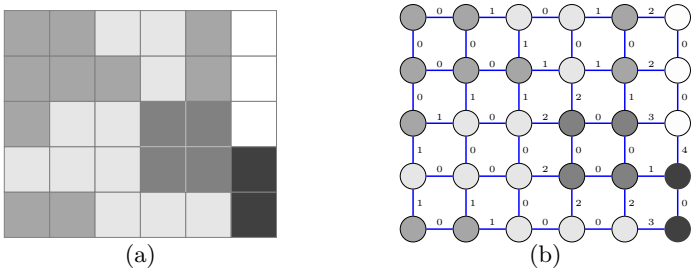
Consider a gray-scale image with the associate matrix:

$$P = \begin{bmatrix} 3\,3\,2\,2\,3\,1 \\ 3\,3\,3\,2\,3\,1 \\ 3\,2\,2\,4\,4\,1 \\ 2\,2\,2\,4\,4\,5 \\ 3\,3\,2\,2\,2\,5 \end{bmatrix} \tag{6}$$

where $P(i,j)$ in (6) represent some gray-scale measure getting homogeneous values $\{1,2,3,4,5\}$. Let

$$d\big((i,j),(i^*,j^*)\big) = \Big|P(i,j) - P(i^*,j^*)\Big|$$

be a Euclidean distance between gray-scale pixels for (i,j) and (i^*,j^*) in $[1,\ldots,5]\times[1,\ldots,6]$. A gray-scale image representation and its associate network to (6) are given in Fig. 2a and Fig. 2b respectively.



(a) (b)

Fig. 2. Gray-scale image (a), and network (b).

Consider a set of α-threshold $\{1,2,3,4\}$ (in decreasing order). The hierarchical segmentation to this set is shown in Fig. 3.

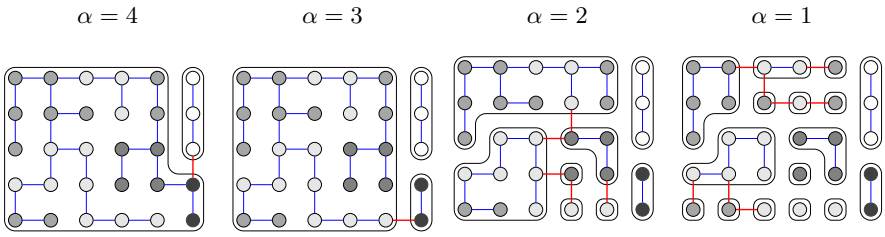


Fig. 3. Hierarchical segmentation by using four thresholds $\{1, 2, 3, 4\}$.

4 A Generalization of the Hierarchical Segmentation Algorithm

Often, we find networks that have very homogeneous regions and we want to remain compact for any threshold used in the segmentation. These groups quite similar (or similar adjacent nodes) are common in many images, so it is desirable to take into account these facts when applying a segmentation process. The hierarchical segmentation process we have developed in the Section 3 is readily adaptable to such events as shown below. Let α_τ be a given similarity threshold, and

$$\underline{E} = \{\, e_{ab} \in E \;\mid\; d_{ab} \leq \alpha_\tau \,\}. \tag{7}$$

If P_a and P_b are nodes such that $e_{ab} \in \underline{E}$; it is expected that $g_t(P_a) = g_t(P_b)$ for all $t \in \{1, \ldots, K\}$. Using (7), we define the set of edges

$$\underline{E}^t = E^t \setminus \underline{E} = \Big\{ e_{ab} \in E \;\mid\; \alpha_\tau < d_{ab} < \alpha_t \;\wedge g_{t-1}(P_a) = g_{t-1}(P_b) \Big\}. \tag{8}$$

If the given set of edges (4) is replaced by (8), a new hierarchical segmentation algorithm is defined. In the illustrative example, if $\alpha_\tau = 0$ the results are:

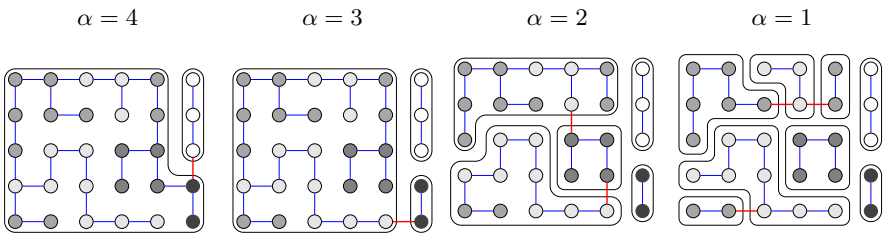


Fig. 4. Generalized Hierarchical Segmentation with $\alpha_\tau = 0$.

This algorithm generalizes the algorithm introduced in Section 3. Note that both algorithm produce the same output for $\alpha_\tau < \underline{\alpha}$.

This generalized algorithm avoids many trees which could separate similar groups of pixels and it produces a better partition, as it as follows.

We can define "adjacent groups" as those that have at least one neighbor pixel inside the other. Formally, for a given α_t, lets A_t and B_t be any two sets of pixels formed on the t-segmentation (note that, $P_a \in A_t \Leftrightarrow g_t(P_a) = A_t$, and $P_b \in B_t \Leftrightarrow g_t(P_b) = B_t$, for some $P_a, P_b \in V$). Now, it says that A_t and B_t are adjacent groups if $\big|A_t, B_t\big| > 0$, with

$$\big|A_t, B_t\big| = \text{card}\Big\{\, e_{ab} \in E \;\big|\; P_a \in A_t \;\wedge\; P_b \in B_t \Big\} \tag{9}$$

where "card" in (9) means the cardinality of the set. Note that, depending of a generated forest, it can be obtained a partitioned groups when some "adjacent" are very similar or dissimilar. One way to measure the quality of some α_t partition is through aggregation operators (see [2]), and measure how dissimilar the adjacent groups are.

Let $\mathcal{A}(C)$ be some aggregator operator defined on a set of pixels C (for instance, the average, median, max, min, among others), and let $H^{\alpha_t}_{\min}$ be the minimum distance between adjacent groups for a fixed $\mathcal{A}$, i.e;

$$H^{\alpha_t}_{\min} = \min_{|A_t,B_t|>0} \Big\{\, \|\mathcal{A}(A_t) - \mathcal{A}(B_t)\| \,\Big\}. \tag{10}$$

where $\|\cdot\|$ is a fixed distance. Note that the larger values of (10) mean more dissimilarity between groups.

In the above example, if we use the average and median as aggregator operators (i.e., $\mathcal{A}(C) = \bar{C}$ and $\mathcal{A}(C) = \tilde{C}$, respectively), the $H^{\alpha_t}_{\min}$ in (10) can be written as

$$\bar{H}^{\alpha_t}_{\min} = \min_{|A_t,B_t|>0} \Big\{ |\bar{A}_t - \bar{B}_t| \Big\} \tag{11}$$

$$\tilde{H}^{\alpha_t}_{\min} = \min_{|A_t,B_t|>0} \Big\{ |\tilde{A}_t - \tilde{B}_t| \Big\} \tag{12}$$

respectively; and then we can compare Hierarchical Segmentation Algorithm (HSA) vs Generalized Hierarchical Segmentation Algorithm (GHSA) as shown in Table 1.

	HSA			GHSA		
$\boldsymbol{\alpha}$	N	$\bar{H}^{\alpha_t}_{\min}$	$\tilde{H}^{\alpha_t}_{\min}$	N	$\bar{H}^{\alpha_t}_{\min}$	$\tilde{H}^{\alpha_t}_{\min}$
4	2	1.89	2.00	2	1.89	2.00
3	3	1.72	2.00	3	1.72	2.00
2	8	0.00	0.00	5	0.52	1.00
1	16	0.00	0.00	8	1.00	1.00

Table 1.

Note that in Hierarchical Segmentation Algorithm (HSA), $\bar{H}^{\alpha_t}_{\min}$ and $\tilde{H}^{\alpha_t}_{\min}$ defined in (11) and (12) are zero for $\alpha = 2, 1$; this means that there are adjacent groups whose average difference is zero (less quality partitions).

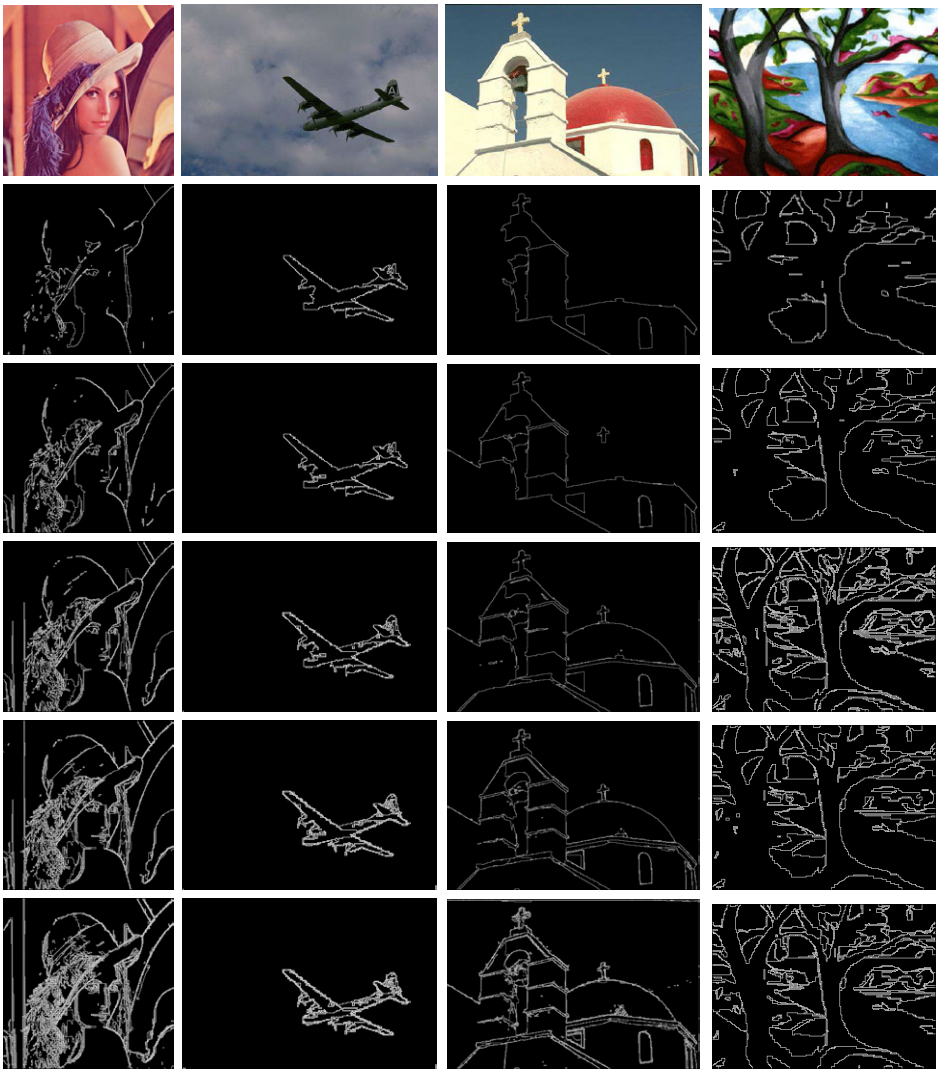


Fig. 5. Hierarchical segmentation process to 4 color images and 5 thresholds.

4.1 Some Hierarchical Segmentation in Color Images

Figure 5 shows the generalized hierarchical segmentation process to four color images to five thresholds and $\alpha_\tau = 1$.

In order to obtain a good measure of color distance, it is necessary to translate an RGB images to an another most uniform color space (i.e. CIELAB or CIELUV color spaces, see [13]). Let $P_1^{\text{rgb}} = (r_1, g_1, b_1)$ and $P_2^{\text{rgb}} = (r_2, g_2, b_2)$ be two colors on the RGB-space, its representation on the CIELAB space is given by $P_1^{\text{lab}} = (L_1, a_1, b_1)$ and $P_2^{\text{lab}} = (L_2, a_2, b_2)$ respectively. Now, one way to measure

the distance between P_1^{lab} and P_2^{lab} is

$$d_{1,2}^{\text{lab}} = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2} \tag{13}$$

where $\Delta L = L_2 - L_1$, $\Delta a = a_2 - a_1$ and $\Delta b = b_2 - b_1$. The distance given in (13) is known as CIE76, and is one of the definitions of distance between colors widely used in the world, given by the CIE (International Commission on Illumination) in 1976. Although currently CIE76 is still used in various applications, there have been several versions of corrections to (13) as are CIE94 and CIE2000 among others (see [13]). However, for our purposes CIE76 will be used as a good approximation for the measure.

5 Final Comments

One of the main problems when working on theories involving segmentation thresholds is to choose a set of appropriate thresholds for the segmentation process. If we want to provide the threshold set explicitly, we need to manipulate the pixel values in any color space, and the distances between them in a measure space. However, in our proposed algorithm is more important the number of cutting edges (edges with large values) included in a given step of segmentation, that the threshold value α_i. Because of this, the only thing that matters to hierarchical segmentation proposed is the number of steps (the K value), we can even say how many cutting edges are to be incorporated in every step.

In the processing of the images of Fig. 5, the CPU time was no more than 1 minute for each hierarchical image segmentation. The algorithm was run on a 1.7GHz Intel Core i5 with 3GB of RAM. The five thresholds have been chosen automatically by the algorithm in order to include a fixed size of cut edges ($100\,t^2$ cut edges are included on each step, for $t = 1, \ldots, 5$).

Finally, our proposed algorithm can be executed using coarse nets, which reduces the computational time substantially. The theory of aggregation functions can be used for this purpose, i.e., to relate sets of pixels through some aggregation function, so that results can be obtained more quickly and preserving a good classification. Performances of different aggregator functions in networs can be a future research work.

Acknowledgement.

This research has been partially supported by the Government of Spain, grant TIN2012-32482.

References

1. S. Andrews, G. Hamarneh and A. Saad. "Fast random walker with priors using precomputation for interactive medical image segmentation". *Lecture Notes in Computer Science*, *6363*, 9–16. (2010).

2. G. Beliakov, A. Pradera and T. Calvo. "Aggregation functions: a guide for practitioners". *Springer*, (2008).
3. Y. Boykov and M. P. Jolly. "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images". *Proceedings of International Conference on Computer Vision*, **1**, 105–112. (2001).
4. Y. Boykov and V. Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(9), 1124–1137. (2004).
5. P. F. Felzenszwalb and D. P. Huttenlocher. "Efficient graph-based image segmentation". *International Journal of Computer Vision*, **59**(2), 167–181. (2004)
6. D. Gómez, J. Montero and J. Yánez. "A coloring algorithm for image classification". *Information Sciences* **176**, 3645–3657 (2006).
7. D. Gómez, J. Montero, J. Yáñez and C. Poidomani. "A graph coloring algorithm approach for image segmentation". *Omega* **35**, 173–183 (2007).
8. L. Grady. "Random walks for image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(11), 1768–1783. (2006).
9. L. Grady, and G. Funka-Lea. "Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials". *Proceedings of the 8th ECCV Workshop on Computer Vision Approaches to Medical Image Analysis and Mathematical Methods in Biomedical Image Analysis*, 230–245. (2004).
10. L. Grady, and A. K. Sinop. "Fast approximate random walker segmentation using eigenvector precomputation". *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. (2008).
11. L. Grady and E. L. Schwartz. "Isoperimetric graph partitioning for image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(3), 469–475. (2006).
12. D. S. Hochbaum. "Polynomial time algorithms for ratio regions and a variant of normalized cut". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(5), 889–898. (2010).
13. N. Ohta and A. Robertson "Colorimetry. Fundamentals and Applications". *John Wiley & Sons, LTd.* (2005).
14. B. Peng and O. Veksler. "Parameter selection for graph cut based image segmentation". *British Machine Vision Conference*. (2008).
15. J. Shi and J. Malik. "Normalized cuts and image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8), 888–905. (2001).
16. Z. Wu and R. Leahy. "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(11), 1101–1113. (1993).
17. L. Xu, W. Li and D. Schuurmans. "Fast normalized cut with linear constraints". *IEEE Conference on Computer Vision and Pattern Recognition*, 2866–2873. (2009).
18. J. Yáñez, S. Muñoz and J. Montero. "Graph coloring inconsistencies in image segmentation". *Computer Engineering and Information Sciences* **1**, 435–440 (2008).
19. W. Yang, J. Cai, J. Zheng and J. Luo. "User-friendly interactive image segmentation through unified combinatorial user inputs". *IEEE Transactions on Image Processing*, **19**(9), 2470–2479. (2010).