ON IMAGE SEGMENTATION USING A COMBINATION OF FELZENSZWALB, SLIC AND WATERSHED METHODS

Alin-Florin MIHĂILĂ[1]
Patricia-Steliana PENARIU[2]
Giorgiana Violeta VLĂSCEANU[3]
Marcel PRODAN[4]

Abstract: *Image segmentation is an essential problem in Computer Vision and it is foundational to the development of next-generation information extraction methods, issued in problems of great interest, such as driving autonomous machines, text analysis, object identification, extracting information from images. Knowing that there are no perfect algorithms for image segmentation, this paper aims to achieve a method that combines the results of different algorithms through various voting schemes in the hope of getting better results.*

Keywords: *Computer vision, image segmentation, image processing, voting algorithms, SLIC, Watershed, Felzenszwalb*

1. Introduction

Image segmentation is a class of image processing algorithms that identify, and group pixels based on specific criteria. The segmentation process itself entails image processing algorithms that strategically isolate areas of interest within images: for example, image processing could use segmentation to direct self-driving machines in traffic. These algorithms generate certain limitations in terms of image quality. Their tendency to over- or under-segment makes it necessary to devise algorithmic combinations that can help maximize image segmentation results. A. Hoover et al. in [1] have outlined five categories of region classifications: correct detection, over-segmentation (many detections of a single surface, which can yield an incorrect topology), under-segmentation (not enough separation of multiple surfaces, results in a subset of the correct topology and a

[1] Engineer, Computer Science and Engineering Department, Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independentei 313, Bucharest 060042, Romania, alin.mihaila@stud.fils.upb.ro
[2] PhD Student, Eng., Computer Science and Engineering Department, Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independentei 313, Bucharest 060042, Romania, patricia.penariu@stud.acs.upb.ro, patriciapenariu@gmail.com
[3] Teaching Assistant, PhD Student, Eng., Computer Science and Engineering Department, Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independentei 313, Bucharest 060042, Romania, giorgiana.vlasceanu@cs.pub.ro
[4] PhD Student, Engineer, Computer Science and Engineering Department, Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independentei 313, Bucharest 060042, Romania, marcel.prodan@stud.acs.upb.ro

deformed geometry), missed (a segmenter fails to find a surface which appears in the image) and noise (the segmenter assumes the existence of a surface not in the image). Given that it is easier to merge segments to obtain bigger ones, as opposed to splitting large regions to yield the true segments, under-segmentation often gets more interest and attention dedicated to it than its counterpart, over-segmentation [2]. The figure below is a relevant example of highlighting the limitations of segmentation algorithms (Fig. 1).

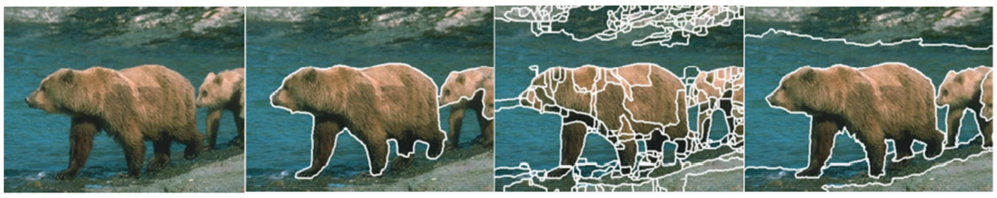

Fig. 1. Example of images over-segmented and under-segmented. From left to right: a) original image, b) under-segmentation, c) over-segmentation, d) ground truth image; image taken from [2]

This paper introduces a voting method that combines different algorithms in order to achieve a better performance in image segmentation than the results obtained with these methods taken separately. The selected approaches utilized for the proposed method are Felzenszwalb segmentation (based on graphs) [3], SLIC (Simple Linear Iterative Clustering) Superpixels [4][5] and Watershed [6][7], which will be described into the next section (Section 2A). These algorithms have been chosen to have partially different (over-segmented or under-segmented) results.

2. Proposed method

A. Segmentation algorithms

The first algorithm mentioned here is Felzenszwalb graph-based segmentation [3]. In this category of segmentation, each pixel from the image matches a node in the graph. The connection between the nodes represents the specific pairs of pixels that are neighbors. For this scenario, each edge has a correlated weight based on the characteristics of the groups of pixels. The graph-based method proposed by Felzenszwalb et al. in [3] takes edges from a graph. For this method, the pixel represents a node in the graph. The undirected edges are in correlation with neighboring pixels. The weights for every edge represent the measure of the variation among the pixels. This technique tunes the criteria of segmentation concerning the variability of the neighboring areas. The weights for every edge represent the measure of the variation among the pixels. This technique stands out in a few ways: it customizes the criteria of segmentation according to the variations in nearby areas and it reveals that the process can be rapacious in its tenacity to attain segmentation that mirrors global characteristics. The pseudocode of

Felzenszwalb graph-based segmentation algorithm consists of four steps. Firstly, a graph *Gph* with *n* vertices and *m* edges is defined as an input, *Gph = (Vert, Edg)*. The output is a segmentation of *Vert* into *Seg* components, *Seg* = ($C_1, \dots, C_z$). The first step is to sort *Edg* by non-decreasing edge weight into ς = *($e_1, \dots, e_m$)*. The second step is to start with a segmentation Seg^0, where each vertex $vert_i$ is in its own component. Then, construct Seg^p, given Seg^{p-1}, as follows: let $vert_i$ and $vert_j$ denote the vertices connected by the *p*-th edge in the ordering, $e_p = (vert_i, vert_j)$; if $vert_i$ and $vert_j$ are in disjoint components of Seg^{p-1} and $w(e_p)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing (step three). The *internal difference* of a component $C \subseteq Vert$is defined as the largest weight in the minimum spanning tree of the component *MinST*(*C*, *Edg*) that is *IntDiff(C),* shown into the following equation (eq. 1):

$$IntDiff(C) = max_{e \epsilon MinST(C,Edg)} w(e) \quad (1)$$

The *difference between* two components *C1, C2* ⊆ *Vert* is defined as the minimum weight edge connecting the two components (eq. 2):

$$Diff(C_1, C_2) = min_{vert_i \epsilon C_1,\, vert_j \epsilon C_2,\, (vert_i, vert_j) \epsilon\, Edg} w((vert_i, vert_j)) \quad (2)$$

The *minimum internal difference*, *MinIntDiff*(eq. 3), is defined as:

$$MinIntDiff(C_1, C_2) = min(IntDiff(C_1) + t(C_1), Int(C_2) + t(C_2)) \quad (3)$$

where *t(C)* is a threshold function (eq. 4) based on the size of the component, |*C*| denotes the size of *C* and *k* is some constant parameter:

$$t(C) = k/|C| \quad (4)$$

Step three is repeated for *p=1, . . . ,m*. The algorithm ends with step four which involves returning $S = Seg^m$.

The second algorithm presented is SLIC Superpixels. Superpixels group similar pixels into atomic areas of pixels which can be used as entities in algorithms in order to reduce the number of primitives for some processing. This method addresses another way of looking at the classic pixel grid. Achanta et al. mention in [4] that this approach reduces the complexity, and the primitive used helps produce the needed features of the image. In terms of requirements, Stutz et al. summarize in [5] a list of general specifications agreed by [4][8][9][10] for the superpixel: partition (the possibility to determine a subdivision for the image), connectivity (describe an associated collection of pixels); boundary adherence; compactness, regularity, and smoothness (there are compact not in the presence of image borders), efficiency (the generating process must be the most efficient), controllable number of superpixels.

Achanta et al. [11] propose an algorithm that produces superpixels by clustering pixels. This operation creates the superpixels that start with areas of the same colors that are in proximity to the image level, in the combined five-dimensional color and image space. In this way, this method generates uniform and compact superpixels. This technique is an extension of the K-means algorithm. The K-means method groups the points in clusters based on a similarity function. The most used similarity function is the Euclidean distance. The algorithm presented by Achanta et al. differs from K-means by two characteristics. The first one is about the optimization step and, most accurately, about the number of distance calculations. By narrowing the search space to an approximate area of the superpixel, the number previously mentioned is decreased. The second one is about the weighted distance measure. In this step, proximity measurements are done (color and spatial) giving access to the dimension of superpixels. The SLIC Superpixels algorithm implies several stages: the first stage consists of initializing clusters centers by sampling pixels at regular grid steps S, $C_k=\{l_k, a_k, b_k, x_k, y_k\}^T$ while in the second stage the centers of the clusters are perturbed in an $n\times n$ neighborhood, to the lowest gradient position. Then repeat until $E \leq threshold$ the following: for each cluster center C_k assign the best matching pixels from $2S\times 2S$ square neighborhood around the cluster center according to the distance measure D_s (eq. 5), then compute new cluster centers and residual error E {*L1* distance between previous centers and recomputed centers}. The algorithm is completed by the last stage which consists of enforcing connectivity. Distance measure D_s is defined as follows:

$$\begin{gathered} d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \\ d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \\ D_s = d_{lab} + \frac{m}{S} d_{xy} \end{gathered} \qquad (5)$$

where D_s is the sum of the lab distance and the xy plane distance normalized by the grid interval S. A variable m is introduced in D_s to control the compactness of a superpixel. The greater the value of m, the more spatial proximity is emphasized and the more compact the cluster. This value can be in the range [1; 20].

The third algorithm considered is Watershed. Watershed is a region-based segmentation method. The first approach is originally in mathematical morphology and was introduced by Digabel and Lantuejoul in 1978 [6]. In this method, the image is seen as a topographic scene [7] with ridges and valleys. The gray values of the pixels or their gradient magnitude define the rise values of the landscape. Preim et al. [12] call that the watershed technique decomposes an image into "catchment basins". Another view on the technique represents each valley as a relief [7], and this relief flood from its minima when two areas join. After joining, it creates a dam. All dams created represent this method. This representation resembles the flooding process [13]. The main aim is to divide the image into regions of interest. The algorithm needs only the pixel intensity to obtain these regions. An example of the watershed algorithm can be seen in the figure below (Fig. 2).

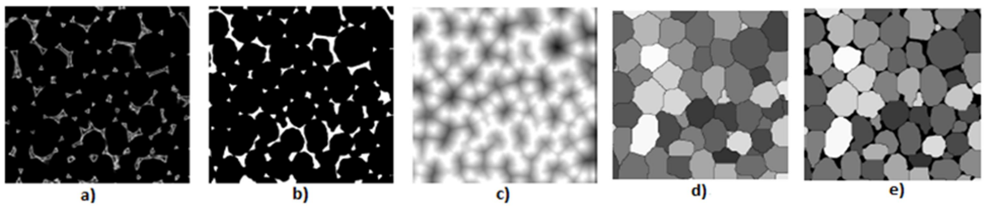

Fig. 2. Watershed segmentation - a) original scan, b) segmented and closed edge system, c) inverted distance map of the back-ground, d) watershed transformation of the distance map, e) reconstructed cells; image taken from [14]

B. Proposed voting algorithm

There is a series of approaches for image segmentation. Each one has downsides or upsides corresponding to the input data. Methods such as model-based, central clustering, graph theoretical based, nonparametric density estimation based methods, empirical and hybrid approaches, and square-error clustering, present a view on some elements on the input data, by grouping or making clusters based on a set of properties [15]. In the current scenario, no single technique could manage all types of clusters. Studies in the last decade show that fusing classifiers helps to increase the accuracy and the diversity of input data [16]. The combining process of the methods' decisions is reviewed, and this technique is met in studies as the voting technique. The method proposed in this article involves two options for voting using the algorithms presented in Section 2.A. The first voting technique is the use of democratic voting, where each algorithm has an equal weight in the segmentation decision. This method has been proposed as a starting point in exploring voting techniques. Voting is represented as a linear function, and each algorithm has equal weight, as can be seen in the equation below (eq. 6).

$$V = \frac{1}{N}(Fz + SLIC + Watershed) \tag{6}$$

This algorithm was tested on several images and has been shown to have weaker results than the independent running of each algorithm separately. Another approach of this method is for each algorithm's vote to come with a weight (eq. 7). The weights are estimated from successive runs, and weights that provided performance were kept for the final vote.

$$V = w_{Fz} * Fz + w_S * SLIC + w_W * Watershed \tag{7}$$

3. Results

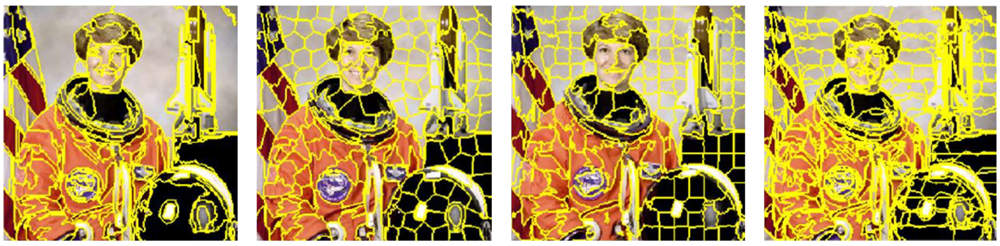
Fig. 3. Astronaut image. From the left to right: a) Felzenszwalb segmentation, b) SLIC segmentation, c) Compact watershed, d) Voting method.

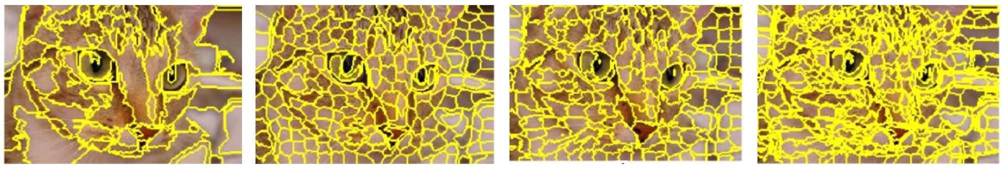
Fig. 4. Chelsea image. From left to right: a) Felzenszwalb segmentation, b)SLIC segmentation, c) Compact watershed, d) Voting method

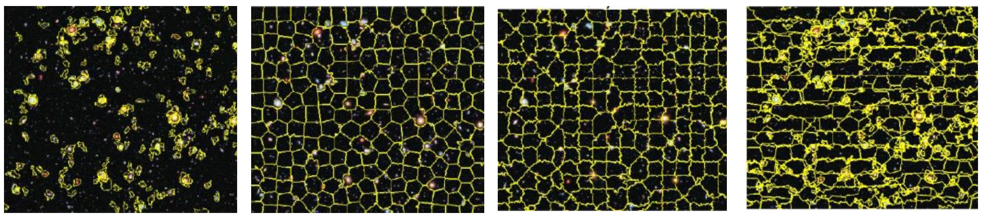
Fig. 5. Hubble deep field image. From left to right: a) Felzenszwalb segmentation, b) SLIC segmentation, c) Compact watershed, d) Voting method.

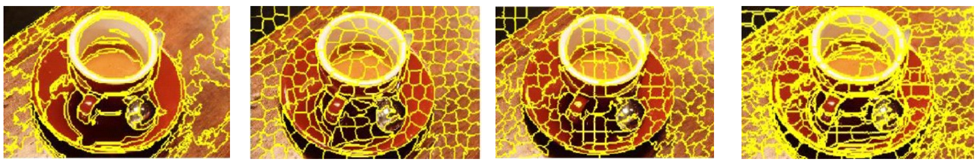
Fig. 6. Coffee image. From left to right: a) Felzenszwalb segmentation, b) SLIC segmentation, c) Compact watershed, d) Voting method.

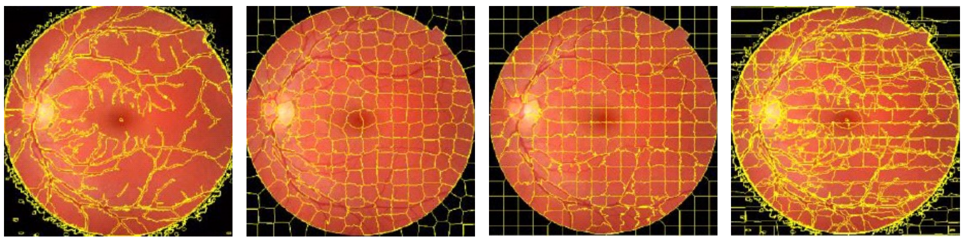
Fig. 7. Retina image - a) Felzenszwalb segmentation, b) SLIC segmentation, c) Compact watershed, d) Voting method

Image	**Number of segments**			
	Felzenszwalb	***SLIC Superpixels***	***Watershed***	***Voting method***
Astronaut	194	190	256	198
Chelsea	86	237	247	117
Hubble deep field	181	212	255	188
Coffee	109	223	260	134
Retina	295	226	256	283

Table 1. The number of segments for each of the images presented in Fig. 3 – Fig. 7. (Astronaut, Chelsea, Hubble deep field, Coffee and Retina) obtained with Felzenszwalb, SLIC Superpixels, Watershed algorithms and the proposed voting method.

In this section are highlighted the differences between the individual algorithms and the proposed weighted voting algorithm. The results obtained with Felzenszwalb, SLIC Superpixels, Watershed, and the proposed method, are shown in figures 3 to 7., representing different samples of images (Astronaut, Chelsea, Hubble deep field, Coffee and Retina) and their corresponding obtained segments (shown in Table 1) after applying these methods.

4. Conclusion

In this article, different voting techniques have been explored using image segmentation algorithms, and from the obtained results, one can observe the influences of each algorithm on voting. For future work, Deep Learning algorithms can be applied to combine results similarly. Because the algorithms presented in this article are in the class of unsupervised algorithms, they cannot be combined with supervised algorithms, such as Deep Learning.

Integrating the proposed voting-based method with another similarly-designed systems [17-19] into an unsupervised document image processing system is the main goal of the future development of this research.

Acknowledgement

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0689/„Lib2Life – Revitalizarea bibliotecilor și a patrimoniului cultural prin tehnologii avansate” / "Revitalizing Libraries and Cultural Heritage through Advanced Technologies", within PNCDI III.

References

[1] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D.B. Goldgof and R.B. Fisher, *An experimental comparison of range image segmentation algorithms*, in IEEE transactions on pattern analysis and machine intelligence, volume 18, issue 7, pp. 673-689, DOI: 10.1109/34.506791, July 1996.

[2] J. Sigut, F. Fumero, O. Nuñez, *Over-and under-segmentation evaluation based on the segmentation covering measure*, 23rd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2015, in volume: Short papers proceedings, pp. 83-89, 2015.

[3] P.F. Felzenszwalb, D. P. Huttenlocher, *Efficient graph-based image segmentation*, International Journal of Computer Vision, volume 59, issue 2, pp. 167-181, DOI: 10.1023/B:VISI.0000022288.19776.77, September 2004.

[4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, *SLIC superpixels compared to state-of-the-art superpixel methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 34, issue 11, pp. 2274-2282, DOI: 10.1109/TPAMI.2012.120, November 2012.

[5] D. Stutz, A. Hermans, B. Leibe, *Superpixels: An evaluation of the state-of-the-art*, in Computer Vision and Image Understanding, Volume 166, pp. 1-27, DOI: 10.1016/j.cviu.2017.03.007, January 2018.

[6] H. Digabel, C. Lantuejoul, *Iterative Algorithms*, in Proceedings of the 2nd European Symposium Quantitative Analysis of Microstructures in Material Science, Biology and Medicine, pp. 85-89. 1978.

[7] A.S. Kornilov and I.V. Safonov, *Review: An overview of watershed algorithm implementations in open source libraries*, in Journal of Imaging, volume 4, issue 10, pp. 123, DOI: 10.3390/jimaging4100123, October 2018.

[8] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, K. Siddiqi, *TurboPixels: Fast superpixels using geometric flows*, IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 31, issue 12, pp. 2290–2297, DOI: 10.1109/TPAMI.2009.96, December 2009.

[9] M. Y. Lui, O. Tuzel, S. Ramalingam, R. Chellappa, *Entropy rate superpixel segmentation*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, USA, pp. 2097–2104, DOI: 10.1109/CVPR.2011.5995323, 20-25 June 2011.

[10] A. Schick, M. Fischer, R. Stiefelhagen, *Measuring and evaluating the compactness of superpixels*, in Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pp. 930–934, Tsukuba, Japan, 11-15 November 2012.

[11] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, *SLIC superpixels,* EPFL Technical Report 149300, June 2010.

[12] B. Preim, C. Botha, *Visual Computing for Medicine (Second Edition),*inChapter 4 - Image Analysis for Medical Visualization, pp. 111-175, DOI: 10.1016/B978-0-12-415873-3.00004-3, 2014.

[13] L.J. Belaid and W. Mourou, *Image segmentation: a watershed transformation algorithm*, in Image Analysis & Stereology, volume 28, issue 2, pp. 93-102, DOI: 10.5566/ias.v28.p93-102, 2009.

[14] B. Wagner, A. Dinges, P. Müller & G. Haase, *Parallel volume image segmentation with watershed transformation*, in Scandinavian Conference on Image Analysis (SCIA 2009), pp. 420-429, Springer, Berlin, Heidelberg, DOI:10.1007/978-3-642-02230-2_43, June 2009.

[15] A. Fred, *Finding consistent clusters in data partitions*, in International Workshop on Multiple Classifier Systems (MCS2001), pp. 309-318, Springer, Berlin, Heidelberg, DOI: 10.1007/3-540-48219-9_31, July 2001.

[16] Costin-Anton Boiangiu, Radu Ioanitescu, *Voting-Based Image Segmentation*, in The Proceedings of Journal of Information Systems & Operations Management, Volume 7 No. 2/December 2013 (Journal of Information Systems, Operations Management), pp. 211-220, 2013.

[17] Costin-Anton Boiangiu, Radu Ioanitescu, Razvan-Costin Dragomir, *Voting-Based OCR System*, The Journal of Information Systems & Operations Management, Vol. 10, No. 2, 2016, pp. 470-486.

[18] Costin-Anton Boiangiu, Mihai Simion, Vlad Lionte, Zaharescu Mihai, *Voting Based Image Binarization*, The Journal of Information Systems & Operations Management, Vol. 8, No. 2, 2014, pp. 343-351.

[19] Costin-Anton Boiangiu, Paul Boglis, Georgiana Simion, Radu Ioanitescu, *Voting-Based Layout Analysis*, The Journal of Information Systems & Operations Management, Vol. 8, No. 1, 2014, pp. 39-47.